Natural

Editors

Version 9.2.5

October 2026

ADABAS & NATURAL

This document applies to Natural Version 9.2.5 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1979-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: NATMF-NNATEDITORS-925-20261009

Table of Contents

Preface ... ix
1 About this Documentation ... 1
Document Conventions ... 2
Online Information and Support ... 2
Data Protection ... 3
I Disabled Natural Editors ... 5
2 Disabled Natural Editors ... 7
II NaturalONE as the Default Development Environment ... 9
3 NaturalONE as the Default Development Environment ... 11
III Locking of Source Objects ... 13
4 Locking of Source Objects ... 15
Source Editing ... 16
Saving Objects ... 17
Unlocking Objects ... 17
Moving, Deleting, Renaming and Replacing Objects ... 18
IV Editors - General Information ... 19
5 Editors - General Information ... 21
Split-Screen Mode ... 22
Editor Profile ... 25
V Program Editor ... 35
6 Program Editor ... 37
Invoking the Program Editor ... 38
Editor Command Line ... 39
Top Information Line ... 40
Editing Area ... 40
Bottom Information Line ... 42
Editor Commands ... 42
Editor Commands for Positioning ... 50
Line Commands ... 51
Special PF-Key Functions ... 53
Cursor-Sensitive Commands ... 56
Saving and Cataloging Sources ... 57
Exit Function ... 58
VI Data Area Editor ... 61
7 Data Area Editor ... 63
Invoking the Data Area Editor ... 64
Top Information Line ... 65
Editor Command Line ... 66
Bottom Information Line ... 67
Using the Editing Area ... 67
Columns in the Editing Area ... 68
Extended Field Definition Editing ... 72
Line Commands ... 77

Editor Commands ... 83
Editor Commands for Positioning ... 87
Storing and Cataloging a Data Area ... 88
User Exit for the Data Area Editor ... 89
Exit Function ... 90
VII Map Editor ... 91
8 Components of the Map Editor ... 93
9 Summary of Map Creation ... 95
Step 1 - Defining a Map Profile ... 96
Step 2 - Defining a Map ... 96
Step 3 - Defining Map Fields ... 96
Step 4 - Saving a Map Definition ... 97
10 Invoking and Leaving the Map Editor ... 99
Invoking the Editor ... 100
Leaving the Editor ... 102
11 Functions in the Edit Map Menu ... 103
Field and Variable Definitions ... 104
Edit Map ... 105
Initialize New Map ... 106
Initialize a New Help Map ... 106
Maintenance of Profiles & Devices ... 106
Save Map ... 107
Test Map ... 107
Stow Map ... 107
Help ... 107
12 Initializing a Map ... 109
Delimiters ... 110
Format ... 112
Context ... 114
Filler Characters ... 115
13 Editing a Map ... 117
Screen Modes ... 118
PF Keys and Commands for Positioning ... 118
Line Commands ... 119
Field Commands ... 121
14 Defining Map Fields ... 125
Defining Fields Directly on the Screen ... 126
Selecting Definitions from Other Objects ... 126
Using System Variables in a Map Definition ... 131
15 Extended Field Editing ... 133
Invoking and Terminating Extended Field Editing ... 134
Fields in the Extended Field Editing Area ... 135
Fields in the Extended Text Field Editing Area ... 139
16 Post Assignment of Fields ... 141
17 Array and Table Definitions ... 143

Array Definition ... 144
Table Definition ... 148
18 Processing Rules ... 153
Field-Related Processing Rules ... 154
Function-Key-Related Processing Rules ... 155
Processing Rule Editing ... 155
VIII Map Editor Tutorial ... 161
19 Opening the Map Editor ... 163
20 Creating, Positioning and Deleting Map Fields ... 169
Creating and Centering Fields ... 170
Moving Fields ... 175
Deleting Fields and Inserting Lines ... 190
21 Testing and Saving a Map ... 201
22 Defining Processing Rules ... 205
23 Naming Fields and Saving/Cataloging a Map ... 215
24 Defining Field Properties ... 219
25 Creating and Testing a Help Map ... 225
26 Invoking a Map with INPUT USING MAP ... 231
27 Creating a Map for WRITE and Copying Field Definitions ... 235
28 Reusing the Layout of a Map ... 243
29 Invoking a Map with WRITE USING MAP ... 247
IX SYSDDM Utility ... 251
30 Principles of Operation ... 253
Storing DDMs ... 254
Restrictions of Use ... 255
31 Invoking and Terminating SYSDDM ... 257
32 Using SYSDDM Maintenance and Service Functions ... 259
Help on Functions ... 260
Performing a Function ... 260
Description of Functions ... 261
DDM Specification ... 264
33 Creating DDMs ... 267
34 Invoking and Terminating the DDM Editor ... 271
Invoking the Editor ... 272
Terminating the Editor ... 273
35 Using the DDM Editor ... 275
DDM Header Information ... 276
Columns of Field Attributes ... 277
Commands for Editing and Function Execution ... 282
Specifying Extended Field Attributes ... 288
36 Cataloging a DDM ... 291
Instructions ... 292
37 Listing DDMs ... 293
38 Maintaining DDMs in Different Environments ... 295
X Software AG Editor ... 297

39 General Information on the Software AG Editor 299
40 Invoking the Software AG Editor 301
41 Using the Editor Screen 303
Using the Input Areas 304
Scrolling the Data in the Editing Area 305
Locating a Line 307
Showing or Hiding Lines 308
Displaying Boundaries, Tab and Column Positions 309
42 Using Commands 311
Executing a Command 312
43 Creating and Modifying Data 315
Inserting and Deleting Lines 316
Copying, Moving, Overlaying and Repeating Lines 317
Copying or Moving a Window with Data 321
Setting Horizontal and Vertical Boundaries 328
Defining a Mask Line 330
Ordering Data between Boundaries 331
Centering Data 332
Justifying Data 332
Using the Physical or Logical Tabulator 335
Sorting Lines in Alphabetical Order 340
Finding a Character String 340
Replacing a Character String 344
44 Setting the Editor Profile 347
Editor Profile Items 349
45 Storing Data and Leaving the Software AG Editor 351
46 Summary of Line Commands 353
47 Summary of Main Commands 359
ADVANCE 361
AORDER 361
AUTOREN 362
AUTOSAVE 362
BNDS 362
BOTTOM 363
CANCEL 363
CAPS 363
CENTER 364
CHANGE 364
COLS 367
CURSOR 368
CWINDOW 368
DELETE 368
DOWN 370
DWINDOW 370
EMPTY 371

END 371
ESCAPE 371
EXCLUDE 372
FIND 373
FIX 376
HEX 376
INCLUDE 377
JLEFT 377
JRIGHT 378
JUSTIFY 379
LABEL 379
LC 380
LEFT 382
LIMIT 382
LOCATE 382
LOG 383
MASK 384
MWINDOW 384
NULLS 384
ORDER 385
POWER 385
PROFILE 386
PROTECT 386
RCHANGE 387
RECOVERY 387
RENUMBER 387
RESET 388
RFIND 388
RIGHT 388
SORT 389
TABS 389
TOP 390
UC 391
UNDO 391
UNREN 392
UP 392
WINDOW 392
XSWAP 393
Common Command Options 393
Index 397

Preface

This documentation describes all editors available in Natural.

For a tutorial on using the editors, see the *First Steps* documentation.

For information on Unicode and code page support for Natural editors, see *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation.

The *Editors* documentation is organized under the following headings:

Disabled Natural Editors	Notes on disabled editors previously available in versions up until 9.1.
NaturalONE as the Default Development Environment	Provides description about NaturalONE as the default development environment and related edit functions.
Locking of Source Objects	Describes the locking and unlocking of Natural source objects.
Editors - General Information	Contains an overview of which Natural objects are edited with which Natural editor. In addition, it contains information on split-screen mode and the editor profile.
Program Editor	Describes the program editor which is used to create and modify Natural programs, subprograms, subroutines, classes, copycodes, helproutines, functions and text objects.
Data Area Editor	Describes the data area editor which is used to create and modify local, global and parameter data areas.
Map Editor	Describes the map editor which is used to create and modify maps (screen layouts).
Map Editor Tutorial	Contains a series of tutorial sessions that introduce you to the use of the Natural map editor.
SYSDDM Utility	Describes the SYSDDM utility which is used to create, maintain and delete Natural data definition modules (DDMs).
Software AG Editor	Describes the Software AG Editor which is used to edit objects in Natural and other products.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Disabled Natural Editors

2 Disabled Natural Editors

Since Natural Version 9.1, the following Natural editors have been disabled (deactivated) by default in a local z/OS and Linux environment:

- Program editor
- Data area editor
- Map editor

This change does not affect the following:

- DDM editor (SYSDDM utility)
- Software AG Editor/Natural ISPF
- `EDT` system command for the line editor
- Batch mode processing (batch programs will run as usual)
- Predict `EDIT DESCRIPTION` system command
- Super Natural `EDIT REPORT` system command
- APIs and user exits that access editor interfaces

You can use the `TECH` system command (or corresponding API `USR2026N`) and the `*EDITOR` system variable to find out whether Natural editors are disabled in the current Natural environment.

With Natural version 9 on z/OS, Linux and Cloud, NaturalONE, and the Natural Development Server are integrated into Natural. The Natural editors (program editor, data area editor and map editor) are disabled. As NaturalONE and Natural Development Server licenses are integrated with Natural, they can be activated with the Natural (NAT) license key. This does not apply to Natural for Windows.

However, an environment with a valid Natural ISPF license, allows for using all Natural editors.

If Natural editors are disabled in the current Natural environment, the editor-related functions of the following commands and features are no longer supported and a corresponding error message is returned when you request an edit function:

- `EDIT` system command,
- `LIST` system command,
- `SCAN` system command,
- `MAINMENU` system command,
- SYSEXT utility.

II NaturalONE as the Default Development Environment

3 NaturalONE as the Default Development Environment

Since Natural Version 9.1 for z/OS and Linux, NaturalONE is the default development environment. The Natural Development Server is integrated into Natural as well.

You can activate both NaturalONE and Natural Development Server with the Natural (NAT) license key.

Note: This does not apply to Natural for Windows.

With NaturalONE the edit functionality is provided in a modern Eclipse environment. The former "character interface" or "green screen" editors (program editor, data area editor, and map editor) in a local z/OS or Linux environment are disabled.

However, an environment with Natural ISPF installed allows you to use all Natural editors.

You can use the `TECH` system command (or corresponding API `USR2026N`) and the `*EDITOR` system variable to find out whether Natural editors are disabled in the current Natural environment.

This change does not affect the following:

- DDM editor (SYSDDM utility)
- Software AG Editor
- `EDT` system command for the line editor
- Batch mode processing (batch programs will run as usual)
- Predict `EDIT DESCRIPTION` system command
- Super Natural `EDIT REPORT` system command
- APIs and user exits that access editor interfaces

If Natural editors are disabled in the current Natural environment, the editor-related functions of the following commands and features are no longer supported and a corresponding error message is returned when you request an edit function:

- `EDIT` system command
- `LIST` system command
- `SCAN` system command
- `MAINMENU` system command
- SYSEXT utility.

III Locking of Source Objects

4 Locking of Source Objects

- Source Editing 16
- Saving Objects 17
- Unlocking Objects 17
- Moving, Deleting, Renaming and Replacing Objects 18

Natural provides locking mechanisms that prevent concurrent updating of Natural source objects. These mechanisms allow the locking of source objects that are edited in a local z/OS environment and/or in a SPoD (Natural Single Point of Development) environment connected to a z/OS server.

You can activate or deactivate different object locking mechanisms by using the profile parameter `SLOCK` (see also the *Parameter Reference*):

- **Locking in Local and SPoD Environments**
  `SLOCK=PRE` activates locking of source objects that are edited either locally or in a SPoD environment, or using Natural ISPF, or in mixed environments.

  `SLOCK=PRE` is the recommended setting when working in mixed environments.
- **Locking in SPoD Environments**
  The default setting `SLOCK=SPOD` activates object locking only in a SPoD environment. A source object is then only locked when it is edited using Natural Studio. For further information, see *Object Locking* in the section *Remote Development using SPoD* in the *Natural for Windows* documentation.

  In a SPoD environment, `SLOCK=SPOD` provides compatibility with previous Natural Versions that supported locking under SPoD.
- **Checking for Latest Modification**
  When setting `SLOCK=POST`, the source object which is being edited can be read into the source work area and modified by multiple users. However, only the user who saves a modification first can update the source object. This is done by comparing the time stamp of the source object stored in the database with the time stamp of the source object when it is read into the source work area. All other users receive appropriate error messages when trying to save the source. This is not compatible with the SPoD locking concept of previous Natural versions.
- **Locking Deactivated**
  `SLOCK=OFF` deactivates all locking mechanisms.

The principles of object locking with profile parameter `SLOCK=PRE` set in a local environment are described in the following sections.

Source Editing

The sources of the following types of Natural object are locked while they are being edited with the appropriate Natural editor:

- Program
- Subprogram
- Subroutine
- Copycode

- Helproutine
- Text
- Map
- Local data area
- Global data area
- Parameter data area
- DDM (data definition module)
- Function

When you invoke a Natural editor, the source contained in the source work area will be locked. If the source object you want to edit has already been locked by another user (as indicated by an appropriate message), the source can be displayed in the editor but without an object name at the top of the screen. If you modify the source and want to keep the modifications, you have to save the source as a new source object with a new name.

Note: Reading in a source into the source work area by using the `READ` command does *not* lock the source object. A source object is only locked when you invoke the Natural editor.

Saving Objects

You cannot save (`SAVE` and `STOW` commands) a source that is being locked by another user.

Unlocking Objects

A locked source object contained in the source work area is unlocked when you do any of the following:

- clear the source work area,
- read in the source of another source object into the source work area,
- log on to another library,
- terminate the Natural session,
- leave the Natural editor while the **Leave Editor with Unlock** option is set in the editor profile (see also *General Defaults* in *Editor Profile* in the section *General Information*). This option determines whether the source contained in the current source work area is unlocked when leaving the editor. This also applies to maps and DDMs.

You can use the `UNLOCK` system command (see the *System Commands* documentation) to view locked source objects or unlock them if required.

Moving, Deleting, Renaming and Replacing Objects

Object locking is also considered when using the system command `DELETE` or `RENAME`, the Object Handler or the utility SYSMAIN.

When you move, delete, rename or replace a source object, the locking state of the object is checked:

- When the source object is locked, command execution is rejected.
- When the source object is not locked, the command is executed.

Restrictions

The utilities SYSRPC, SYSPARM and SYSERR do not support object locking.

IV Editors - General Information

5 Editors - General Information

- Split-Screen Mode 22
- Editor Profile 25

This section gives an overview of which Natural objects are edited with which Natural editor. In addition, it contains information on split-screen mode and the editor profile.

You invoke a Natural editor with the system command `EDIT` as described in the *System Commands* documentation and in *Creating and Editing an Object* in the *Using Natural* documentation. For the names to be used when editing or saving an object, see *Object Naming Conventions* in the *Using Natural* documentation.

Which editor is invoked depends on the type of object you want to edit:

- Programs, subprograms, subroutines, helproutines, classes, copycode, text objects and functions are created and edited in the **program editor**.
- Global data areas, local data areas and parameter data areas are created and edited in the **data area editor**.
- Maps and help maps are created and edited in the **map editor**.
- Predict descriptions are edited in the Predict description editor (see the *Predict* documentation).

Note: The Natural program, data area, and map editor have been disabled in your environment by default. For more information, see ***NaturalONE as the Default Development Environment*** .

An online help system is provided with each editor.

Tutorials which introduce you to the main features of the editors are provided in *First Steps* and in the ***Map Editor Tutorial***.

In addition to the Natural editors, the Software AG Editor is used by several Natural utilities and other products. You can also use the Software AG Editor as an alternative to the Natural program editor.

Split-Screen Mode

You can use all three Natural editors in split-screen mode: you can use one half of the screen for editing an object and at the same time have another Natural object displayed in the other half. Split-screen mode can be used to display a view (DDM; Data Definition Module), a data area, a Predict program description or a Natural program in the lower half of the screen. In addition, you can include items shown in the display section of the screen into the editing section that is, into the object you are currently editing.

Example:

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):

```
>                                        > +  Program     SAGDEMO  Lib SAGTEST

  Top    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.
    0010 DEFINE DATA LOCAL USING L-INVOIC
    0020             LOCAL USING L-INV-LN
    0030 END-DEFINE
    0040 *
    0050 READ INVOICE-VIEW BY INVOICE-NO FROM 1
    0060 *
    0070   FIND INVOICE-LINE-VIEW WITH INVOICE-NO = INVOICE-NO (0050)
    0080     DELETE
    0090   END-FINE
    0100 *
         ....+....1....+....2....+....3....+....4....+....5....+.. S 16   L 1
  Split All       Local       L-INVOIC  Library SAGTEST
    0010 V 1 INVOICE-VIEW                          INVOICE
    0020   2 CUST-NO                         N   8
    0030   2 INVOICE-NO                      N   8
    0040   2 DATE                           A   8
    0050   2 AMOUNT                         N 9.2
    0000
    0000
    0000
    0000
```

Split-Screen Commands

The following commands can be used to display and position an object in split-screen mode when using the program editor or the data area editor. For instructions on displaying objects in split-screen mode with the map editor, see ***Selecting Data Definitions*** in the *Map Editor* documentation.

All commands begin with an `S` or with `SPLIT` to indicate split screen mode. The `SPLIT` command is a **cursor-sensitive command** as described in the section *Program Editor*.

In the following table, an underlined text portion represents an acceptable command abbreviation.

Command	Function
SPLIT ++	Position to bottom of object.
SPLIT B	
SPLIT --	Position to top of object.
SPLIT T	
SPLIT +	Position one page forwards.
SPLIT +P	
SPLIT -	Position one page backwards.
SPLIT -P	

Command	Function
SPLIT +*nnn*	Position *nnn* lines forwards. This function only applies to the program editor.
SPLIT -*nnn*	Position *nnn* lines backwards. This function only applies to the program editor.
SPLIT . or SPLIT END	Terminate split-screen mode.
SPLIT 7 *function-name* [*library*]	Display function. *function-name* is the name of the function as used in the DEFINE FUNCTION statement (not the object name of the function). This function only applies to the program editor.
SPLIT CLASS *class-name* [*library*]	Display class. *class-name* is the name of the class as used in the DEFINE CLASS statement (not the object name of the class). This function only applies to the program editor.
SPLIT DATA *name* [*library*]	Display data area (global, local, parameter). In the data area editor, you can use asterisk (*) notation for *name* to display a list of the specified object range.
SPLIT DESCRIPTION *program-name* [*library*]	Display the Predict description from the Predict Data Dictionary for the specified *program-name*.
SPLIT FUNCTION *subroutine-name* [*library*]	Display subroutine. *subroutine-name* is the name of the subroutine as used in the DEFINE SUBROUTINE statement (not the object name of the subroutine). This function only applies to the program editor.
SPLIT PROGRAM *name* [*library*]	Display program, subprogram, subroutine, helproutine, copycode, text, map, class, Natural command processor, recording, adapter, dialog, function or resource. In the data area editor, you can use asterisk (*) notation for *name* to display a list of the specified object range.
SPLIT SCAN [*scan-value*]	Scan source code. *scan-value* is the character string to be found in the current source code. Each line containing the *scan-value* is marked with a greater than (>) sign. To further scan for the same *scan-value*, enter S SC only.

Command	Function
SPLIT VIEW *name* [SHORT]	Display view (DDM, as defined in Predict or SYSDDM). If SHORT is specified, the DDM 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. In the data area editor, you can use asterisk (*) notation for *name* to display a list of the specified object range.
SPLIT	Display previous object. If SPLIT is entered without any parameter, the source of the object previously displayed in the split-screen area is displayed again. This applies for the duration of the current editor session. This function only applies to the program editor.

name is the name of the object as contained in the Natural library and/or the Natural system file.

A *library* can be specified with the program editor only. Under Natural Security, a library cannot be specified.

Editor Profile

The Natural program editor and data area editor provide an editor profile, which determines the settings to be in effect during the current program editor or data area session. Editor profile settings are, for example, the standard assignment of PF and PA keys. You can define your own editor profile settings according to your preferences.

This section covers the following topics:

- Invoking the Editor Profile
- Editor Profile Screen
- Additional Options
- Editor Defaults
- General Defaults
- Color Definitions
- Direct Commands

- Exit Profile Maintenance

Invoking the Editor Profile

To invoke your current editor profile

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

```
PROFILE ↩
```

An **Editor Profile** screen similar to the example below appears:

```
11:06:16                 ***** NATURAL EDITORS *****                2010-11-22
                             - Editor Profile -

Profile Name .. SAG_____

PF and PA Keys
  PF1 ... HELP___________   PF2 ... _______________   PF3 ... EXIT___________
  PF4 ... _______________   PF5 ... _______________   PF6 ... _______________
  PF7 ... -______________   PF8 ... +______________   PF9 ... _______________
  PF10 .. SC=____________   PF11 .. _______________   PF12 .. CANCEL________
  PF13 .. _______________   PF14 .. _______________   PF15 .. MENU___________
  PF16 .. _______________   PF17 .. _______________   PF18 .. SCAN___________
  PF19 .. --_____________   PF20 .. ++_____________   PF21 .. _______________
  PF22 .. _______________   PF23 .. _______________   PF24 .. _______________
  PA1 ... _______________   PA2 ... SCAN___________   PA3 ... _______________

Automatic Functions
  Auto Renumber .. Y   Auto Save Numbers .. 0__   Source Save into .. EDITWORK

Additional Options .. N
Command ===>









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  AddOp Save  Reset                          Del   Canc
```

The screen displays your own editor profile if an individual profile exists for your user ID as described for the field **Profile Name**. If such a profile does not exist, the default profile `SYSTEM` is displayed instead. The `SYSTEM` profile is read from the user exit routine USR0070P (see the *Operations* documentation).

To invoke a specific editor profile

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

  ```
  PROFILE profile-name
  ```

  where *profile-name* denotes a valid user ID or SYSTEM.

  When you are in an editor session and enter the PROFILE command together with your own user ID as profile name, your profile is always invoked directly from the database; any modifications made during the current editor session, but not yet saved in the database, will not apply. Therefore, to invoke your current editor profile, enter the PROFILE command only.

Caution: Any changes to the current editor profile are lost when you enter the LOGON system command or open another editor prior to saving the changes to the database.

Editor Profile Screen

This section describes the items contained on the **Editor Profile** screen.

Item	Explanation
Profile Name	The name of the editor profile. Your own editor profile is displayed. If such a profile does not exist, you can modify the default profile to suit your own requirements. To do so, overwrite the profile name SYSTEM with your user ID and save the renamed profile to the database. If you overwrite the name of your profile with any other valid profile name (that is, any other valid user ID) and press ENTER, the profile of the corresponding user is invoked. Only one profile can be established per user ID, and any modifications made to another user's profile are only valid for the current editor session; they cannot be saved to the database. You can, however, overwrite the profile name of another user's profile with your own user ID and then save the renamed profile in the database.
PF and PA Keys	The commands assigned to the PF and PA keys are displayed. Any Natural editor or system command can be assigned. Combinations of commands (separated by a comma) are also possible.
Auto Renumber	Y indicates that the source code in the program editor is to be renumbered automatically if any of the following occurs: ■ A CATALOG, CHECK, RUN, SAVE or STOW command is issued. ■ A .I line command is issued and no line number is available for the line to be inserted. **Note:** See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.
Auto Save Numbers	If a numeric value is entered, a copy of the current source is saved automatically into the source object specified in the **Source Save into** field after the specified number of

Item	Explanation
	modifications have taken place. Modification means each time that the source has been changed as a result of information entered on the screen. **Auto Save Numbers** applies to the map editor, too.
Source Save into	The name of the source object into which a copy of the source is to be saved automatically in the current library. The default name `EDITWORK` can be modified. The specified source object is overwritten each time the number of changes specified in the **Auto Save Numbers** field has been exceeded.
Additional Options	`Y` opens a list of additional options described in the following section.

Additional Options

The **Additional Options** window provides the following:

- **Editor Defaults**
- **General Defaults**
- **Color Definitions**

A plus (+) sign in front of an option indicates that some values have already been set in the corresponding window or with an appropriate editor command.

Enter a `Y` to select the required option.

All options are explained in the following section.

Editor Defaults

Option	Explanation	
Escape Character for Line Command	The escape character which must precede each line command; the default escape character is a period (.).	
Empty Line Suppression	This option applies when inserting lines into the source of an object (except text objects) by using the line command `.I` as described in *Program Editor* and *Data Area Editor*. Possible option settings:	
	`Y`	Any lines left blank are removed from the source as soon as you press ENTER. This is the default setting.
	`N`	Any lines left blank are *not* removed from the source when you press ENTER.

Option	Explanation	
Empty Line Suppression for Text	This option applies when inserting lines into the source of a text object by using the line command `.I` as described in ***Program Editor***. Possible option settings:	
	`Y`	Any lines left blank are removed from the source as soon as you press ENTER.
	`N`	Any lines left blank are *not* removed from the source when you press ENTER. This is the default setting.
Source Size Information	`Y`	The actual size of the object being edited and the remaining space available is displayed in the bottom information line of the editor screen. In addition, in the program editor, the programming mode (reporting or structured) is displayed in the top information line of the editor screen.
	`N`	No such information is displayed.
Source Status Message	`Y`	Program editor: A transaction message is displayed in the top information line each time the source is modified, checked, saved, cataloged or stowed. In addition, an asterisk (*) is displayed in the editor command line if the source contains unsaved modifications. For details, see **Modification Indicator** in the *Program Editor* documentation. Data area editor: An asterisk (*) is displayed in the top information line if the source contains unsaved modifications. For details, see **Modification Indicator** in the *Data Area Editor* documentation.
	`N`	No transaction message and/or asterisk (*) is displayed.
Absolute Mode for SCAN/CHANGE	This option applies when entering the editor command `SCAN` or `CHANGE`.	
	`Y`	Corresponds to the editor command `SET ABS ON`.
	`N`	Corresponds to the editor command `SET ABS OFF`.
	See also *Editor Commands* in the sections ***Program Editor*** and ***Data Area Editor***.	
Range Mode for SCAN/CHANGE	This option applies when entering the editor command `SCAN` or `CHANGE`.	
	`Y`	Corresponds to the editor command `SET RANGE ON`.

Option	Explanation	
	N	Corresponds to the editor command `SET RANGE OFF`.
	See also ***Editor Commands*** in the section *Program Editor*.	
Ignore Case Mode for SCAN/CHANGE	This option applies when entering the editor command `SCAN` or `CHANGE`.	
	This option is relevant only if the option Editing in Lower Case is set to Y, meaning that the scanned value can be entered in upper and lower-case characters.	
	Y	Corresponds to the editor command `SET IGNORECASE ON` which means that any combination of upper and lower-case scan value will be found.
	N	Corresponds to the editor command `SET IGNORECASE OFF` which means that only the exact scan value will be found.
	See also ***Editor Commands*** in the section *Program Editor*.	
Direction Indicator	Indicates the direction (+ or -) in which several editor commands are to work (see also *Editor Command Line* in the sections ***Program Editor*** and ***Data Area Editor***).	
Numeric Input as Line Number	Determines whether numeric input in the first four positions of a source line changes the line number when you press `ENTER`. Input is then not accepted as text.	
	Y	Numeric input is accepted as a source line number. Corresponds to the editor command `SET SEQ OFF`.
	N	Numeric input is accepted as text. Corresponds to the editor command `SET SEQ ON`.
	See also ***Editor Commands*** in the section *Program Editor*.	

General Defaults

Option	Explanation	
Editing in Lower Case	Y	Lower-case characters in the source code are *not* automatically converted to upper case.
	N	Lower-case characters in the source code are automatically converted to upper case. Automatic conversion is in effect by default.
	Caution: Do *not* use the terminal commands %L or %U within the editor.	
Dynamic Conversion of Lower Case	This option is relevant only if the above option is set to Y.	
	Y	All lower-case characters in the source code are automatically converted to upper case except ■ text strings that are enclosed in apostrophes; ■ text strings that are enclosed in quotation marks; ■ comments. These remain as you enter them (see also the section ***Program Editor***).
	N	Any source code remains as you enter it.
Position of Message Line	Indicates the position of the message line; possible values are TOP, BOT, *nn* and -*nn*.	
Cursor Position in Command Line	Y	Indicates that the cursor is positioned in the edit command line after the source has been modified and you pressed ENTER.
	N	The cursor remains in the source area line after the source has been modified and you pressed ENTER.
Stay on Current Screen	Determines whether the current screen stays when you press ENTER.	
	Y	Corresponds to the editor command SET STAY ON.
	N	Corresponds to the editor command SET STAY OFF.
	See *Editor Commands* in the sections ***Program Editor*** and ***Data Area Editor***.	

Option	Explanation	
Prompt Window for Exit Function	Y	When you enter the EXIT command in the editor command line, a confirmation window is displayed (see also *Exit Function* in the sections *Program Editor* and *Data Area Editor*).
ISPF Editor as Program Editor	Y	Natural ISPF (if installed) is invoked instead of the Natural program editor.
Leave Editor with Unlock	Y	Unlocks source code when leaving the editor.
	N	Leaves the editor (default setting).
	C	Unlocks source code and clears the source work area when leaving the editor.
	For more information on locking, see the section *Locking of Source Objects*.	

Color Definitions

In this window, you can specify the colors in which the various elements of the edit-work and split-screen area of your program or data area editor are to be displayed.

For a list of valid colors and color codes, enter a question mark (?) in any input field or press PF1.

Item	Description
Command Line	Editor command line above the edit-work area.
Label Indicator	Leftmost column of the editor screen; used, for example, to label a source code line on which a certain command has been performed (for example, the .X and .Y line commands).
Line Numbers	Column of the source code line numbers (program editor only).
Editor Lines	Lines of source code currently in the editing area and/or split-screen area.
Scan and Error Line	All lines marked with an S (or a greater than (>) sign in split-screen mode) as a result of a scan operation, any line where an error was detected (marked with an E and applicable in the editing area of the program editor only), and the error message line itself.
Information Text	All fields in the information or command line with general information such as the type of object (for example, Program) currently in the editing area.
Information Value	All fields in the information or command line with user-specific information such as the name of the object currently in the editing area.
Information Line	Top and bottom information lines on the editor screen.

Direct Commands

The following direct commands and corresponding PF keys are provided on the **Editor Profile** screen. A direct command is entered in the command line of the screen.

Command	Description
RESET	This command (or PF6) resets all profile parameters to the settings displayed when the **Editor Profile** screen was last opened or when the editor profile was last saved to the database. RESET only applies to the parameter settings of the current editor profile; the editor profile saved in the database is *not* affected by the command.
SAVE	This command (or PF5) saves all current editor profile settings both for the current editor session and in the database. SAVE does *not* terminate the profile maintenance function.
DELETE	This command (or PF11) deletes the editor profile from the database. Before the profile is deleted, however, a confirmation window pops up, in which you can either type the name of the profile and press ENTER to confirm the deletion of the profile. You can only delete your own editor profile.
EXIT	This command (or PF3) terminates the current profile maintenance function, regardless of the **Prompt Window for Exit Function** setting in the **General Defaults** window. If any unsaved profile changes exist, you are prompted to save these changes to the database. See also ***Exit Profile Maintenance***.
CANCEL	This command (or PF12) cancels the current profile maintenance function and returns to the screen from which it was invoked. Any modifications made to the profile have no effect on the current editor session.

Exit Profile Maintenance

After modifying the parameter settings of the current editor profile, you can close a profile maintenance window or leave the **Editor Profile** screen by using one of the following methods:

- Press ENTER.

  The current maintenance window closes. If no window is open, you return to the command prompt of the program editor or data area editor respectively.

  Any modifications are saved for the current editor profile as described for **Exit without Saving** below.

- Or: Press PF3 (Exit) or enter EXIT in the command line of the **Editor Profile** screen (if no maintenance window is open).

  If no modifications were made to the current editor profile, the current maintenance window closes. If no window is open, you return to the command prompt of the program editor or data area editor respectively.

If you have modified the current editor profile, the **EXIT Function** window opens. You can then select one of the following options by either positioning the cursor on the required option or entering any character next to the option:

Function	Explanation
Save and Exit	Returns you to the screen from where the current profile maintenance function was invoked and saves any modifications made to the current editor profile. Modifications are saved both for the current editor session and in the database. If you are working with another user's editor profile, however, modifications made to that profile cannot be saved to the database. They are valid for the current editor session only; a corresponding message is returned.
Exit without Saving	Returns you to the screen from where the current profile maintenance function was invoked. Any modifications made to the current editor profile are only valid for the current editor session; they are *not* saved to the database. **Exit without Saving** corresponds to pressing ENTER.
Resume Function	Closes the prompt window and returns you to the current profile maintenance function.

V Program Editor

6 Program Editor

- Invoking the Program Editor 38
- Editor Command Line 39
- Top Information Line 40
- Editing Area 40
- Bottom Information Line 42
- Editor Commands 42
- Editor Commands for Positioning 50
- Line Commands 51
- Special PF-Key Functions 53
- Cursor-Sensitive Commands 56
- Saving and Cataloging Sources 57
- Exit Function 58

The Natural program editor is used to create and modify the source code of a Natural object of the type program, subprogram, subroutine, helproutine, copycode, text, class or function.

Note: The Natural program editor has been disabled in your environment by default. For more information, see *NaturalONE as the Default Development Environment* in the *Editors* documentation.

Related Topic:

For information on Unicode and code page support for Natural editors, see *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation.

Invoking the Program Editor

To invoke the program editor

- Use the system command `EDIT` as described in the *System Commands* documentation.

  When the program editor is invoked, an editor screen similar to the example below appears:

```
>                                       > +  Program     SAGDEMO  Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 ** EXAMPLE 'SAGDEMO': DISPLAY
  0020 ************************************************************************
  0030 DEFINE DATA LOCAL
  0040 1 VIEWEMP VIEW OF EMPLOYEES
  0050   2 PERSONNEL-ID
  0060   2 NAME
  0070   2 BIRTH
  0080   2 JOB-TITLE
  0090 END-DEFINE
  0100 *
  0110 READ (3) VIEWEMP BY BIRTH
  0120   DISPLAY PERSONNEL-ID NAME JOB-TITLE
  0130 END-READ
  0140 END
  ....
  0280
       ....+....1....+....2....+....3....+....4....+....5....+... S 14   L 1
```

  The editor screen contains the following items (from top to bottom): the **editor command line**, the **top information line**, the **editing area** and the **bottom information line**. These items are explained in the following sections.

Note: If Natural ISPF is installed and the **editor profile** option **ISPF Editor as Program Editor** is set to `Y`, instead of the program editor, either the Natural ISPF main menu (if the `EDIT` command is entered without an object name) or the Natural ISPF editor screen with the specified object is invoked.

Editor Command Line

The editor command line is indicated by the leftmost greater than sign (>) in the top line of the editor screen. In the command line, you can enter one of the following:

- Any Natural system command.

  For example: The system command `CHECK` can be used for checking the syntax of source code and `SAVE` for saving source code (see also ***Saving and Cataloging Sources***).

  For other system commands related to maintaining and using object sources, see *Managing Applications with Natural Objects* in the *System Commands* documentation.
- One or more **editor commands**.
- The name of a Natural program to be executed.

Additionally, the top line can contain the following items (from left to right):

Direction Indicator: + or -	The direction indicator can be set to control the direction of the editor commands `ADD` and `SCAN`, and of the line commands `.C`, `.I` and `.M`. The plus sign (+) indicates *after* and the minus sign (-) indicates *before*. The exact interpretation is described with the relevant command description. See also the editor profile option **Direction Indicator** described in ***Editor Profile***.
`Object Type`	The type of object currently in the source work area. If no object type or object name is specified when the program editor is invoked, object type **Program** is displayed by default. The object type can be changed by using the editor command `SET TYPE`.
Modification Indicator: *	An asterisk (*) indicates whether the source code currently in the source work area contains unsaved modifications. The asterisk (*) also appears for new source code that has not yet been saved as a source object. The asterisk (*) is only visible if the editor profile option **Source Status Message** is set to `Y` (see ***Editor Profile***). The asterisk (*) disappears when you execute a successful `SAVE` or `STOW` command on the source. See also ***Exit Function***.

Object Name	The name of the object currently in the source work area. No name is displayed if the source work area is empty or if the current source code has not yet been saved as a source object with the SAVE, CATALOG or STOW command.
Lib	The library to which you are currently logged on.

Top Information Line

The top information line of the editor screen is a scale line. It can contain the following:

- A message indicating object modification. This information is only displayed if the editor profile option **Source Status Message** is set to Y (see ***Editor Profile***).
- The programming mode (structured or reporting) currently in effect. When a Natural object is read into the source work area, the mode is set to the one which was in effect when the object was saved with the SAVE or STOW command. This information is only displayed if the editor profile option **Source Size Information** is set to Y (see ***Editor Profile***).

  For information on the differences between structured and reporting mode, see *Purpose of Programming Modes* in the *Programming Guide*.

Editing Area

The editing area of the editor screen contains the numbered lines where you add or modify source code.

The editing area is either empty or contains source code that was last read into the source work area with the command EDIT or READ as shown in the example of a program in ***Invoking the Program Editor***.

When you read in the source of an existing object, the entire source code is loaded into the source work area and is available for editing. However, depending on the size of the source, the editing area may not show all of the lines that belong to the source. In this case, you have to scroll down the source (see ***Editor Commands for Positioning***) to go to the line you want to view or modify.

In addition, if you use split-screen mode, the editing area displays fewer lines of source code. See also ***Split-Screen Mode***.

To create or edit source code, you can perform multiple functions:

- Type in or update code directly in the relevant source line.
- Use one or more **editor commands** as described in the relevant section.
- Use one or more **line commands** as described in the relevant section.

When performing multiple functions, consider the following:

- Only one insert line command (`.I`) can be performed at a time.
- You can enter multiple commands in the command line of the editor: you can enter more than one editor command, but only the last command entered in the editor command line can be a system command. For example:

```
SC 'MOVE',-2,RENUMBER
```

> **Note:** Natural treats the editor command `N` like a system command. `N` corresponds to the system command `RENUMBER`.

- If you have changed the source code by typing in a modification or by using an editor command, a system command cannot be entered until you press ENTER.

Dynamic Conversion from Lower to Upper Case

You can activate or deactivate dynamic conversion to upper case, by setting the appropriate **editor profile** options **Editing in Lower Case** and **Dynamic Conversion of Lower Case** to `Y` (see ***Editor Profile***). All source code you enter in the editing area is then converted to upper case, with the following exceptions:

- The contents of a Natural object of the type text remain as entered.
- A text string that is not
  - a hexadecimal constant or
  - a Unicode hexadecimal constant

  and is enclosed in apostrophes or quotation marks remains as you enter it.
- DBCS characters enclosed in shift-out and shift-in characters remain as you enter them.
- A comment indicated by the character string blank-slash-asterisk (/*) remains as you enter it.

> **Caution:** If the character string slash-asterisk (/*) denotes an executable part of a statement, it must be specified *without* a blank character in front of the string (/*). The string will otherwise be considered a comment.

Bottom Information Line

The bottom information line of the editor screen is a scale line. It can contain the following:

- **Current Source Size**

  The size (number of characters) of the current source. As source lines are stored in variable length in the source work area, trailing blanks within a source line are not counted; leading and embedded blanks are counted. This information is only displayed if the editor profile option **Source Size Information** is set to `Y` (see *Editor Profile*).

- **Char. Free**

  The number of characters still available in the source work area. This information is only displayed if the editor profile option **Source Size Information** is set to `Y` (see *Editor Profile*).

- **S**

  The size (number of lines) of the source being edited.

- **L**

  The number of the source line currently displayed as the top line.

Editor Commands

Editor commands are entered in the command line of the program editor. The command parameters must be separated either by the input delimiter character as defined with the Natural session parameter ID (the default delimiter character is comma (,)) or by a blank. When multiple commands are entered, these must also be separated by the delimiter character or by blanks.

The editor commands available are described in the following table and in the section *Editor Commands for Positioning*. For explanations of the syntax symbols used in the editor commands, refer to *System Command Syntax* in the *System Commands* documentation. An underlined portion of a command denotes a valid abbreviation.

Editor Command	Function
ADD[(*n*)]	Adds *n* blank lines. If the **direction indicator** is set to + (plus sign), the lines are added after the last line of the object being edited; if the direction indicator is set to - (minus sign), the lines are added before the first line of the object. The value for *n* can be in the range from 1 to 9. If *n* is not (or not correctly) specified, 9 lines (4 in split-screen mode) are added by default.

Editor Command	Function
	With the next ENTER, lines that are still left blank are removed.
CANCEL or . (a period)	Leaves the editor. Any modifications made since the last time the SAVE command was entered are *not* saved.
CATALOG [*object-name*]	Executes the system command CATALOG which checks and catalogs the current source code. You must supply an object name with the command if you catalog new source code or if you want to copy the current source code. See also ***Saving and Cataloging Sources***.
CHANGE ['*scan-value*'*replace-value*']	Scans the source code for the character string entered as *scan-value* and replaces each such *scan-value* found with the character string entered as *replace-value*. Each line in which a character string is replaced is marked with an R to the left of the line. Any special character which is not valid within a Natural variable name can be used as the delimiter character. If you enter CHANGE without any parameter, the **SCAN/REPLACE** window is invoked. In addition to the *scan-value* and the *replace-value*, you can specify the following in this window: ■ the null value option (see SET NUL), ■ the absolute scan mode (see SET ABS), and ■ the range mode (see SET RANGE).
CHECK	Executes the system command CHECK which checks the syntax of the current source code. If an error is found, the erroneous line is marked with an E and an appropriate error message appears in the message line. If no errors are found, a message appears indicating successful completion of the check.
CLEAR	Executes the system command CLEAR which clears the source work area (including the object name and the line markers X and Y).
DX or DY	Deletes the X-marked or the Y-marked line. See also the line commands .X and .Y.
DX-Y	Deletes the block of lines delimited by the X and Y markers. See also the line commands .X and .Y.

Editor Command	Function	
EX or EY	Deletes lines from the top of the editing area to, but not including, the X-marked line; or from the line following the Y-marked line to the bottom of the editing area. See also the line commands .X and .Y.	
EX-Y	Deletes all lines in the editing area excluding the block delimited by X and Y. See also the line commands .X and .Y.	
EXIT	Leaves the editor. Any modifications to the source are saved depending on the setting of the **editor profile** described in ***Exit Function***.	
LET	Undoes all modifications made to the current screen since the last time ENTER was pressed. In addition, LET ignores all line commands already entered but not yet executed.	
N [(*nnnn*)]	This command corresponds to the system command RENUMBER. It renumbers the lines of the source code currently in the source work area. If you only enter N, the lines are numbered in increments of 10; if you enter N (*nnnn*), the lines are renumbered in increments of *nnnn*. If the value specified for *n* is too big, lines are numbered in increments of 5. **Note:** See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.	
PROFILE [*name*]	Invokes the **Editor Profile** screen where you can view or change your current editor profile settings. For details, see the section ***Editor Profile***.	
REN ON|OFF	ON	Renumbers the lines of a source whenever the CHECK, RUN, SAVE, CATALOG or STOW command is executed on the source.
	OFF	Indicates that automatic renumbering is not in effect.
	The default is ON. The REN command corresponds to the editor profile option **Auto Renumber** described in ***Editor Profile***. **Note:** See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.	
RESET	Deletes the current X and Y line markers and any marker previously set with the line command .N. See also the line commands .X and .Y.	

Editor Command	Function
SAVE [*object-name*]	Executes the system command SAVE which saves the current source code. You must supply an object name if you save new source code or if you want to copy the current source code. See also ***Saving and Cataloging Sources***.
SCAN ['*scan-value*']	Scans the source code for a character string (*scan-value*). Each line in which the *scan-value* is found is marked with an S to the left of the line. If the supplied *scan-value* is entered without delimiter characters, for example, SCAN ABC D, the entire character string which follows the keyword SCAN is used as the scan value. **Note:** The SCAN command performs an exact search for the supplied *scan-value*. This should be taken into account when searching for DBCS (Double Byte Character Set) characters. If the direction indicator is set to + (plus sign), the scan is performed from the first line shown on the screen to the last line of the source work area. If the direction indicator is set to - (minus sign), the scan is performed from the last line shown on the screen to the first line of the source work area. If you enter SCAN without any parameter, the **SCAN/REPLACE** window is invoked. In addition to the *scan-value*, you can specify the following in this window: ■ a *replace-value* (see CHANGE), ■ the null value option (see SET NUL), ■ the absolute scan mode (see SET ABS), ■ the range mode (see SET RANGE), and ■ the ignore case mode (see SET IGNORECASE). SCAN is a cursor-sensitive command which provides additional options described in ***Cursor-Sensitive Commands***.
SCAN=[+|-]	Scans for the next occurrence of *scan-value* specified with the SCAN command. The direction for a given scan command can be explicitly specified by entering SCAN=+ or SCAN=-. The setting of the direction indicator is then ignored. **Note:** The equal sign (=) used with the SCAN command is the default input assign character. If another character has been specified as the

Editor Command	Function	
	input assign character (see session parameter `IA` as described in the *Parameter Reference*), that other character must be used instead.	
`SET ABS [ON|OFF]`	`ON`	The `SCAN` and `CHANGE` commands operate in absolute mode, which means that the *`scan-value`* and the *`replace-value`* need not be delimited by blanks or special characters.
	`OFF`	The `SCAN` and `CHANGE` commands do not operate in absolute mode, which means that the *`scan-value`* and the *`replace-value`* must be delimited by blanks or special characters.
	The default is `OFF`. The `SET ABS` command corresponds to the editor profile option **Absolute Mode for SCAN/CHANGE** described in ***Editor Profile***.	
`SET CAPS [ON|OFF|ALL]`	This command affects the settings of the editor profile options **Editing in Lower Case** and **Dynamic Conversion of Lower Case** described in *Editor Profile*.	
	`ON`	Converts all lowercase characters displayed on the current screen to upper case, except for those characters placed in a comment or enclosed in apostrophes or quotation marks. Sets **Editing in Lower Case** to `Y` and **Dynamic Conversion of Lower Case** to `Y`.
	`OFF`	Characters are not converted, they remain as entered. Sets **Editing in Lower Case** to `Y` and **Dynamic Conversion of Lower Case** to `N`.
	`ALL`	Converts all characters to uppercase. Sets **Editing in Lower Case** to `N` and **Dynamic Conversion of Lower Case** to `N`.
	The default is `ON`.	
`SET ESCAPE` *`character`*	The escape character which must precede each line command. The **default escape character** is the period (.).	
`SET IGNORECASE [ON|OFF]`	`ON`	The `SCAN` and `CHANGE` commands operate in ignore case mode which means that any combination of upper

Editor Command	Function	
		and lower-case scan value will be found.
	OFF	The SCAN and CHANGE commands do not operate in ignore case mode which means that only the exact scan value will be found.
	The default is OFF. The SET IGNORECASE command corresponds to the editor profile option **Ignore Case Mode for SCAN/CHANGE** described in ***Editor Profile***.	
SET NUL [ON|OFF]	ON	All occurrences of a value scanned with the SCAN command are deleted. After the deletion of the scanned value, the SET NUL command is automatically set to OFF.
	The default is OFF.	
SET RANGE [ON|OFF]	ON	The SCAN and CHANGE commands operate in range mode, which means that the value to be scanned/changed must be located within the range of lines delimited by the X and Y line markers.
	OFF	The SCAN and CHANGE commands operate in non-range mode, which means that no range limit is to be in effect.
	The default is OFF. The SET RANGE command corresponds to the editor profile option **Range Mode for SCAN/CHANGE** described in ***Editor Profile***.	
SET RELINE [ON|OFF]	ON	Removes empty lines from the source of an object (except text objects) when you press ENTER.
	OFF	Empty lines are retained.
	The default is ON. This command corresponds to the **Empty Line Suppression** option described in ***Editor Profile***.	
SET SEQ [ON|OFF]	OFF	If your input is numeric, the first four positions in the editing area are considered as the line number and are moved to the line number position once you press ENTER.

Editor Command	Function	
		This feature is useful, for example, if a statement line is to be referenced by a source code line number in another statement line; when you renumber the source code, the referencing line number is renumbered, too.
	ON	Numeric input in the first four positions remains as entered.
	The default is OFF except for object type text. The SET SEQ command corresponds to the editor profile option **Numeric Input as Line Number** described in ***Editor Profile***.	
SET SIZE [ON|OFF]	ON	The size of the source is displayed at the bottom information line of the editor screen and the programming mode is displayed on the scale line.
	OFF	This information is not displayed.
	The default is OFF. The SET SIZE command corresponds to the editor profile option **Source Size Information** described in ***Editor Profile***.	
SET STAY [ON|OFF]	ON	The current screen will stay when ENTER is pressed. Forward and backward positioning can be done by positioning commands only.
	OFF	Pressing ENTER positions to the next screen.
	The default is OFF. The SET STAY command corresponds to the editor profile option **Stay on Current Screen** described in ***Editor Profile***.	
SET TYPE *object-type*	This command is used to change the object type (displayed in the editor command line) for the source currently contained in the source work area. *object-type* represents one of the following values that must be specified with the command:	
	P	or PROGRAM
	S	or SUBROUTINE
	N	or SUBPROGRAM
	H	or HELPROUTINE
	T	or TEXT
	C	or COPYCODE

Editor Command	Function	
	4	or CL or CLASS
	7	or FUNCTION
SHIFT [-*nn*|+*nn*]	This command shifts each line delimited by the X and Y markers to the left or right. The *nn* parameter represents the number of characters the line is to be shifted. Comment lines are not shifted.	
SHIFT --	This command shifts each line delimited by the X and Y markers to the leftmost position. Comment lines are not shifted.	
SHIFT ++	This command shifts each line delimited by the X and Y markers to the rightmost position (maximum 99 positions). Comment lines are not shifted.	
SPLIT [*parameter*]	This command splits the editor screen and displays the source of another Natural object in one half of the screen as described in ***Split-Screen Mode***. *parameter* represents a parameter that must be specified with the command as described in ***Split-Screen Commands*** SPLIT is a cursor-sensitive command which provides additional options described in ***Cursor-Sensitive Commands***. If SPLIT is entered without any parameter, the source of the object previously displayed in the split-screen area is displayed again. This applies for the duration of the current editor session.	
STOW [*object-name*]	Executes the system command STOW which saves and catalogs the current source code. You must supply an object name if you STOW new source code or if you want to copy the current source code. Otherwise, an appropriate message appears. See also ***Saving and Cataloging Sources***.	
STRUCT [DISPLAY]	This command performs structural indentation of source code. The default indentation is in increments of two positions. If DISPLAY is specified, the source code is displayed in compressed form: see the system command STRUCT in the *System Commands* documentation.	
*	This command displays the editor command most recently entered.	
*=	This command again executes the command most recently entered in the command line.	

Editor Commands for Positioning

Editor commands for positioning are entered in the command line of the program editor. The following commands are available for positioning:

Command	Function
ENTER or +P or +	Position forwards one page.
-P or -	Position backwards one page.
+H	Position forwards half a page.
-H	Position backwards half a page.
T or -	Position to top of source.
B or ++	Position to bottom of source.
+*nnnn*	Position forwards *nnnn* lines (maximum 4 digits).
-*nnnn*	Position backwards *nnnn* lines (maximum 4 digits).
nnnn	Position to line number *nnnn*.
X	Position to the line marked with an X.
Y	Position to the line marked with a Y.
POINT	Positions to the line in which the line command .N was entered. See also the line command .P.

Line Commands

Line commands are entered in the first position of a source line. The line commands provided by the program editor are listed below. The notation (*nn*) or (*nnnn*) indicates a repetition factor. The default repetition value is 1 (with the exception of the `.I` command; see below). For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Note: We recommend that you enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

Line Command	Function
`.C[(nnnn)]`	Copies the line in which the command was entered.

`.CX[(nnnn)]`
or
`.CY[(nnnn)]`
	Copies the X-marked or the Y-marked line. See also the line commands `.X` and `.Y` as well as ***Notes for Line Commands***.

`.CX-Y[(nnnn)]`	Copies the block of lines delimited by the `X` and `Y` markers. (See also ***Notes for Line Commands***.)
`.D[(nnnn)]`	Deletes one or more lines beginning with the line in which you enter the command towards the end of the source (regardless of any direction indicator setting). The default is 1 line.
`.I[(nn)]`	Inserts *nn* empty lines, where *nn* can range from 1 to the total number of lines shown in the editing area minus two. For example, if a total of 28 lines is shown in the editing area, you can insert a maximum of 26 lines. If *nn* is not (or not correctly) specified, 9 lines (4 lines in split-screen mode) are inserted by default. Lines that are left blank are then removed from the source, depending on the setting of the editor profile options **Empty Line Suppression** and **Empty Line Suppression for Text** described in ***Editor Profile***. See also ***Notes for Line Commands***.
`.I(obj,ssss,nnnn)`	Includes into the source an object contained in the current library or in the steplib (the default steplib is SYSTEM). Depending on the **direction indicator**, the object is inserted before or after the line in which you enter the command. If you want to include only part of the object, you specify as *ssss* the first line to be included (for example, `20` means the inclusion will start from the 20th line), and as *nnnn* the number of lines to be included. If you enter multiple commands, this command is always executed after all other line and/or editor commands have been executed.

Line Command	Function
	If the object is a map, an `INPUT USING MAP` statement (see *INPUT Syntax 2 - Using Predefined Map Layout* in the *Statements* documentation) with all defined variables is automatically included in the current line. If the object is a data area, the entire data area is included, except comment lines. Only local and parameter data areas that were saved and cataloged with the `STOW` command can be included into the source work area; global data areas cannot be included. If the object is an adapter, a `PROCESS PAGE USING` (see *Syntax 2 - PROCESS PAGE USING* in the *Statements* documentation) with all defined variables is automatically included in the current line.
`.J`	Joins the current line with the next line. If the resulting line exceeds the length of the editor screen line, the line is marked with an `L` and must be split in two with the `.S` command (see below) before it can be modified.
`.L`	Undoes all modifications that have been made to the line since the last time ENTER was pressed.

`.MX`
 	
or
 	
`.MY`
	Moves the X-marked or the Y-marked line. See also the line commands `.X` and `.Y` as well as ***Notes for Line Commands***.

`.MX-Y`	Moves the block of lines delimited by the `X` and `Y` markers (see also ***Notes for Line Commands***).
`.N`	Marks (invisibly) a line to be positioned at the beginning of the source work area by the editor command `POINT`. The mark is automatically deleted when an error with a line command or editor command occurs, or when the `RESET` command is executed.
`.P`	Positions the line marked with this command to the top of the screen.
`.S`	Splits the line at the position marked with the cursor.
`.X`	Marks a line with an `X` (see also ***Notes for Line Commands***).
`.Y`	Marks a line with a `Y` (see also ***Notes for Line Commands***).

Notes for Line Commands:

- If both the commands `.X` and `.Y` are applied to one line, it is treated as being marked with an `X` and with a `Y`; the line marker actually shown to reflect this status is a `Z`.
- If the **direction indicator** is set to + (plus sign), the copied, inserted or moved lines are placed *after* the line in which the corresponding command was entered; if the direction indicator is set to - (minus sign), the copied, inserted or moved lines are placed *before* the line in which the command was entered.

Special PF-Key Functions

The following special functions can also be controlled using PF keys:

Function	Explanation
`*CURSOR`	A line split function can be combined with the command `.I`, `.CX`, `.CX-Y`, `.MX` or `.MX-Y`. This is accomplished by assigning the value `*CURSOR` to a PF key in the editor profile (see **PF and PA Keys** in ***Editor Profile***). If this PF key is then pressed instead of ENTER after a line command has been entered, the line in which the command was entered is first split at the cursor position and then the line command is executed. See also ***Example of *CURSOR on PF Key***.

`*X`
or
`*Y`
	If a PF key is assigned the value `*X` or `*Y` in the editor profile (see **PF and PA Keys** in ***Editor Profile***), the cursor position is marked `X` or `Y` whenever this PF key is used. These position markers are then used to determine which portion of a line is to be included in the command operation. See also ***Example of *X and *Y on PF Keys***.

Example of *CURSOR on PF Key:

The following are instructions for using a PF key to which the value `*CURSOR` is assigned.

1. In the **PF and PA Keys** section of the **Editor Profile** screen (see ***Editor Profile***), enter `*CURSOR` next to PF6.
2. Open the program editor, type in the following text, and press ENTER:

```
>                                       > +  Program              Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 MOVE A TO B
  0020 WRITE A B C
  0030 MOVE C TO B MOVE A TO C
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0280
       ....+....1....+....2....+....3....+....4....+....5....+... S 3    L 1
```

3. In line `0020`, enter the line command `.X` and press ENTER.

   The line is marked as indicated by the `X` next to it.
4. In line `0030`, enter the line command `.CX`, place the cursor on the `M` of the second `MOVE`, and press PF6.

The screen then looks similar to the one below:

```
>                                       > +  Program              Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 MOVE A TO B
X 0020 WRITE A B C
  0030 MOVE C TO B
  0020 WRITE A B C
  0030 MOVE A TO C
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0260
       ....+....1....+....2....+....3....+....4....+....5....+... S 5    L 1
```

Line `0030` is split before the cursor position, line `0020` is copied to the line after the line in which you entered the `.CX` command, and the second half of the split line is moved to the last line.

Example of *X and *Y on PF Keys

The following are instructions for using PF keys to which the values `*X` and `*Y` are assigned.

1. In the **PF and PA Keys** section of the **Editor Profile** screen (see *Editor Profile*), enter `*X` next to PF4 and `*Y` next to PF5.
2. Open the program editor, type in the following text, and press ENTER:

```
>                                       > +  Text                 Lib SAGTEST
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 THIS PORTION
  0020 OF TEXT IS
  0030 FOR DEMONSTRATION OF
  0040 PF-KEY ASSIGNMENTS.
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0280
       ....+....1....+....2....+....3....+....4....+....5....+... S 0    L 1
```

3. In line `0010`, place the cursor on the letter `P` and press PF4.

The position of P is marked as shown on the following example screen.

4. In line 0030, place the cursor on the blank character behind DEMONSTRATION and press PF5.

   The screen then looks similar to the one below:

```
>                                       > +  Text                 Lib SAGTEST
All    ....+X...1....+..Y.2....+....3....+....4....+....5....+....6....+....7..
X 0010 THIS PORTION
  0020 OF TEXT IS
Y 0030 FOR DEMONSTRATION OF
  0040 PF-KEY ASSIGNMENTS.
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0280
       ....+X...1....+..Y.2....+....3....+....4....+....5....+... S 4    L 1
```

   The positions of the characters (P and blank) are marked as indicated by the X and Y respectively, which appear in the top and bottom information lines and next to the source lines that contain the marked characters.

5. In line 0040, enter the line command .MX-Y and press ENTER.

   The screen then looks similar to the one below:

```
>                                       > +  Text                 Lib SAGTEST
All    X...+....1....+.Y..2....+....3....+....4....+....5....+....6....+....7..
  0010 THIS
  0030 OF
  0040 PF-KEY ASSIGNMENTS.
X 0010 PORTION
  0020 OF TEXT IS
Y 0030 FOR DEMONSTRATION
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  ....
  0250
       X...+....1....+.Y..2....+....3....+....4....+....5....+... S 6    L 1
```

The block of text starting with `P` in line `0010` and ending with `N` in line `0030` is moved to the line below the line in which you entered the command. The moved block of text and the remaining text in line `0010` and `0030` are left-justified.

Cursor-Sensitive Commands

Cursor-sensitive commands are entered without any parameters but with the cursor positioned anywhere on the editor screen (except in the command line).

To use a cursor-sensitive command

1 Enter a cursor-sensitive command in the editor command line.

2 Position the cursor on any character string contained on the editor screen (except in the command line).

3 Press ENTER.

Alternatively, you can use a PF key:

1. Position the cursor on any character string contained on the editor screen (except in the command line).
2. Press the PF key to which the required cursor-sensitive command is assigned (see **PF and PA Keys** in ***Editor Profile***).

The following topics are covered below:

- SCAN Command
- SPLIT Command
- EDIT and LIST System Commands

SCAN Command

The cursor-sensitive `SCAN` command searches the current source for the character string on which the cursor is positioned. (If the cursor is positioned on a blank character, however, the **SCAN/REPLACE** window is invoked.)

In split-screen mode, the cursor can be placed on a string in the split-screen area, too.

When using the `SPLIT SCAN` command, the same applies as for the `SCAN` command, but the scan operation is performed in the split-screen area only (see also the section ***Split-Screen Commands***).

For more information on `SCAN`, see the relevant description in ***Editor Commands***.

SPLIT Command

Instead of using object-specific `SPLIT` commands to display the source of an object (see the section *Split-Screen Commands*), you can use the cursor-sensitive `SPLIT` command.

If you use the cursor-sensitive `SPLIT` command and position the cursor on a character string that corresponds to an object name or extended name (long name), the source code of the respective object is displayed in the split-screen area of the editor. The required object must be contained in the current library.

This feature is particularly useful, for example, if you do not know the object name that corresponds to the extended name (long name) of an object (for example, a subroutine) referenced in the current source.

EDIT and LIST System Commands

If you use the cursor-sensitive `EDIT` or `LIST` command and position the cursor on a character string that corresponds to an object name or extended name (long name), the source of the respective object is shown:

- With the `EDIT` command, the source is loaded into the editing area. If necessary, even a different editor is invoked.
- With the `LIST` command, the source is listed, even if a view (DDM) has been referenced.

The objects to be edited or listed must be contained in the current library.

For more information on `EDIT` and `LIST`, see the *System Commands* documentation.

Saving and Cataloging Sources

You can save the source code currently in the source work area as a source object and also as a cataloged object, which are stored in a Natural library in a Natural system file.

To save and/or catalog the current source

- Use the system command `SAVE`, `CATALOG` or `STOW` as described in *Saving and Cataloging Objects* in the *Using Natural* documentation.

  Note: When you leave the program editor with the `EXIT` editor command, the current source code is saved automatically if the appropriate **editor profile** option is set accordingly as described in *Exit Function*.

To keep a copy of the current source

- Use the editor options **Source Save into** and **Auto Save Numbers** as described in *Editor Profile*.

  A copy of the source edited last with any of the Natural editors is then automatically saved as a source object in the current Natural environment.

Exit Function

The exit function is used to terminate the current editor session by issuing the `EXIT` command.

The effect of the `EXIT` command depends on the setting of the **editor profile** option **Prompt Window for Exit Function:**

- If the option is set to `Y` (default setting), the `EXIT` command invokes the **EXIT Function** window whenever you execute the command on a source that contains unsaved modifications (see also **Modification Indicator**). If no modifications were made to the source, the window does *not* appear and the editor closes without saving the source.

  The **EXIT Function** window provides the following options:

Option	Explanation
Save and Exit	Saves all modifications made to the current source code and leaves the editor. See also *SAVE Object Window*.
Exit without Saving	Leaves the editor without saving any modifications made to the current source code since it was last saved.
Resume Function	Neither leaves the editor nor saves any modifications; the prompt window is closed and the current function is resumed.

- If the option is set to `N`, the `EXIT` command executes the **Save and Exit** function: saves all modifications made to the current source and leaves the editor. See also *SAVE Object Window*.

SAVE Object Window

The **Save and Exit** function invokes the **SAVE Object** window if no object name has yet been defined for the current source code (as indicated in the editor command line).

This window prompts you to enter an object name and type. If you confirm your entries with ENTER, the source is saved under the specified name as a new object in the current library and the editor is terminated. PF3 closes the window without any action and the editor session is retained.

VI Data Area Editor

7 Data Area Editor

- Invoking the Data Area Editor 64
- Top Information Line 65
- Editor Command Line 66
- Bottom Information Line 67
- Using the Editing Area 67
- Columns in the Editing Area 68
- Extended Field Definition Editing 72
- Line Commands 77
- Editor Commands 83
- Editor Commands for Positioning 87
- Storing and Cataloging a Data Area 88
- User Exit for the Data Area Editor 89
- Exit Function 90

The data area editor is used to create and modify a data area. A data area is a Natural object of the type global data area (GDA), local data area (LDA) or parameter data area (PDA). For information on using a data area, see *Data Areas* in the *Programming Guide*.

A data area contains data element definitions, such as user-defined variables, constants and database fields referenced with a data view in a data definition module (DDM), which can be used by one or more Natural objects. You can also create copycode from a data area. Note that data views from a DDM cannot be defined in PDAs. However, you can convert a data view in a GDA or an LDA into a group structure and then save this GDA or LDA as a PDA.

Note: The data area editor has been disabled in your environment by default. For more information, see ***NaturalONE as the Default Development Environment*** in the *Editors* documentation.

Related Topic:

For information on Unicode and code page support for Natural editors, see *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation.

Invoking the Data Area Editor

You invoke the data area editor with the system command `EDIT` described in the *System Commands* documentation.

To invoke the data area editor for a new data area

- Issue the `EDIT` command specifying the type of data area (`GLOBAL`, `LOCAL` or `PARAMETER`) you want to create.

  For example:

  ```
  EDIT LOCAL
  ```

  An editor screen with an empty editing area appears for a local data area (indicated in the top left corner of the screen) similar to the example shown in the following instructions.

To invoke the data area editor for an existing data area

- Issue the command `EDIT` specifying the name of a data area that has been stored as a source object in your current Natural environment.

  For example:

  ```
  EDIT LDA1
  ```

An editor screen similar to the example below appears which contains the source of the local data area `LDA1`:

```
Local     LDA1     Library SAGTEST                          DBID  10 FNR  32
  Command                                                                  > +
  I T L  Name                            F Length     Miscellaneous
  All -- ------------------------------- - ---------- ------------------------->
    *    LDA for new application
       1 INCOME                          A         20 (1:3,1:5) INIT ALL<'0'>
       1 PERSON
       2 SEX                             A          6
       2 AGE                             N          3
       1 NAME                            A         24
    R  1 NAME                                        /* REDEF. BEGIN : NAME
       2 FIRST-NAME                      A         10
       2 MIDDLE-INIT                     A          2
       2 LAST-NAME                       A         10
    C  1 DOLLAR                          A          5 CONST<'$US'>
    V  1 FINANCE-VIEW                                FINANCE
       2 PERSONNEL-NUMBER                N        8.0
    P  2 MAJOR-CREDIT                                (1:1) /* PERIODIC GROUP
       3 CREDIT-CARD                     A         18 (EM=XXX.XXX.XXX.XXX.XXX.XXX)
       3 CREDIT-LIMIT                    N        4.0
       3 CURRENT-BALANCE                 N        4.0
  -- Current Source Size: 1969  Free: 78200 ----------------------- S 12   L 1
```

The editor screen contains the following items (from top to bottom): the **top information line**, the **editor command line**, the **editing area** and the **bottom information line**. These items are explained in the following sections.

Top Information Line

The top information line of the editor screen can contain the following items (from left to right):

Data Area Type	Indicates the type of data area currently in the source work area: `Local`, `Global` or `Parameter`. The type can be changed by using the editor command `SET TYPE`.
Modification Indicator: *	An asterisk (*) indicates whether the source code currently in the source work area contains unsaved modifications. The asterisk (*) also appears for new source code that has not yet been saved as a source object. The asterisk (*) is only visible if the editor profile option **Source Status Message** is set to `Y` (see ***Editor Profile***). The asterisk (*) disappears when you execute a successful `SAVE` or `STOW` command on the source.

	See also ***Exit Function***.
`Data Area Name`	The name of the data area currently in the source work area. No name is displayed if the source work area is empty or if the current source code has not yet been saved as a source object with the `SAVE`, `CATALOG` or `STOW` command.
Lib	The library where you are currently logged on.
DBID	The database ID of the current system file.
FNR	The file number of the current system file.

Editor Command Line

The command line is indicated by the editor's **Command** prompt. In the command line, you can enter one of the following:

- Any Natural system command.

  For example: The system command `CHECK` can be used for checking the syntax of source code and `SAVE` for saving source code (see also ***Storing and Cataloging a Data Area***).

  For other system commands related to maintaining and using object sources, see *Managing Applications with Natural Objects* in the *System Commands* documentation.
- The name of a Natural program to be executed.
- One or more **editor commands**.

Note: If you have changed a definition by typing in a modification or by using an editor command, a system command cannot be entered until you press ENTER.

Direction Indicator

The direction indicator entered next to the > (greater than) sign in the command line determines the operation direction of particular editor and line commands:

- **+**

  (plus sign)

  The command executes from the top line displayed on the screen (or from the line in which a line command is entered) towards the end of the source. This is the default setting.
- **-**

  (minus sign)

  The command executes from the top line displayed on the screen (or from the line in which a line command is entered) towards the beginning of the source.

More detailed information on the direction indicator can be found in the descriptions of the editor and line commands affected by the operation direction.

See also the editor profile option **Direction Indicator** described in *Editor Profile*.

Bottom Information Line

The bottom information line of the editor screen can contain the following:

- **Current Source Size**

  The size (number of characters) of the current source. This information is only displayed if the editor profile option **Source Size Information** is set to `Y` (see *Editor Profile*).

- **Free**

  The number of characters still available in the source work area. This information is only displayed if the editor profile option **Source Size Information** is set to `Y` (see *Editor Profile*).

- **S**

  The size (number of lines) of the source being edited.

- **L**

  The number of the source line currently displayed as the top line.

Using the Editing Area

The editing area is either empty or contains source code that was last read into the source work area with the command `EDIT` or `READ` as shown in the example in *Invoking the Data Area Editor*.

When you read in the source of an existing object, the entire source code is loaded into the source work area and is available for editing. However, depending on the size of the source, the editing area may not show all of the lines that belong to the source. In this case, you have to scroll down in the source to go to the line you want to view or modify.

In addition, if you use split-screen mode, the editing area displays fewer lines of source code. See also *Split-Screen Mode*.

To navigate in the editing area

- Use the editor commands described for the program editor in *Editor Commands for Positioning*.

All positioning commands described for the program editor can be used with the data area editor as well.

To create or modify variables or fields

- Type in or modify all variable or field definitions in the columns of the relevant source line.

  You can specify whether the characters you type are automatically converted to upper case by using the editor profile options **Editing in Lower Case** and **Dynamic Conversion of Lower Case** (see *Editor Profile*).

  Or:

  Use one or more **line commands** as described in the relevant section.

  A line command, for example, is used to insert a line, copy variable or field definitions from another Natural object, or invoke the **extended field definition editing** function.

  Or:

  Use one or more **editor commands** as described in the relevant section.

  An editor command, for example, is used to delete a block of lines or specify prefixes for names.

Columns in the Editing Area

The editing area of the editor screen is organized in columns where all attribute definitions that belong to a variable or field are maintained in one line.

The editing area contains the following columns:

Column Heading	Explanation	
I	The label indicator. An information field supplied by the editor. This column is not modifiable. Possible column entries are:	
	+	Indicates that more than one of the entries listed below exist for the variable or field.
	E	Indicates that a definition error has been detected.

Column Heading	Explanation	
	A	Indicates that array bounds have been defined by using the `.E` line command.
	I	Indicates that an initial value has been defined by using the `.E` line command.
	M	Indicates that an edit mask and/or a header has been defined by using the `.E` line command.
	S	Indicates that both an initial value and an edit mask have been defined by using the `.E` line command.
	The following only applies to PDAs:	
	blank	Indicates the parameter specification call-by-reference (default).
	V	Indicates the parameter specification call-by-value.
	R	Indicates the parameter specification call-by-value-result.
	O	Indicates an optional parameter that *can* be passed.
	For details, see **function code P** in *Extended Field Definition Editing*.	
T	The type of variable or field. Possible types are:	
	B	A data block within a GDA.
	C	A user-defined constant (not applicable to PDAs) or a counter field (`C*` variable). A counter field is used to retrieve the number of occurrences of a multiple-value field or a period group in a view (DDM). See also *CONSTANT* in the *Statements* documentation and *Referencing the Internal Count for a Database Array (C* Notation)* in the *Programming Guide*.
	G	A group within a view (DDM).
	M	A multiple-value field within a view (DDM).

Column Heading	Explanation
	`O` — The handle of an object. `P` — A periodic group within a view (DDM). `R` — The redefinition of a variable or field. `U` — The Globally Unique Identifier (GUID). `V` — Not applicable to PDAs. A view definition created from a DDM. *blank* — A user-defined variable or field, or a group structure (not within a view). `*` — A comment field.
L	The level number of the variable or field (1 - 99). Variables which are not within a hierarchical structure and view definitions must be assigned level 1. Level numbers cannot be used with data block definitions.
Name	The name of the variable or field, block or view. For valid names, see *Naming Conventions for User-Defined Variables* in the *Using Natural* documentation. For a user-defined constant, see also `CONSTANT` in the *Statements* documentation. Instead of specifying a variable name, the filler option *n*`X` can be used. With the filler option, *n* filler bytes can be denoted within the field or variable being redefined, where *n* can be up to 10 digits (smaller than 1 GB). The definition of trailing filler bytes is optional.
F	The Natural data format of the variable or field. For valid formats, see *Format and Length of User-Defined Variables* and *Special Formats* in the *Programming Guide*. For a counter field (`C*` variable), you can specify the Natural data format/length I2 or I4 (the default setting is N3 for no format/length).
Length	The length of the variable or field. For valid lengths, see *Format and Length of User-Defined Variables* in the *Programming Guide*. No length is permitted for the Natural data formats C, D, T and L. You can define dynamic variables by specifying `DYNAMIC` in the **Length** field.

Column Heading	Explanation
	For a counter field (`C*` variable), you can specify the Natural data format/length I2 or I4 (the default setting is N3 for no format/length).
Miscellaneous	This input field can be used to enter the definitions described in ***Using the Miscellaneous Column***.

Using the Miscellaneous Column

The definitions that can be entered in the fields of the **Miscellaneous** column are described in this section.

As the **Miscellaneous** field may be too short to make all required specifications, the `.E` line command is provided for **extended field definition editing**.

A definition can be of up to 32 characters, whereby only 26 characters are displayed on the screen. You can scroll in the field by using the editor command `M +/-`. You can display all of the 32 characters or enter additional characters in an extra window, which opens when you enter a question mark (?) in the first position of the **Miscellaneous** field.

You can define the following:

Array

Enter the upper and lower bounds of an array. For detailed information on defining arrays, see *Arrays* in the *Programming Guide*.

Examples:

```
(2,2)   /* 2 dimensions, 2 occurrences
(2,2,2) /* 3 dimensions, 2 occurrences
(1:10,2)
(-1:3,2)
```

Initial Value

Not applicable to PDAs.

Enter an initial value according to the common Natural syntax definitions in a `DEFINE DATA` statement. For detailed information on defining initial values, see *Initial-Value Definition* and *Initial/Constant Values for an Array* in the *Statements* documentation.

Examples:

```
INIT<3>
INIT<'ABC'>
INIT<H'F1F2'> /* binary variable (B2)
CONST<12>
INIT ALL<'ABC'>
```

Edit Mask, Header and/or Print Mode

Edit masks and headers do not apply to PDAs.

Enter an edit mask or a header definition and/or the print mode according to the syntax rules that apply to the corresponding session parameters `EM` or `EMU`, `HD` and `PM` described in the *Parameter Reference* documentation.

Examples:

```
(EM=999.99)
(HD='TEXT' EM=XXX.XXX.XX PM=N)
```

Comment

A commentary text which must be preceded by a slash and an asterisk (/*).

Name of a DDM

For a view definition, you must enter the name of the DDM from which the view is derived.

You can modify the name of the DDM if all fields of the view are also contained in the DDM with the modified name.

Name of a Parent Block

For a block definition, you must enter the name of the corresponding parent block.

Extended Field Definition Editing

The extended field definition editing function can be used to define the following:

- Parameters and arrays within PDAs.
- Arrays, initial values, edit masks and headers within LDAs and GDAs. This is an alternative to using the **Miscellaneous** column.

To execute the extended field definition editing function

1 In the **T** column, next to the variable or field for which you want to define extended attributes, enter the following line command:

```
.E
```

An **Extended Field Definition Editing** menu similar to the example screen for a user-defined variable in an LDA is shown below:

```
17:54:34                *****  EDIT FIELD  *****                    2010-08-23
                   - Extended Field Definition Editing -

Local     *LDA2     Library SAGTEST                        DBID    10 FNR    32



         Code  Function                        Definition
        -----  ------------------------------  ----------
          S     Single Value Initialization    no
          F     Free Mode Initialization       no
          E     Edit Mask Definition           no
          A     Array Index Definition         no
          ?     Help
          .     Exit


        -----  ------------------------------  ----------

   Code   ?  for Field: #USER-VARIABLE-1(A10)
```

The functions provided on the **Extended Field Definition Editing** menu depend on the type of the data area, the type of variable and the contents of the **Miscellaneous** field. For example, if a variable has already been initialized in the **Miscellaneous** field, the functions **Single Value Initialization** and **Free Mode Initialization** are not available.

Note: If `.E` is executed for a DDM field, the **Define Edit Mask / Header** screen (see the following step) is invoked immediately, because only edit masks and headers can be defined for DDM fields. It is not possible to define initial values for DDM fields.

2 Select the function required by entering the code that corresponds to the function required. For explanations of the functions available, see *Functions in the Extended Field Definition Editing Menu*.

Depending on the function selected, either another menu or an extended field editing area similar to the example of a **Define Edit Mask / Header** screen below appears:

```
17:50:59                ******  EDIT FIELD  *****                    2010-08-23
                      - Define Edit Mask / Header -
Local     *LDA2     Library SAGTEST                        DBID    10 FNR    32
Command

#USER-VARIABLE-1(A10)
-----------------------------------------------------------------------
(EM=                                                                   )
-----------------------------------------------------------------------
Save as unicode edit mask (EMU) .. N  (Y/N)

#USER-VARIABLE-1(A10)
--------------------------------------------------------------------
(HD='                                                              ')
--------------------------------------------------------------------
```

3 Type a definition or enter a function code respectively.

Note: A definition is *not* checked for syntax errors during editing. You can check a definition with the `CHECK` command after you terminated extended field definition editing.

4 When you are finished and return to the **Extended Field Definition Editing** menu, the **Definition** column reflects the changes as shown in the following example:

```
17:59:00                ******  EDIT FIELD  *****                    2010-08-23
                  - Extended Field Definition Editing -

Local     *LDA2     Library SAGTEST                        DBID    10 FNR    32



         Code  Function                        Definition
        -----  ------------------------------  ----------
          S     Single Value Initialization    no
          F     Free Mode Initialization       no
          E     Edit Mask Definition           yes
          A     Array Index Definition         no
          D     Delete all Definitions
          ?     Help
          .     Exit

        -----  ------------------------------  ----------

   Code   ?  for Field: #USER-VARIABLE-1(A10)
```

If any initial values, edit masks, headers or array index definitions have been defined, the corresponding status message in the **Definition** column changes from `no` to `yes`. If in a PDA

any parameter type has been defined, an abbreviation of the parameter type (for example, `Val` for call-by-value) is displayed in the **Definition** column.

Any definitions made within the **Initial Values** and **Edit Mask / Header** subfunctions are immediately incorporated into the data area currently displayed in the data area editor but are not displayed in the **Miscellaneous** column of the editing area. A corresponding entry is only displayed in the **I column** (label indicator).

The functions available in the **Extended Field Definition Editing** menu and the commands available in an extended field editing area are described in the following section.

- Functions in the Extended Field Definition Editing Menu
- Commands in the Extended Field Editing Area

Functions in the Extended Field Definition Editing Menu

All functions that can be available in the **Extended Field Definition Editing** menu are described in the following table.

For an attribute control variable, only the functions codes **S**, **F**, **P**, **A** and **D** are allowed.

For a field that redefines another field, only the function codes **E**, **A** and **D** are allowed.

Function Code	Function
S	Defines an initial value for the specified variable or field in single-value mode. You only enter the required variable or field value; any further specifications necessary (including apostrophes for alphanumeric variables or fields, and value prefixes such as `H` for hexadecimal) are generated automatically. For example, from an initial value of `F1F2` for a binary variable (B2), the data editor will generate `INIT <H'F1F2'>`. If the variable or field is an array, an initial value can (but does not necessarily have to) be defined for each occurrence. For arrays of large alphanumeric variables or fields (for example, arrays of dynamic variables), enter `Y` (yes) in **Next index (Y/N)** to position to the next index of the array. With arrays, asterisk notation (*) can be entered in the command line to repeat the value in the last line of the previous page until the end of the current page. For attribute control variables, a screen is displayed where you can select attributes and colors as initial values. For details on attributes and colors, see the session parameters `AD` and `CD` in the *Parameter Reference* documentation. To define a constant value instead of an initial value, enter `Y` in the field **Define as CONSTANT (Y/N)**.

Function Code	Function
F	Defines an initial value for the specified field in free mode. A free-mode editor is provided where you can enter your initial value definitions according to the common Natural syntax definitions in a `DEFINE DATA` statement. For detailed information on defining initial values, see *Initial-Value Definition Initial/Constant Values for an Array* in the *Statements* documentation. See also ***Examples*** in *Initial Value*.
E	Defines an edit mask and/or header for the specified field according to the Natural rules for edit mask and header specifications. If both an edit mask and a header are specified, together they must not exceed 57 characters in length. However, if only an edit mask is specified, it can be up to 63 characters long; if only a header has been specified, it can be up to 58 characters long. If `.E` is entered for a DDM field, this function is invoked immediately, as only edit masks and headers can be defined for DDM fields. It is not possible to define initial values for DDM fields.
P	This function only applies to PDAs. It can be used to determine the way in which the value of a field specified as a parameter in a `CALLNAT` statement is passed from a program to an invoked object (for example, a subprogram). You can enter one of the following codes in the upper input field: `D` — Call-by-reference (default). `V` — Call-by-value. `R` — Call-by-value-result. For detailed information, see the corresponding options `BY VALUE` and `BY VALUE RESULT` described for the `DEFINE DATA` statement in *Parameter Data Definition*, and *`operand2`* described for the `CALLNAT` statement in the *Statements* documentation. Additionally, you can specify whether a parameter must be passed by entering one of the following values in the lower input field: `N` — A parameter must be passed (default). `Y` — An optional parameter that *can* be passed. For detailed information, see the corresponding option `OPTIONAL` described for the `DEFINE DATA` statement in *Parameter Data Definition*, and *`operand2`* described for the `CALLNAT` statement in the *Statements* documentation.
A	Defines array bounds for the specified field. A free-mode editor is provided where you can enter your bound definitions in accordance with the common Natural syntax definitions. While you are editing, however, the specified values will not be checked (unless you enter the `CHECK` command).

Function Code	Function
D	Deletes all definitions made with the **S**, **F**, **E**, **P** and **A** function codes. An additional screen is provided, where you can specify the definitions to be deleted. By default, all definitions are marked with `Y`. If you do not want to delete a definition, remove the `Y` behind the definition or replace the `Y` by `N`.

Commands in the Extended Field Editing Area

The commands that can be entered in the command line of an extended field editing area are described in the following table:

Command	Function
`EDIT`	Returns to the editing area of the editor screen.
`I`	Positions to the next index of the array.
`.`	Returns to the previous screen to continue processing.
`--`	Returns to the beginning of the initial value specification(s).
`+`	Scrolls down one page. If the last page has been reached or if there is only one page available, returns to the editing area of the editor screen.
`*`	Copies the initial value of the last occurrence of the previous page to all empty fields of the current page. It is only available for arrays in single-value mode.

Line Commands

You enter a line command in the **T** column of a source line. We recommend that you enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

The default escape character which must precede each line command is a period (.). You can change the default character by using the editor profile option **Escape Character for Line Command** (see ***Editor Profile***).

The line commands provided by the program editor are described in the following section. The notation `(n)`, `(nnn)` or `(nnnn)` indicates a repetition factor. The default repetition value is 1 (with the exception of the `.I` line command). For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Command	Function
`.C[(nnnn)]`	Copies the line in which the command was entered. See also *Notes for Line Commands*.

`.CX[(nnnn)]`
 	
or
`.CY[(nnnn)]`
	Copies the X-marked or the Y-marked line.
	
	See also the line commands `.X` and `.Y` and *Notes for Line Commands*.

`.CX-Y[(nnnn)]`	Copies the block of lines delimited by the X and Y markers. See also the line commands `.X` and `.Y` and *Notes for Line Commands*.
`.D`	Deletes one or more lines beginning with the line in which you enter the command towards the end of the source (regardless of any direction indicator setting). When entered for an individual field, only that field definition is deleted. When entered for a part of a hierarchical structure (view, group, redefinition), all subsequent definitions on subordinate levels are also deleted. For example, if you enter `.D` for a group defined at level 2, everything belonging to that group and with a level number greater than 2 is also deleted up to (but not including) the next level 2 definition. Comment lines (which usually are not assigned a level) are also considered to be at a subordinate level. To avoid the undesired deletion of a comment, assign an appropriate level to it.
`.D(nnnn)`	Deletes *nnnn* lines beginning with the line in which you enter the command towards the end of the source (regardless of any direction indicator setting). Unlike `.D` (see above), `.D(nnnn)` affects only the number of lines specified, regardless of any hierarchical structure.
`.E`	Invokes the **Extended Field Definition Editing** screen which is used to define array bounds, initial values, edit masks, headers and parameter attributes. For more information, see the section *Extended Field Definition Editing*.
`.F(file-name)`	This command includes a Predict file (applicable to the file types Conceptual, Standard, Sequential and Other). The name of a Predict file is shown in the **Miscellaneous** column.
`.I[(n)]`	This command adds *n* empty lines, where *n* can be in the range from 1 to 9. If *n* is not (or not correctly) specified, 10 lines (5 lines in split-screen mode) are added by default. Lines that are left blank are eliminated from the source, depending on the setting of the editor profile option **Empty Line Suppression** described in *Editor Profile*. **Note:** Only one `.I` can be performed at a time. See also *Notes for Line Commands*.

Command	Function
`.I(obj)`	Copies variable or parameter definitions from another Natural object of one of the following types: Data area Program Subprogram Subroutine Helproutine Map Function If the object specified as *obj* is not a data area, it must be available as a cataloged object. A window appears in the data area editor screen where you can select one of the following data definitions to be incorporated into your current data area: ■ All local variables and parameters contained in the specified object (including those incorporated from LDAs and/or PDAs). ■ All local variables contained in the specified object (including those incorporated from LDAs). ■ Only those local variables contained within the specified object. ■ All parameters contained in the specified object (including those incorporated from PDAs). ■ Only those parameters contained within the specified object. Additionally, you can select one of the following: ■ All unused and used variables (selected by default). ■ Only used local variables that are referenced or modified. ■ Only unused variables that are defined but not referenced or modified. If you incorporate unused variables, the level numbers might not be in the correct order. So, before compiling the data area, check the levels of all incorporated unused variables for correct numbering. If you incorporate variable definitions from objects without a `DEFINE DATA` definition (that is, from objects coded in reporting mode), variable redefinitions (see the `REDEFINE` statement in the *Statements* documentation) might be placed at the wrong position; that is, after the wrong variable. So, before compiling your new data area, check all variable definitions and redefinitions for correct positioning. If a variable redefinition results in more than one variable, each variable is incorporated as one individual redefinition by using filler bytes where appropriate.

Command	Function
	If the specified object has been cataloged with the Natural Optimizer Compiler, initial values and constants cannot be incorporated. If the object you want to insert has features the data area editor does not support, an appropriate message appears and the relevant line is marked as a comment line. See also ***Notes for Line Commands***.
`.I(obj,ssss,nnnn)`	Includes a GDA, an LDA or a PDA. This feature is only supported for data areas which do not contain initial values or edit masks. The `ssss` entry can be used to indicate at which line the insertion is to begin. For example, when setting `ssss` to `20`, the insertion begins with the 20th line of the data area. The `nnnn` entry can be used to indicate the number of lines to be inserted. If `ssss` and/or `nnnn` is specified for an object other than a data area (see the `.I(obj)` command), the specified value(s) are ignored. See also ***Notes for Line Commands***.
`.L`	Undoes all modifications that have been made to the line since the last ENTER.

`.MX`
 	
or
 	
`.MY`
	Moves the X-marked or the Y-marked line.
	
	See also the line commands `.X` and `.Y` and ***Notes for Line Commands***.

`.MX-Y`	Moves the block of lines delimited by the X and Y markers. See also the line commands `.X` and `.Y`, and ***Notes for Line Commands***.
`.N`	Marks (invisibly) a line to be positioned at the beginning of the source work area by the editor command `POINT` described in *Editor Commands for Positioning*. The mark is automatically deleted when an error with a line command or editor command occurs, or when the `RESET` command is executed.
`.P`	Positions the line marked with this command to the top of the screen.
`.R`	Redefines a variable or field as a single variable or a group of variables. With the filler option (`nX`), `n` filler bytes can be denoted within the variable or field being redefined. The definition of trailing filler bytes is optional. See also ***Notes for Line Commands***.
`.V[(ddm-name[,NOFL])]`	Not applicable to PDAs. Defines a view from a DDM. Specify the DDM (`ddm-name`) from which you want to define a view. The fields of this DDM are then displayed in the editing area. Mark the fields to be incorporated into the view by entering any character in the **I** column next to

Command	Function
	the field(s) required. When you press ENTER, these fields are copied as a view definition into the current data area with the name of the view (default is the name of the DDM) assigned at level 1. In **split-screen mode**, the DDM currently in the split screen is displayed in the editing area when you enter `.V` without *`ddm-name`*. If `.V(`*`ddm-name`*) is specified within a view of the same name as specified for *`ddm-name`*, the selected fields are included in this view and no new view is defined. If `NOFL` is specified, the selected fields are included without format and length specification. When a periodic group or multiple-value field defined - in a DDM generated with Predict - as `PC` or `MC` respectively is included in a data area, a counter field (`C*` variable) for the group or field is automatically generated and placed before the group or field. The index for such a periodic group or multiple-value field is defined with the number of occurrences defined in Predict. If the number of occurrences has not been defined in Predict, the value 191 is used. If Predict is active, Predict redefinitions and comments are incorporated too. With VSAM views, the actual number of occurrences is always displayed. In addition, VSAM views contain information on subdescriptors and superdescriptors. For further information, see the *Natural for VSAM* documentation.
`.VG`	Only applies to views. Converts all fields of a view definition to fields of a group structure. You enter `.VG` next to the line that contains a `V` in the **T** column. If the view definition was inserted with the `NOFL` option (see the `.V` line command), the Natural data format/length is read from the DDM and added to the view fields. If the view definition contains a counter field (`C*` variable), the field name changes from `C*NAME` to `C_NAME` and the Natural data format/length is set to N3. **Note:** `.VG` does not apply to edit mask or header definitions. If you want to convert a data area to a PDA, you must remove these definitions from the LDA or GDA.
`.X`	Not applicable to periodic groups, multiple-value fields or view definitions. Marks a line with an X. See also ***Notes for Line Commands***.

Command	Function
`.Y`	Not applicable to views, periodic groups or redefinitions. Marks a line with a Y. See also ***Notes for Line Commands***.
`.*`	Generates a counter field (`C*` variable) for multiple-value fields or fields within a periodic group. See also ***Notes for Line Commands***.
`number` `[(`*`nnn`*`[,`*`m`*`])]`	This command is available in split-screen mode and with a DDM in the split-screen area only. To obtain fields and groups from the split-screen area, the line number of the field or group from the split-screen area must be specified in the first column, without a period (.). Fields and groups from the split-screen area can be included as fields of a view (if *`number`* is entered inside a view) or as user-defined variables. If the selected field has the same name as the field for which the command was entered, it is substituted instead of inserted. Multiple lines can be obtained from the split screen by using the *`nnn`* notation where *`nnn`* is the number of lines to be included. The *`m`* notation can be used to specify a level number to be assigned to the field or group to be inserted. The level number in the data area can be modified. See also ***Notes for Line Commands***.

Notes for Line Commands:

- The commands `.I(`*`obj`*`)`, `.R` and `.*` are available in full-screen mode only, not in split-screen mode.
- If both the commands `.X` and `.Y` are applied to one line, it is treated as being marked with an X and with a Y; the line marker actually shown to reflect this status is a `Z`.
- If the **direction indicator** is set to + (plus sign), the copied, inserted or moved lines are placed *after* the line in which the corresponding command was entered; if the direction indicator is set to - (minus sign), the copied, inserted or moved lines are placed *before* the line in which the command was entered.

Editor Commands

The editor commands that can be entered in the command line of the data area editor are described in the following section. For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Command	Function
ADD[(*n*)]	Adds *n* blank lines. If the **direction indicator** is set to + (plus sign), the lines are added after the last line of the object being edited; if the direction indicator is set to - (minus sign), the lines are added before the first line of the object. The value for *n* can be in the range from 1 to 9. If *n* is not (or not correctly) specified, 9 lines (4 in split-screen mode) are added by default. With the next ENTER, lines that are still left blank are eliminated.
CANCEL or . (a period)	Leaves the editor. Any modifications made since the last time the SAVE command was entered are *not* saved.
CATALOG [*object-name*]	Executes the system command CATALOG which checks and catalogs the current data area definition. You must supply an object name with the command if you catalog a new data area definition or if you want to copy the current data area. See also ***Storing and Cataloging a Data Area***.
CHANGE ['*scan-value*'*replace-value*']	Scans the data area for a character string (*scan-value*) and replaces each such *scan-value* found with the character string entered as *replace-value*. Any special character which is not valid within a Natural variable name can be used as the delimiter character. Each line in which a character string is replaced is marked with an R to the left of the line. For information on how the scan operation is performed, see the SCAN command.
CHECK	Executes the system command CHECK which checks the syntax of the current data area definition. If an error is found, the erroneous line is marked with an E and an appropriate error message appears in the message line. If no errors are found, a message appears indicating successful completion of the check.

Command	Function
	CHECK also orders the entries in the **Miscellaneous** column in the following sequence: Array definition Initial value Edit mask, header and/or print mode Name of a DDM or parent block Comment
CLEAR	Executes the system command CLEAR which clears the source work area. Changes to the data area currently contained in the source work area are lost if they were not previously saved.
DX or DY	Deletes the X-marked or the Y-marked line. See also the line commands .X and .Y.
DX-Y	Deletes the block of lines delimited by the X and Y markers. See also the line commands .X and .Y.
EX or EY	Deletes lines from the top of the editing area to, but not including, the X-marked line; or from the line following the Y-marked line to the bottom of the editing area. See also the line commands .X and .Y.
EX-Y	Deletes all lines in the source work area excluding the block delimited by X and Y. See also the line commands .X and .Y.
EXIT	Leaves the editor. Any modifications to the source are saved depending on the setting of the **editor profile** described in ***Exit Function***.
GENERATE [*object-name*]	Generates a Natural object of the type copycode from the data area definition currently in the source work area. The program editor opens with the generated copycode source in the editing area including a DEFINE DATA LOCAL and corresponding END-DEFINE statement. If an *object-name* is specified, the generated copycode is saved under this name in the current Natural library in the current system file.
M +|-	Scrolls the **Miscellaneous** column. + : Scrolls to the right. - : Scrolls to the left.

Command	Function
PROFILE [*name*]	Invokes the **Editor Profile** screen where you can view or change your current editor profile settings. For details, see the section ***Editor Profile***.
READ *object-name*	Executes the system command READ which reads an existing data area definition into the source work area. For all syntax rules that apply to the command, see READ in the *System Commands* documentation.
RESET	Deletes the current X and Y line markers and any marker previously set with the line command .N. See also the line commands .X and .Y.
SAVE [*object-name*]	Executes the system command SAVE which saves the current data area definition. You must supply an object name if you save a new data area definition or if you want to copy the current data area. See also ***Storing and Cataloging a Data Area***.
SCAN *scan-value*	Scans the data area for a character string (*scan-value*) in the **Name** (default) and/or the **Miscellaneous** column of the editor screen, depending on whether the SET SCAN command was executed earlier. Each line in which the *scan-value* is found is marked with an S to the left of the line. The first line which contains the *scan-value* is positioned to the top line or the bottom line, depending on the current setting of the **direction indicator**. **Note:** The SCAN command performs an exact search for the specified *scan-value*. This should be taken into account when searching for DBCS (Double Byte Character Set) characters. If the direction indicator is set to + (plus sign), the scan is performed from the first line shown on the screen to the last line of the source work area. If the direction indicator is set to - (minus sign), the scan is performed from the last line shown on the screen to the first line of the source work area.
SCAN =[+|-]	Scans for the next occurrence of the *scan-value* specified with the SCAN command. The direction for a given scan command can be explicitly specified by entering SCAN =+ or SCAN =-. The setting of the direction indicator is then ignored. **Note:** The equal sign (=) used with the SCAN command is the default input assign character. If another character has been specified as the input assign character (see session parameter IA described in the

Command	Function
	Parameter Reference documentation), that other character must be used instead.
`SET ABS [ON|OFF]`	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 does not have to 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`. The `SET ABS` command corresponds to the editor profile option **Absolute Mode for SCAN/CHANGE** described in ***Editor Profile***.
`SET PREFIX` *`prefix`*`|OFF`	Specifies a prefix for variable or field names. This prefix is then automatically placed before the value entered in the **Name** column for each line that is entered or modified, unless the name already begins with this prefix. If the concatenated variable or field is longer than 32 bytes, an appropriate message appears and the value in the **Name** column can be shortened. If this is not done, the prefix will not be inserted.

`SET SAVEFORMAT V31|V41`
 	
or
`SET SF V31|V41`
	Specifies the default source format of data areas.
	
	If set to `V31`, data area sources are stored in compatible Natural Version 3.1 format.
	
	If is set to `V41`, data area sources are stored in extended source format (default).
	
	See also ***Source Format for Data Area Storage***.

`SET SCAN COMMENT|NAME|ALL`	Determines the column(s) in which the `SCAN` command searches for a *`scan-value`*: `COMMENT` — The **Miscellaneous** column is scanned. `NAME` — The **Name** column is scanned. `ALL` — The **Name** and **Miscellaneous** columns are scanned.

Command	Function
	The default is NAME.
SET SIZE ON|OFF	If SET SIZE is set to ON, the size of the data area is displayed in the bottom information line of the editor screen. The SET SIZE command corresponds to the editor profile option **Source Size Information** described in *Editor Profile*.
SET STAY ON|OFF	If STAY is set to ON, the current screen will stay when ENTER is pressed. Forward and backward positioning can be done by positioning commands only. If STAY is set to OFF, pressing ENTER positions to the next screen if no changes were applied to the current screen. The SET STAY command corresponds to the editor profile option **Stay on Current Screen** described in *Editor Profile*.
SET TYPE G|L|A	Changes the type of the current data area: G — Global data area L — Local data area A — Parameter data area
SPLIT *parameter*	Splits the editor screen and displays the source of another Natural object in one half of the screen as described in *Split-Screen Mode*. *parameter* represents a parameter that must be specified with the command as described in *Split-Screen Commands*.
STOW [*object-name*]	Executes the system command STOW which saves and catalogs the current data area definition. You must supply an object name if you STOW a new data area definition or if you want to copy the current data area. Otherwise, an appropriate message appears. See also *Storing and Cataloging a Data Area*.

Editor Commands for Positioning

The editor commands that can be used for navigating through the current data area are described in the following section. You enter an editor command in the command line of the data area editor.

Command	Function
ENTER or +P or +	Positions forwards one page.
-P or -	Positions backwards one page.
+H	Positions forwards half a page.
-H	Positions backwards half a page.
T or --	Positions to top of source.
B or ++	Positions to bottom of source.
+*nnnn*	Positions forwards *nnnn* lines (maximum 4 digits).
-*nnnn*	Positions backwards *nnnn* lines (maximum 4 digits).
X	Positions to the line marked with an X.
Y	Positions to the line marked with a Y.
POINT	Positions to the line in which the line command .N was entered. See also the line command .P.

Storing and Cataloging a Data Area

Before a data area can be used in a Natural program (or another object), it must be saved and cataloged as a source object and/or a cataloged object that is stored in a Natural library in the current system file.

To save and/or catalog the current data area

- Use the system command SAVE, CATALOG or STOW as described in *Saving and Cataloging Objects* in the *Using Natural* documentation.

Note: When you leave the data area editor with the `EXIT` editor command, the current source code is saved automatically if the appropriate **editor profile** option is set accordingly as described in *Exit Function*.

To keep a copy of the current source

- Use the editor options **Source Save into** and **Auto Save Numbers** as described in *Editor Profile*.

  A copy of the source edited last with any of the Natural editors is then automatically saved as a source object in the current Natural environment.

Source Format for Data Area Storage

The data area editor uses an internal source format to store the sources of data areas in the FUSER system file. New features and definitions that are available from Natural Version 4.1 onwards require that the data area source is stored in the FUSER system file using an extended source format.

The space the extended source format requires to store the extended fields of Natural Version 4.1 (and above) features and definitions in the system file is higher than the space required by the old source format.

Data areas that are stored using the extended source format cannot be used or edited with Natural Version 3.1 where a different source format was used. The data area editor of Natural Version 4.1 (and above) supports the Natural Version 3.1 format *and* the extended source format. The editor can read both formats and converts the Natural Version 3.1 format to the extended source format.

Data areas are stored in the Natural Version 4.1 source format by default.

As long as no Natural Version 4.1 (and above) features or definitions are used, data areas can also be stored in the Natural Version 3.1 format.

The source format to be used as a default for storing data areas can be specified with the user exit routine GDA-EX01 (see *User Exit for the Data Area Editor*) or, during an editor session, with the following editor command: `SET SAVEFORMAT V31` or `SET SAVEFORMAT V41`.

User Exit for the Data Area Editor

The data area editor provides a user exit routine for specifying default settings. The source of the user exit routine is provided in the library SYSEXT and named GDA-ES01. To activate this exit, `CATALOG` or `STOW` the source object as GDA-EX01 and copy GDA-EX01 to the library SYSLIB. For a detailed description, see the source object of GDA-ES01 in the library SYSEXT.

Exit Function

The exit function is used to terminate the current editor session by issuing the `EXIT` command.

The effect of the `EXIT` command depends on the setting of the **editor profile** option **Prompt Window for Exit Function:**

- If the option is set to `Y` (default setting), the `EXIT` command invokes the **EXIT Function** window whenever you execute the command on a source that contains unsaved modifications (see also **Modification Indicator**). If no modifications were made to the source, the window does *not* appear and the editor closes without saving the source.

  The **EXIT Function** window provides the following options:

Option	Explanation
Save and Exit	Saves all modifications made to the current source code and leaves the editor. See also *SAVE Object Window*.
Exit without Saving	Leaves the editor without saving any modifications made to the current source code since it was last saved.
Resume Function	Neither leaves the editor nor saves any modifications; the prompt window is closed and the current function is resumed.

- If the option is set to `N`, the `EXIT` command executes the **Save and Exit** function: saves all modifications made to the current source and leaves the editor. See also *SAVE Object Window*.

SAVE Object Window

The **Save and Exit** function invokes the **SAVE Object** window if no object name has yet been defined for the current source code (as indicated in the top information line).

This window prompts you to enter an object name and type. If you confirm your entries with ENTER, the source is saved under the specified name as a new object in the current library and the editor is terminated. PF3 closes the window without any action and the editor session is retained.

VII Map Editor

The Natural map editor is used to create a Natural object of type map. A map is a screen layout that can be referenced in a Natural object such as a program by using either an `INPUT USING MAP` statement (for input maps) or a `WRITE USING MAP` statement (for output maps).

A map contains text fields and data fields. Text fields are literal strings and data fields are variables. Data fields can be either user-defined variables or Natural system variables.

Once a map has been created, it can be stored as a source object and a cataloged object in a library in a Natural system file.

Notes:

1. When using Natural Studio in a Windows environment, the map editor supports fields with Unicode format and Unicode strings. However, when reading the source of a Unicode map into the editing area of a map editor in a local z/OS or a Linux environment, all Unicode strings will be removed from the source.
2. The map editor has been disabled in your environment by default. For more information, see *NaturalONE as the Default Development Environment* .

Related Topics:

- *Map Editor Tutorial*
- *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation

Components of the Map Editor

Summary of Map Creation

Invoking and Leaving the Map Editor

Functions in the Edit Map Menu

Initializing a Map

Editing a Map

Defining Map Fields

Extended Field Editing

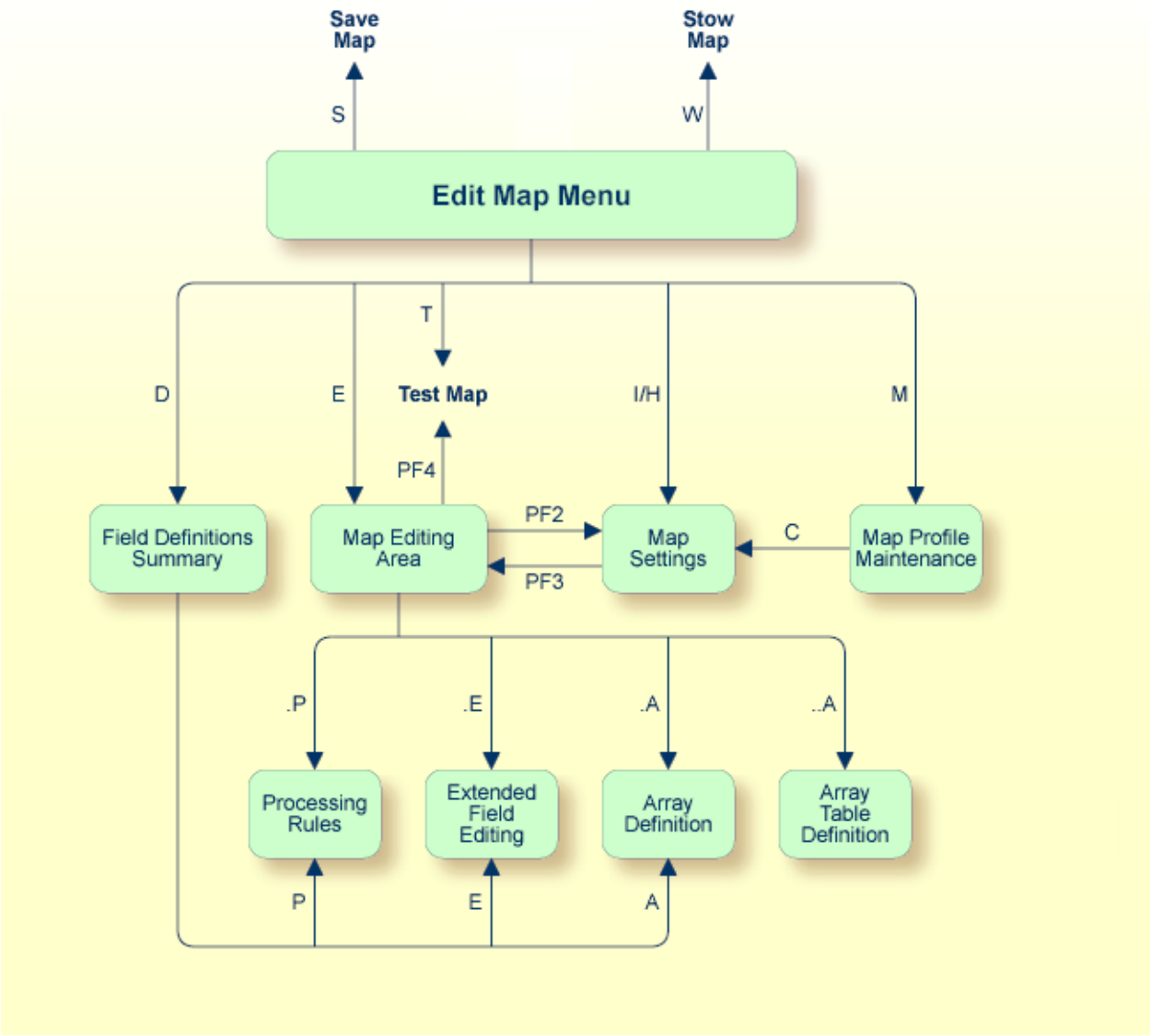
Post Assignment of Fields

Array and Table Definitions

Processing Rules

8 Components of the Map Editor

The following figure provides an overview of the various components of the map editor and also shows the possible ways to get from one component to another. The characters along the arrowed connection lines denote the function codes or commands that can be used to invoke a screen or execute a command.


Save Map
Stow Map
S
W
Edit Map Menu
T
D
E
Test Map
I/H
M
PF4
Field Definitions Summary
Map Editing Area
PF2
PF3
Map Settings
C
Map Profile Maintenance
.P
.E
.A
..A
Processing Rules
Extended Field Editing
Array Definition
Array Table Definition
P
E
A

9 Summary of Map Creation

- Step 1 - Defining a Map Profile 96
- Step 2 - Defining a Map 96
- Step 3 - Defining Map Fields 96
- Step 4 - Saving a Map Definition 97

Step 1 - Defining a Map Profile

You define a map profile (that is, the field delimiters, format settings, context settings and filler characters to be used) by selecting the required settings from a menu.

Step 2 - Defining a Map

You define a map before or after you make the corresponding data definitions in the Natural object(s) that reference the map, such as a program or data area. The two methods of defining a map are described below:

- First define a prototype map, next make the corresponding data definitions in the object that references the map, then integrate the map into the application.

  Fields can be defined directly in the map editing area. Each field is assigned a default name. Subsequently, when the corresponding data definitions have been made in the respective object, these data definitions can be assigned to the map fields (post assignment).
- Define a map by using existing data definitions.

  If data definitions already exist in an object that references the map, the map fields can be created by using the data definitions contained in this object. In this case, all characteristics of the data definitions are copied into the map.

Step 3 - Defining Map Fields

Map fields can be created by either typing the field definitions directly in the map editing area or by selecting data definitions from another Natural object as explained in *Selecting Data Definitions*.

The map editor provides the following functions:

- Full-screen or split-screen editing mode.

  Full-screen mode displays the map editing area where you actually design the map and enter the field definitions.

  In split-screen mode (this is the default setting), the upper half of the editing screen can be used to display data definitions of other Natural objects. The lower half of the screen displays the map editing area.
- Screen positioning commands.
- Line commands which are used to define tables and manipulate lines.

- Field commands which are used to define arrays and manipulate fields.
- Editor facilities which are used to edit processing (validation) rules.

Step 4 - Saving a Map Definition

Once a map has been defined as described in the previous steps, it can be saved as a source object and/or cataloged object in the current library and Natural system file. Once saved as a source object, the map can be read and modified during a subsequent map editor session. Once saved as a cataloged object, the map can be invoked from a Natural program.

Note: The map editor uses the **Auto Save Numbers** function of the program and data area editors described in *Editors - General Information*.

10 Invoking and Leaving the Map Editor

- Invoking the Editor 100
- Leaving the Editor 102

Invoking the Editor

There are different commands you can use to invoke the editor for creating a new map or editing an existing one.

To invoke the map editor for a new map

- Enter either of the following system commands:

```
EDIT MAP
```

Or:

```
E M
```

(command abbreviation)

If there is already a map in the source area, the map definition of this map is displayed in the map editing area (see also *Editing a Map*). You can clear the source area by entering the system command `CLEAR` before you invoke the map editor.

If the source area is empty, an **Edit Map** menu similar to the example below appears:

```
13:05:54                ***** NATURAL MAP EDITOR *****              2014-07-14
User SAG                         - Edit Map -                  Library SYSTEM

                   Code    Function
                   ----    --------------------------------
                     D     Field and Variable Definitions
                     E     Edit Map
                     I     Initialize new Map
                     H     Initialize a new Help Map
                     M     Maintenance of Profiles & Devices
                     S     Save Map
                     T     Test Map
                     W     Stow Map
                     ?     Help
                     .     Exit



             Code .. I     Name .. ________       Profile .. SYSPROF_

Command ===>


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Test  Edit
```

The **Edit Map** menu is the main menu of the map editor. The menu functions are described in *Functions in the Edit Map Menu*. The fields contained in the upper and lower screen section are described in the following table:

Field	Explanation
User	The Natural user ID of the current user.
Library	The Natural library ID currently in effect.
Code	The code of the function to be executed (see *Functions in the Edit Map Menu*).
Name	The source object which contains the map or help map. For multi-lingual maps, one digit of the source object name should be reserved for the language code. For example: `USERMAP1` (language code is 1) The map above is called from the program by: `INPUT USING MAP 'USERMAP&'` where `&` is replaced with the content of the system variable `*LANGUAGE` at execution time.
Profile	The session profile currently in effect. The profile name is set to the current library ID. If this profile ID is not available, it is set to the current user ID. If this profile ID is not available, the profile name is set to `SYSPROF`.

To invoke the map editor for an existing map

- Enter the following system command:

```
EDIT map-name
```

or use the abbreviation

```
E map-name
```

where *map-name* denotes the name of the map to be edited.

If a map with the specified name exists as a source object in the current library and has not been locked by another user, the map definition of this map is displayed in the editing area of the map editor (see also *Editing a Map*).

Locking of a map avoids concurrent source updates and depends on the current setting of the profile parameter `SLOCK` (see also *SLOCK - Source Locking* in the *Parameter Reference* documentation).

Or:

If the map you want to edit is still contained in the source area from an earlier editor session, you can also use the following command:

```
E M
```

Leaving the Editor

To leave the map editor

- In the **Edit Map** menu, press PF3 or enter a period (.) in the **Code** field.

  Or:

  In the command line, enter a period (.) or the command EXIT.

Note: The map editor uses the current setting of the editor profile option **Leave Editor with Unlock** which determines whether to unlock source code when leaving the map editor. This option is described in *General Defaults* in *Editor Profile* in the section *General Information*.

11 Functions in the Edit Map Menu

- Field and Variable Definitions 104
- Edit Map 105
- Initialize New Map 106
- Initialize a New Help Map 106
- Maintenance of Profiles & Devices 106
- Save Map 107
- Test Map 107
- Stow Map 107
- Help 107

Field and Variable Definitions

This function can be used to view or edit all map fields, parameters and local variables used by the map.

The **Field and Variable Definitions** function is equivalent to the line command `..E`, which can be entered in the map editing area (see also ***Line Commands*** in *Editing a Map*).

The **Field and Variable Definitions** function invokes the **Field and Variable Definitions - Summary** screen, which displays the following information for each map field:

Column	Explanation	
Name	Field name.	
Mod	Field mode (type of field):	
	D	Data area field. Field is from a local data area, global data area or parameter data area.
	S	System variable.
	U	User-defined field.
	V	View (DDM) field.
	blank	Undefined field.
Format	Natural data format and length of the field.	
Ar	Field is an array if an `A` is entered; otherwise, this column is blank.	
Ru	Number of attached processing rules.	
Lin	Line position.	
Col	Column position.	

The following line commands and PF keys are available on the **Field and Variable Definitions - Summary** screen:

Command/PF Key	Explanation
A	Defines an array.
D	Deletes a field.
E	Edits a field.
P*rr*	Edits a processing rule.
- -	Goes to the top of the screen.
.	Exits the screen.
PF9	Invokes the **PARAMETER DEFINITIONS** window to view, add or modify parameters that do not appear as map fields in the map editing area but are nevertheless associated with the map as attribute control variables, start values or help parameters.
PF10	Invokes the **LOCAL DATA DEFINITIONS** window to view, add or modify local variables that can be used to pass values from one processing rule to another.

The following commands are available in the **PARAMETER DEFINITIONS** or **LOCAL DATA DEFINITIONS** window:

Command	Explanation
A	Defines an array.
D	Deletes a variable. **Note:** Command D does not delete a parameter if this parameter is still applied to any map field as an attribute control variable, start value or help parameter.
- -	Goes to the top of the window.
.	Exits the window.

Edit Map

Invokes the map editing screen to modify an existing map or help map definition.

The map editor starts an editing session in split-screen mode. If the map being edited is a help map definition, full-screen mode is in effect.

Initialize New Map

This function can be executed only if no object with the same name is stored in the Natural system file. See also ***Initializing a Map***.

Initialize a New Help Map

This function should be used to create a help map, since it offers you the most flexibility when entering and editing text (leading blanks must be entered). It also provides additional checks to ensure that a valid help map is created.

The function can be executed only if no source object and no cataloged object with the same name is present in the Natural system file.

A help map is stored as a map and can be referenced with the parameter `HE` in the map definition.

When initializing or editing a help map, you can specify in the map settings where the help map is to appear on the screen at execution time.

See also ***Initializing a Map***.

Maintenance of Profiles & Devices

This function allows you to add, modify or delete session, map and device profiles.

A session profile is used to assign default map settings to be used when a map or a help map is initialized.

A map profile defines the map settings to be in effect during map definition and execution.

A device profile defines the standard characteristics and settings for a device. This profile can be used to ensure compatibility between the map definition and the device to be used.

See also ***Device Check*** in *Context* in the section *Initializing a Map*.

Save Map

The map definition is saved as a source object in the current Natural library in the current system file.

Test Map

The current map definition is tested to ensure that it can be executed successfully. This includes testing of all processing rules (including Predict rules) and help facilities.

When testing a map, any additionally created numeric map parameters are initialized with the value 1.

Stow Map

The map definition is saved as a source object and cataloged object in the current Natural library and system file.

Help

This function invokes the help facility of the map editor with information on all functions provided by the map editor.

To display map editor help information

1 In the **Edit Map** menu, press PF1.

Or:

In the command line of the **Edit Map** menu, enter a question mark (?).

The **Help Main Menu** appears.

2 In the **Select chapter** field, enter the number or letter that corresponds to the required help topic and press ENTER.

A help screen appears for the selected topic or another menu is invoked with further help topics that help narrow down your search.

In addition to the help facility, the map editor provides individual information on any input field available on any map editor screen.

To display help information about a field

- Place the cursor in the field for which you require help and press PF1.

  Or:

  In the field for which to invoke help, enter a question mark (?).

  Depending on the field selected, a screen with appropriate help information appears or a window from which you can select a valid input value.

12 Initializing a Map

- Delimiters 110
- Format 112
- Context 114
- Filler Characters 115

This section describes how to define the settings (profile) for a new map or help map.

When you select the function **Initialize New Map** or **Initialize a New Help Map**, the first screen to be invoked is the **Define Map Settings** screen shown in the example below:

```
13:14:20                Define Map Settings for MAP                    2014-07-14

 Delimiters            Format                           Context
 -----------------     ----------------------------     --------------------------
 Cls Att CD  Del       Page Size ...... 31              Device Check .... ________
  T   D      BLANK     Line Size ...... 79              WRITE Statement   _
  T   I      ?         Column Shift ... 0 (0/1)         INPUT Statement   X
  A   D      _         Layout ......... ________        Help  ____________________
  A   I      )          dynamic ....... N (Y/N)          as field default N (Y/N)
  A   N      ^         Zero Print ..... N (Y/N)
  M   D      &         Case Default ... UC (UC/LC)
  M   I      :         Manual Skip .... N (Y/N)         Automatic Rule Rank  1
  O   D      +         Decimal Char ... .               Profile Name .... SYSPROF
  O   I      (         Standard Keys .. N (Y/N)
                       Justification .. L (L/R)         Filler Characters
                       Print Mode ..... __              ------------------------
                                                        Optional, Partial ....
                       Control Var .... ________        Required, Partial ....
                                                        Optional, Complete ...
                                                        Required, Complete ...


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Let
```

The sections contained in the **Define Map Settings** screen are described in the following section.

Delimiters

The **Delimiters** section of the **Define Map Settings** screen displays the default delimiters that apply to the current map.

A delimiter is used to assign characteristics to a field. Field characteristics are the class (for example, input/output field), attribute (for example, typeface) and color settings of a field.

Note: Attributes, colors and print modes require corresponding hardware features, and will be ignored at runtime if these features are not available. See also the session parameters `AD`, `CD` and `PM` described in the *Parameter Reference* documentation.

Each class, attribute and color setting is denoted by a one- or two-letter code. For example, the letter code `A` identifies an input field, the letter `I` identifies intensified. Class, attribute and color can be combined in a delimiter character. A delimiter character, for example, specifies a field as

an input field (letter code A) *and* intensified (letter code I). In the **example screen** above, the delimiter character for this combination (letter codes A and I) is the right parenthesis ( )).

A delimiter character is a non-alphabetical character that is prefixed to the field in the map editing area. (See also ***Defining Map Fields*** for examples of delimiter usage.) To display or modify the class, attribute and color settings assigned to a field by a delimiter character, use the **extended field editing** function described in the relevant section. Any non-alphabetical character can be defined as a delimiter character - except the control character for terminal commands, the control character for map commands and the decimal notation character.

Letter codes and delimiter characters can be entered in the columns **Cls** (Class), **Att** (Attribute), **CD** (Color Definition) and **Del** (Delimiter) or in the editing section provided by the extended field editing function.

To change the default delimiter settings for the current map

- In the **Delimiters** section, in the columns **Cls**, **Att**, **CD** and/or **Del**, overwrite a value with the value required, or, in a blank column, enter a value.

To change the default delimiter settings for the current session

- Before you initialize a map, on the **Edit Map** screen, in the **Profiles** field, replace the default map profile SYSPROF by the name of the profile you created earlier with the function **Maintenance of Profiles & Devices** (see *Functions in the Edit Map Menu*).

  The profile SYSPROF can only be modified by the Natural system administrator.

The table below lists and explains all valid letter codes for class, attribute and color that can be entered in the columns **Cls** (Class), **Att** (Attribute) and **CD** (Color Definition) or in the extended field editing section.

Valid letter codes for classes, attributes and colors are:

Class (Cls)		Attribute (Att)		Color (CD)	
A	Input field	**B**	Blinking	**BL**	Blue
M	Modifiable field	**C**	Cursive/italic	**GR**	Green
O	Output field (non-modifiable)	**D**	Default (for example, non-intensified, non-blinking)	**NE**	Neutral
T	Text field	**I**	Intensified	**PI**	Pink
		N	Non-display	**RE**	Red
		U	Underlined	**TU**	Turquoise
		V	Reversed video	**YE**	Yellow
		Y	Dynamic (attributes to be assigned dynamically by a program)		

Format

The following map format settings can be used:

Field	Explanation
Page Size	The number of map lines to be edited (1 - 250); if **Standard 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, you specify the number of lines of the logical page output with the WRITE statement, not the map size. Thus, the map can be output several times on one page.
Line Size	The number of map columns to be edited (5 - 249).
Column Shift	Column shift (0 or 1) to be applied to the map. This feature can be used to address all 80 columns on an 80-column screen (**Column Shift** = 1, **Line Size** = 80). Positional commands (PF10, PF11) must be used to edit all map positions. The largest field you can display on a 24 x 80 screen must not exceed 79 characters. If you want to display a field with 80 characters or more, you have to adjust the **Line Size** accordingly. For example, to display a field with the format/length A80, set **Line Size** = 81.
Layout	The name of a map source definition which contains a predefined layout.
dynamic	Y: 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 PF9 within the **Field and Variable Definitions** function. N: 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.
Zero Print	Y: Displays a field value of all zeros as one zero only. N: Displays a zero value as blanks. N is the default setting. 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 (see the relevant section).

Field	Explanation	
Case Default	UC	Indicates that all input entered for fields at map execution time is to be converted to upper case, that is, the session parameter AD=T is used as a field default. See also *AD - Attribute Definition* in the *Parameter Reference* documentation.
	LC	Indicates that no lower to upper case conversion is to be performed, that is, the session parameter AD=W is used as a field default. To make the value LC effective, you have to specify the value ON for the Natural profile parameter LC. See also *AD - Attribute Definition* and *LC - Lower to Upper Case Translation* in the *Parameter Reference* documentation.
	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 (see the relevant section).	
Manual Skip	Y	Does *not* automatically move the cursor to the next field in the map at execution time even if the current field is completely filled.
	N	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.
Decimal Char	The character to be used as the decimal notation character. This character can only be changed with the GLOBALS command.	
Standard Keys	Y	Leaves the last two lines of the map blank so that function-key specifications can be entered at execution time.
	N	Causes all lines to be used for the map. N is the default setting.
Justification	The type of field justification to be used for numeric and alphanumeric fields taken from the data definitions in another Natural object:	
	L	Left justified
	R	Right justified
	This value is copied into the field definition when a new field is created.	
Print Mode	The default print mode for variables:	

Field	Explanation	
	C	Indicates that an alternative character set is to be used (special character table as defined by the Natural administrator).
	D	Indicates that double-byte character mode is to be used.
	I	Indicates inverse print direction.
	N	Indicates that no hardcopy can be made.
	This value is copied into the field definition when a new field is created.	
Control Var	The name of an attribute control variable, the content of which determines the attribute characteristics of fields and texts that have the attribute definition AD=Y or Y. The attribute control variable referenced in the map must be defined in the program using that map. Removing an attribute control variable from the format map settings implies that the attribute control variable is removed from the map, too, unless it is associated with any other map field.	

Context

The following map context settings can be used:

Field	Explanation	
Device Check	If a device name is entered in this field, the map settings are checked for compatibility with the device profile of the specified device. If a setting is not compatible, a warning message is issued (see also ***Maintenance of Profiles & Devices*** in the section *Functions in the Edit Map Menu*).	
WRITE Statement	Marking this field with a non-blank value produces a WRITE statement at the end of the map definition process. The resulting map can then be invoked from a Natural program using a WRITE USING MAP statement. Blank lines at the end of the map are automatically deleted so that the map can be output several times on one page.	
INPUT Statement	Marking this field with a non-blank value causes the result of the map definition process to be an INPUT statement. The resulting map can then be invoked from a Natural program using an INPUT USING MAP statement.	
Help	The name of a helproutine or help map which is invoked at execution time when the help function is invoked for this map (global help for map). The syntax that applies to entering values in the **Help** field corresponds to the syntax of the HE session parameter described in *HE Parameter Syntax* (*Parameter Reference* documentation).	
as field default	Y	Specifies that the helproutine or help map entered in the **Help** field is to apply as default to each individual field

Field	Explanation	
		on the map, which means that the name of each field is passed individually to the helproutine.
	N	Specifies that the name of the map is passed to the helproutine or help map. N is the default setting.
	Note: If you define the map settings for a help map, on the **Define Map Settings for HELPMAP** screen, the **Help** and **as field default** fields are replaced by the **Position Line Col** field described below.	
Position Line Col	The position where the help map is to appear on the screen at execution time. This field only appears if you define the map settings for a help map created with the **Initialize a new Help Map** function. This field replaces the **Help** and **as field default** fields on the **Define Map Settings for HELPMAP** screen.	
Automatic Rule Rank	The rank (priority) assigned to Predict automatic rules when they are linked to the map during field definition. Default is 1.	
Profile Name	The name of the profile which was active at map initialization time. If ENFORCED is displayed, the following map settings are protected: ■ All map delimiters ■ Static and dynamic layout ■ Device check ■ WRITE and INPUT statements ■ All filler characters ■ Automatic rule rank ■ Positioning of help maps The name of the profile active at the time the map is created is stored within the map. When the map is edited later and another profile is active, a warning is produced but editing is allowed.	

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
Optional, Partial	Input not mandatory, field need not be completely filled.
Required, Partial	Input mandatory, field need not be completely filled (`AD=E`).
Optional, Complete	Input not mandatory; if filled, field must be completely filled (`AD=G`).
Required, Complete	Input mandatory, field must be completely filled (`AD=EG`).

Filler characters can also be defined for individual fields using the **extended field editing** function (see the relevant section). For definition of field types, see also the session parameter `AD` described in the *Parameter Reference* documentation.

13 Editing a Map

- Screen Modes 118
- PF Keys and Commands for Positioning 118
- Line Commands 119
- Field Commands 121

Screen Modes

The map editor begins an editing session always in split-screen mode. In split-screen mode, the upper half of the map editor screen contains data definitions of other Natural objects (as described in *Selecting Data Definitions*) and/or the current delimiter settings, and the lower half of the screen contains the editing area as shown in the example below:

```
Ob _                                    Ob D CLS ATR DEL     CLS ATR DEL
 .                                       .     T  D    Blnk    T  I    ?
 .                                       .     A  D    _       A  I    )
 .                                       .     A  N    ¬       M  D    &
 .                                       .     M  I    :       O  D    +
 .                                       .     O  I    (
 .                                       .
 001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+-




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

PF9 can be used to switch between full-screen and split-screen mode.

You can leave the map editing area by pressing PF3 or by entering a period (.) in the **Ob** input fields in the upper section of the screen.

PF Keys and Commands for Positioning

The PF keys and commands listed below can be used to invoke an editor function or navigate in the map editing area; you enter the commands at the beginning of a map line:

PF Key	Command	Explanation
PF1		Invokes the map editor help facility.
PF2		Displays/modifies the current map settings.
PF3	.Q	Terminates map editing and returns to the **Edit Map** menu.
PF4		Tests the map definition.
PF5		Invokes the **extended field editing** function (see the relevant section) for the field at which the cursor is currently positioned.

PF Key	Command	Explanation
PF6	.--	Moves to top of map.
PF7	.-	Moves upwards half a window page.
	.-*nnn*	Moves upwards *nnn* lines.
PF8	.+	Moves downwards half a window page.
	.+*nnn*	Moves downwards *nnn* lines.
	.++	Moves to bottom of map.
PF9	./	Switches between split-screen and full-screen mode.
PF10	.<	Moves to the left half a window page.
	.<*nnn*	Moves to the left *nnn* columns.
	.<<	Moves to the left border of the map.
PF11	.>	Moves to the right half a window page.
	.>*nnn*	Moves to the right *nnn* columns.
	.>>	Moves to the right border of the map.
PF12		Ignores changes made on screen subsequent to last use of ENTER.
	.*	Moves top left corner to cursor position.

Line Commands

Line commands must be entered in the following form:

```
..line-command
```

where the two periods (..) represent two occurrences of the control character in effect for the map definition.

We recommend that you enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

The following line commands are available:

Command	Explanation
..A	Array table definition (see the section ***Array and Table Definitions***).
..A*n*	Array table definition with *n* occurrences. This command can be used to create a table with *n* occurrences vertically for all fields specified in the current line.
..C	Centers a single line (that is, the line in which the command was entered). Two ..C commands entered on the same screen center the first line and adjust the rest of the selected lines.

Command	Explanation
..C*n*	Centers the line and moves the *n*-1 lines below it accordingly.
..C*	Centers the line and moves all lines below it accordingly.
..D	Deletes a single line (that is, the line in which the command was entered). Two ..D commands entered on the same screen delete the block of lines delimited by these commands.
..D*n*	Deletes the line and the *n*-1 lines below it.
..D*	Deletes the line and all lines below it. If the delete operation affects array elements, the array is deleted in total.
..E	Invokes the **Field and Variable Definitions - Summary** screen of the **Field and Variable Definitions** function (see *Functions in the Edit Map Menu*) for all fields contained in the line. Two ..E commands entered on the same screen display all fields within the range of lines delimited by these commands for possible extended field editing.
..E*n*	Invokes the **Field and Variable Definitions - Summary** screen of the **Field and Variable Definitions** function (see *Functions in the Edit Map Menu*) for the line and the *n*-1 lines below it.
..E*	Invokes the **Field and Variable Definitions - Summary** screen of the **Field and Variable Definitions** function (see *Functions in the Edit Map Menu*) for the line and all lines below it. The ..E commands display a screen with the name and Natural data format/length of the requested fields. The field names shown can be modified. The **Cmd** column can be used to select the required function: **extended field editing**, **array definition** and **processing rule editing**.
..F*c*	Fills the blank spaces of a line with the character *c*.
..I	Inserts a single line. The last blank line on the screen is deleted in order to allow for the line insertion.
..I*n*	Inserts *n* lines below the line in which the command was entered.
..I*	Inserts as many lines as possible below the command line.
..J	Joins the line in which the command was entered with the line below it. Two ..J commands entered on the same screen joins the range of lines delimited by the commands.
..J*n*	Joins the line in which the command was entered with the *n*-1 lines below it.
..J*	Joins the line with all lines below it. If a join operation results in a line being too long, the lower line is split at the rightmost possible position and the left part is then joined with the previous line. The right part of the split line is then shifted to the left to align it with the line in which the command was entered.
..L	Invokes the **Modify INCDIR Statements of Map** screen which can be used to list and update the INCDIR statements generated for map fields that were copied from DDMs (data definition modules). For detailed instructions, see *Checking and Correcting References to DDMs*.

Command	Explanation
`..M`	Moves the line in which the command was entered below the cursor line. If two `..M` commands are entered on the same screen, the block of lines delimited by the commands is moved below the line marked with the cursor.
`..M`*n*	Moves the line and the *n*-1 lines below it below the line marked with the cursor.
`..M*`	Moves the line in which the command is entered and all lines below it to the line below the line marked with the cursor. This command is only practical if the line marked with the cursor is above the line in which the command is entered.
`..P`	Invokes PF-key processing rule editing. PF-key processing rules are special processing rules to define activities assigned to program sensitive PF keys.
`..P`*n*	Invokes PF-key processing rule editing for rank level *n*.
`..P*`	Lists all processing rules defined for the PF keys in this map.
`..Q`	Terminates map editing and returns to the **Edit Map** menu.
`..R`	Repeats once all literal strings on the line in which the command was entered. The cursor position is used to indicate the target line. If two `..R` commands are entered on the same screen, the literal strings within the block of lines delimited by the commands are repeated.
`..R`*n*	Repeats all literal strings on this and the *n*-1 following lines. If the cursor is located below the command line, the same text is repeated *n* times.
`..S`	Splits the line at the cursor position. If two `..S` commands are entered on the same screen, the block of lines delimited by the commands are split.
`..S`*n*	Splits the line where the command is entered and the *n*-1 lines below it at the cursor position.

Field Commands

Field commands must be entered in the following form:

```
.field-command
```

where the period (.) represents the control character in effect for the map definition. Each command must begin in the first position of a text or data field.

A field command can be applied to a range of fields or constants. A range can be specified in any of the following ways:

- Two or more of the same field commands can be used on the same screen. The column range (horizontal range) and the line range (vertical range) are determined by the positions of the commands. (The ***Map Editor Tutorial*** provides examples which illustrate this.)
- A repetition factor *n* can be used. It can be enclosed within parentheses. The command is applied to the designated field and also to the fields in the *n*-1 lines below it. A repetition factor * (asterisk) causes repetition until the bottom of the map is reached.

We recommend that you enter a blank at the end of each field command. This prevents the editor from attempting to interpret part of the field as part of the field command.

The following field commands are available:

Command	Explanation
.A	Defines an array. This command can be applied to a single field only and not to a range of fields. The **array definition** (see *Array and Table Definitions*) is **specified** on the screen provided. The resulting array is positioned with its left upper corner at the position where this command was entered. An array can be redefined by applying the .A command to one of its elements.
.A*n*	Supplies a repetition factor *n* with the .A command for the purpose of defining a one-dimensional array (no spacing, no offsets) without having to use a separate screen.
.C	Centers a field or a range of fields between adjoining fields. To center a single field, enter .C in the field to be centered. To center a range of fields, enter .C in the first and last field to be centered, or enter .C in the first field and position the cursor to the last field to be centered. In the event that an adjoining field or fields are not present, the column boundaries in effect for the map definition are used instead.
.D	Deletes a field or a range of fields. To delete a single field, enter .D in the field to be deleted. To delete a range of fields, enter .D in the first and last field to be deleted. The field range to be deleted may extend beyond a single line. If an array element is deleted, the entire array is deleted.
.E	Invokes the **extended field editing** function (see the relevant section) for the field at which the cursor is currently positioned. This command can be applied only to a single field and not to a range of fields. Extended field editing can also be invoked by positioning the cursor to the selected field and pressing PF5.
.J	Joins fields located on consecutive lines. The left boundary of the join operation corresponds to where the .J command is entered and the right one corresponds to the cursor position.

Command	Explanation
.M	Moves a field or a range of fields. To move a single field, enter .M in the field to be moved and place the cursor at the target position. To move a range of fields, enter .M in the first and last field to be moved and place the cursor at the target position.
.P[*n*]	Edits **processing rules** (see the relevant section) for a field. Supply a parameter *n* with the .P command to indicate the priority (rank) of the processing rule to be edited. If necessary, the value specified for *n* can be enclosed in parentheses (()).
.R	Repeats (copies) a field or a range of fields. To copy a single field, enter .R in the field to be copied and place the cursor at the target position. To copy a range of fields, enter .R in the first and last field to be copied and place the cursor at the target position. Repetition is always done downwards and from left to right. Fields generated by this command are assigned a dummy name. A valid name for each such field must be defined by using the post assignment function or the **extended field editing** function (see the relevant section). **Note:** Arrays cannot be copied.
.S	Splits (moves) a line or a range of lines. Enter .S in the field at which splitting is to begin and place the cursor at the target position. The line is divided at the position where the .S command was entered. The right portion is then moved to the cursor position.
.T	Truncates (deletes) a field or a range of fields from a line. Enter .T in the field at which truncation is to begin. If this command is used to truncate (delete) an array element, the entire array is deleted.

14 Defining Map Fields

- Defining Fields Directly on the Screen 126
- Selecting Definitions from Other Objects 126
- Using System Variables in a Map Definition 131

Defining Fields Directly on the Screen

Map fields are defined directly in the map editing area by entering a delimiter character followed by the number of positions to be allocated for the field. The following characters can be used:

Character	Meaning
9	Numeric position
0	Numeric right justified
.	Decimal notation (numeric field only)
S	Sign position (numeric field only)
HH	Hexadecimal (binary) (must be entered in groups of two)
X	Alphanumeric position

A repetition factor can also be specified in the form `(n)`, for example, `X(5)` is equivalent to `XXXXX`.

The following are examples of field definitions (the delimiter character can be changed as required):

:999	3 positions, numeric
:000	3 positions, numeric right justified
:99.9	3 positions numeric with decimal point
:S9(6)	6 positions, signed numeric
:HHHH	4 positions, hexadecimal
:X	1 position, alphanumeric
:X(7)	7 positions, alphanumeric

Fields entered as shown above are assigned a dummy field name by the map editor. Each field must be assigned a name prior to map execution by using either the **extended field editing** or **post assignment function** (see the relevant sections). Other field formats can be specified using the extended field editing function.

Selecting Definitions from Other Objects

You can define a map by selecting definitions from the local data definition of another Natural source object. Data definitions are either field definitions in a DDM or variable definitions.

For a list of objects that can be used to select data definitions, see *Object Types and Codes*.

Restrictions of Selection:

- Programs, subroutines, subprograms, helproutines and functions can only be used if they contain a `DEFINE DATA LOCAL` statement. The definitions of a `DEFINE DATA USING` statement are not available for selection.
- You cannot select fields of the Natural data format U (Unicode), C (Attribute Control) or Handle (object handles). They are not available for selection.
- If you use constants for the bounds of an array and these constants are not defined in the local data definition of the selected Natural source object, the bounds are lost when you select the array.
- You cannot select an item preceded by a period (.), such as a group.
- Input and modifiable (input/output) fields can only be specified once in the map editing area.

This section covers the following topics:

- Listing and Copying Data Definitions
- Checking and Correcting References to DDMs
- Object Types and Codes

Listing and Copying Data Definitions

To list and copy definitions from other Natural objects

1 Switch on split-screen mode, if required.

2 In the **Ob** (Object) input fields in the upper left screen section, enter a valid object code (see also ***Object Types and Codes***) followed by an object name, for example, `P TEST` as shown in the **example** in Step 4. If required, delete any remaining characters in this line.

Or:

Leave the first input field (denoted by an underscore) of the upper left **Ob** input fields blank and enter a question mark (?) in the second input field or press PF1 (Help). If required, delete any remaining characters in this line.

When you press ENTER, the **Select an Object** screen appears with a list of all objects contained in the current library. You can press ENTER to scroll don the list. In the **Select** field, enter the number that corresponds to the object you want to select.

3 Press ENTER.

The data definitions in the `DEFINE DATA` section of the Natural program (in the following **example**, program TEST) are displayed in the upper left section of the map editing screen.

To scroll in the list of data definitions, in the **Ob** input fields, overwrite the existing entries (in the following **example**, `P TEST`) with any of the following positioning commands and leave the remaining characters:

Command	Explanation
+	Scrolls down one page in the list.
++	Scrolls to the end of the list.
-	Scrolls up one page in the list.
–	Scrolls to the beginning of the list.
+*n*	Scrolls down *n* lines.
-*n*	Scrolls up *n* lines.

4 To display the data definitions of another object, in the **Ob** input fields in the upper right screen section, enter a valid object code (see ***Object Types and Codes***) followed by an object name and delete any remaining characters in this line.

For example:

```
Ob P TEST                                Ob L LDA01
1 #NAME-START                A20         A #NAME-START                A20
2 #NAME-END                  A20         B #NAME-END                  A20
3 #MARK                      A1          C #PERS-ID                   A8
. EMPLOYEES-VIEW             *V1         D #MAKE                      A20
4 PERSONNEL-ID               A8          E #MODEL                     A20
5 NAME                       A20         .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

Or:

Select an object from a list as described earlier in **Step 2**.

5 Copy data definitions into the map editing area:

- In the map editing area, in any position of a blank line, enter a delimiter character followed by the sequential number or letter displayed next to the data definition required. For example:

  ```
  (3
  ```

  Selects the data definition assigned to the number 3 (in the example above, `#MARK`).

```
:C
```

Selects the data definition assigned to the letter C (in the example above, `#PERS-ID`).

- Press ENTER.

  The characteristics of the data definitions (name, format, length or array definitions) are copied into the map.

  You can display or modify the copied values by using the map editor's field editing functions described in the sections ***Extended Field Editing*** and ***Array and Table Definitions***.

Checking and Correcting References to DDMs

For each map field copied from a DDM, an `INCDIR` statement is automatically generated into the map source. The `INCDIR` statement contains the name of the DDM file and field referenced by the map field. You can check whether an `INCDIR` statement contains an invalid DDM reference by switching on the `CHKRULE` option of the `COMPOPT` system command. Natural then issues an appropriate error message when you catalog the map definition. For detailed information, see *CHKRULE - Validate INCDIR Statements in Maps* in the *System Commands* documentation.

From the map editing area, you can list all `INCDIR` statements and change the name of a DDM if required.

To list and update single or multiple `INCDIR` statements

1 In any line of the editing area, enter the following line command:

```
..L
```

A **Modify INCDIR Statements of Map** screen similar to the example below appears with a list of all `INCDIR` statements generated into the source of the current map definition:

```
16:24:23        - Modify INCDIR Statements of Map  MAPTEST  -       2008-05-14
                                                                   Top of List

      DDM Name                          Field Name
      AUTOMOBILES                       MAKE
      AUTOMOBILES                       OWNER-PERSONNEL-NUMBER
      FINANCE                           BANK
      AUTOMOBILES                       MODEL
      EMPLOYEES                         PERSONNEL-ID
      AUTOMOBILES                       BODY-TYPE
      EMPLOYEES                         FIRST-NAME
      AUTOMOBILES                       NUMBER-OF-CYLINDERS
      EMPLOYEES                         MIDDLE-I
      AUTOMOBILES                       HORSEPOWER
      EMPLOYEES                         NAME
      AUTOMOBILES                       PISTON-DISPLACEMENT
      EMPLOYEES                         MIDDLE-NAME
      AUTOMOBILES                       WEIGHT
      EMPLOYEES                         MAR-STAT
      AUTOMOBILES                       COLOR



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        GenC  --    -     +     ++
```

The list contains the names of all DDM files and DDM fields from which you copied a map field.

2 If required, change the DDM name for a `INCDIR` statement by replacing the name(s) in the **DDM Name** column.

Or:

Press PF5 to open the **Generic Change** window. In the **New DDM Name** field, enter the name of the new DDM to be used by all `INCDIR` statements that reference this DDM.

The DDM name(s) in the corresponding `INCDIR` statement(s) are updated when you save the map definition by using either the **Save Map** or **Stow Map** function.

Object Types and Codes

Listed below are the object types and the corresponding object-type codes that can be used to select data definitions:

Object Type	Type Code
Parameter Data Area	A
Predict Conceptual Files (only if Predict is installed)	C
Global Data Area	G
Helproutine	H
Local Data Area	L
Map	M
Subprogram	N
General Object	O
Program	P
Subroutine	S
View (DDM)	V
Function	7

Note: Object type `O` (General Object) is offered as a convenience option, allowing the user to import data from a source without specifying the correct object type. This is useful, if the user can specify an object name, but can not specify the object type. It is then possible to enter `O objectname` in the **OB** field of the map editor (where `objectname` represents the object name). The editor then automatically changes the object type from `O` to the correct type and imports the data.

Using System Variables in a Map Definition

Natural system variables can also be specified in a map definition. For information about system variables, see *System Variables and System Functions* in the *Programming Guide* and the detailed variable descriptions in the *System Variables* documentation.

A system variable must be preceded by an output delimiter as indicated in the following examples:

```
(*TIME
(*DATE
(*APPLIC-ID
```

15 Extended Field Editing

- Invoking and Terminating Extended Field Editing 134
- Fields in the Extended Field Editing Area 135
- Fields in the Extended Text Field Editing Area 139

The extended field editing function is used to define additional attributes for fields.

Invoking and Terminating Extended Field Editing

To invoke extended field editing

- From within the map editing area:

  In the line that contains the field(s) for which you want to define additional attributes, enter the following line command:

  ```
  ..E
  ```

  For additional options, see also ***Line Commands*** in the section *Editing a Map*.

  Or:

  From within the map editing area:

  In the field for which you want to define additional attributes, enter the following field command:

  ```
  .E
  ```

  Or:

  From the **Edit Map** menu:

  Execute function `D` (**Field and Variable Definitions**) for the map that contains the required field(s).

  On the **Field and Variable Definitions - Summary** screen that appears, next to the field for which you want to define additional attributes, enter the following line command:

  ```
  .E
  ```

  For a data field, an extended field editing area similar to the example below is displayed in the upper section of the map editor screen:

```
Fld #001                                                          Fmt A10
--------------------------------------------------------------------------------
AD= MIT_______      ZP=         SG=        HE= ______________________   Rls 0
AL= _____           CD= __      CV= ________________________________   Mod Undef
PM= __  DF=                     DY= ______________________________
EM= ___________________________ SB= ___________________________________

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----

.EXXXXXXXXX



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  <---  --->  --    -     +           <     >     Let
```

It is possible to invoke the extended field editing function for the next or previous field in the map editing area by pressing PF4 or PF5 respectively.

To terminate extended field editing

- Press PF3.

  Or:

  Choose ENTER.

Fields in the Extended Field Editing Area

The fields contained in the extended field editing area of the editor screen are described in the following table:

Field	Explanation
Fld **Arr**	The field or array name. If the name is longer than the available space, enter the command `.E` at the beginning of the line to open additional space. The name entered in the **Fld** or **Arr** field depends on the method used when creating the field: ■ If the field was copied from a variable in another Natural object, the variable name used in this object is entered. ■ If the field was copied from a DDM field, the name of the DDM followed by the field name used in the DDM is entered; for example, `EMPLOYEES.PERSONNEL-ID` where `EMPLOYEES` denotes the name of the DDM and `PERSONNEL-ID` the name of the field defined in this DDM. ■ If the field was specified as a Natural system variable, the name of the specified variable is entered.

Field	Explanation
	■ If the field is neither of the above, it is assigned a dummy name. You must assign a name to a field prior to map execution. The name of a field can be changed. However, a prefix cannot be used for a field which did not have a prefix assigned previously. To obtain a prefixed field name, select the field from a data definition in another Natural object. **Note:** Duplicate field names are only allowed for fields defined as output-only fields. See the section ***Defining Map Fields*** for additional information.
Fmt	The Natural data format and length of the field. These can be changed by overwriting the current entry. The data formats U (Unicode), C (Attribute Control) and Handle (object handles) are *not* allowed. To define a reference to a dynamic alphanumeric variable, specify `(D)` or `DYNAMIC` behind the entry. The `AL` parameter will automatically be set either to the specified value or the maximum length available on the screen.
AL or **FL** or **NL**	The length to be used when displaying the field. For dynamic variables and long variables the length is automatically set, but can be modified. These fields correspond to the session parameters `AL`, `FL` and `NL` respectively. For detailed information on using this field and valid input values, see *AL - Alphanumeric Length for Output*, *FL - Floating Point Mantissa Length*, and *NL - Numeric Length for Output* in the *Parameter Reference* documentation.
Rls	The number of processing rules currently defined for the field.
ZP	Zero printing. You can only enter a value in **ZP** if the field is numeric or a time system variable. This field corresponds to the session parameter `ZP`. For detailed information on using this field and valid input values, see *ZP - Zero Printing* in the *Parameter Reference* documentation.
SG	Sign position. You can only enter a value in **SG** if the field is numeric or a time system variable. This field corresponds to the session parameter `SG`. For detailed information on using this field and valid input values, see *SG - Sign Position* in the *Parameter Reference* documentation.
PM	Print mode. This field corresponds to the session parameter `PM`. For detailed information on using this field and valid input values, see *PM - Print Mode* in the *Parameter Reference* documentation.
DF	Date format (only applies to date fields). This field corresponds to the session parameter `DF`. For detailed information on using this field and valid input values, see *DF - Date Format* in the *Parameter Reference* documentation.

Field	Explanation
DY	Dynamic string attributes. The dynamic string parameter is used to define certain characters contained in the text string of an alphanumeric variable to control the attribute setting. This field corresponds to the session parameter DY. For detailed information on defining dynamic attributes, see *DY - Dynamic Attributes* in the *Parameter Reference* documentation. If your require more than 30 characters for your DY definition, enter a plus sign (+) in the first position of the **DY** field and press ENTER. A **DY Extension** window opens in which you can enter a maximum of 59 characters. A DY definition that extends beyond the **DY** field is indicated by a plus sign (+) in the **DY** field. Overwrite this plus sign (+) with any character to invoke the **DY Extension** window and view, edit or delete the extended definition. **Important:** You can only enter an extended DY definition if no control variable (see also the **CV** field) has yet been defined for this map field.
HE	Helproutine or help map. The **HE** option is used to assign a helproutine or a help map to the map field. A helproutine or help map is then invoked at execution time when a help request is made for the map field. For detailed information, see the description of the HE session parameter in *HE Helproutine* in the *Parameter Reference* documentation. In the **HE** field, you can enter the name of a helproutine or help map and the parameters to be passed to this helproutine or help map. If you require additional input space, use the field command .E or enter a plus sign (+) in the field to invoke an extra window with additional input lines. The syntax that applies to specifying names and parameters in the **HE** field corresponds to the syntax of the HE session parameter described in *HE Parameter Syntax* (*Parameter Reference* documentation). In addition to the syntax explanations provided there, the following applies when using the map editor: ■ *operand1*: ■ If a variable name is specified which corresponds to the name of a map field, this field must be in the Natural data format/length A8. ■ If a variable name is specified for which no map field yet exists, a map parameter with that name is automatically defined in the Natural data format/length A8. ■ *operand2*: If a variable name is specified for which no map field yet exists, a map parameter with that name is automatically defined in the Natural data format/length N7. Removing a parameter from the **HE** field implies that the parameter is also removed from the map, unless the parameter is a map field or is associated with any other map field as a help parameter or **Starting from** value (see *Array Definition*).

Field	Explanation
AD	Field attributes. This field corresponds to the session parameter `AD`. For detailed information on using this field and valid input values, see *AD - Attribute Definition* in the *Parameter Reference* documentation. For source optimization reasons, the default values `D`, `H`, `F` and `W` are accepted but not retained (see also the session parameter `AD`).
CD	Color attributes. This field corresponds to the session parameter `CD`. For detailed information on using this field and valid input values, see *CD - Color Definition* in the *Parameter Reference* documentation.
CV	Attribute control variable for dynamic field attributes. This field corresponds to the session parameter `CV`. For detailed information on using this field and valid input values, see *CV - Attribute Control Variable* in the *Parameter Reference* documentation. **Note:** Removing an attribute control variable from a field implies that the attribute control variable is removed from the map, too, unless it is associated with any other map field.
EM or **EMU**	Edit mask (**EM**) or Unicode edit mask (**EMU**) to be used for the field. You can enter the toggle command `.U` in this field to switch between the two. **EM** is the default setting. This field corresponds to the session parameter `EM` or `EMU` respectively. For detailed information on using this field and valid input values, see *EM - Edit Mask* and *EMU - Unicode Edit Mask* (*Parameter Reference* documentation) and *EMU, ICU, LCU, TCU versus EM, IC, LC, TC* (*Unicode and Code Page Support* documentation). If you require additional input space, enter the command `.E` in the first position of the field to invoke an extra window with additional space. In the map editing area, a field that uses an edit mask is denoted by an `M`.
SB	The name of an array in which the values for a selection box are provided. The indicator `V` will be displayed to show that a selection box is available. The format of the source field applies. You can change it in the **Parameter Definitions** window.
Mod	Mode indicates how the field was created:

`Data`	The field was created by selecting a field from a `DEFINE DATA` definition.
`Sys`	The field is a system variable.
`Undef`	The field was created directly on the screen and has a dummy name.
`User`	The name of the field was created using the **extended field editing** function (see the relevant section).

Field	Explanation	
	View	The field was created by selecting a field from a view (file).

Fields in the Extended Text Field Editing Area

This section only applies if your terminal or terminal emulation supports viewing boxes.

The fields contained in the extended text field editing area of the editor screen are described in the following table:

Field	Explanation
AD	Field attributes. This field corresponds to the session parameter AD. For detailed information on using this field and valid input values, see *AD - Attribute Definition* in the *Parameter Reference* documentation. The default value assigned is D.
CD	Color attributes. This field corresponds to the session parameter CD. For detailed information on using this field and valid input values, see *CD - Color Definition* in the *Parameter Reference* documentation.
PM	Print Mode. This field corresponds to the session parameter PM. For detailed information on using this field and valid input values, see *PM - Print Mode* in the *Parameter Reference* documentation. If you enter D in this field (PM=D) to define an extended text field as a native DBCS field, the characters of the extended text field are represented by one or more pairs of the letter K in the map editing area. See also **Value**.
Hex Editing	If you enter Y (Yes), the **Text Field Hex Editing** window appears in which you can modify the hexadecimal equivalent of the non-blank or blank value entered in the **Value** field. A blank value is represented by a series of the hexadecimal values 00. You can create a blank text field with the map editor by replacing the hexadecimal values of an existing text field by 00. For example, for a text field with value TEST (see **Value** below), you would replace its equivalent hexadecimal string A385A2A3 by 00000000. The default setting of the **Hex Editing** field is N (No).
Value	The alphanumeric character string of an extended text field or a blank value for an extended blank text field, which is represented by a number of periods (.) in the map editing area. This field can also contain a double-byte character if D is entered in the **PM** field. If the terminal emulation supports DBCS, **Value** shows the decoded graphic of such a character. You can change the current value by replacing either the character string in the **Value** field or the equivalent hexadecimal values entered in the **Text Field Hex Editing** window.

Field	Explanation
	Value may *not* contain any blank characters.

16 Post Assignment of Fields

A field which has been previously defined in the screen layout of a map may be assigned the field name and field attributes of a DDM field definition 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 from a DDM can be redefined by using the appropriate DDM field definition from this DDM or a `DEFINE DATA` definition.

Post assignment can be done by entering the number (or letter) assigned to a DDM field definition as described in *Selecting Data Definitions*.

Post assignment can only be done if the format of the layout agrees with the field definition. `N` and `P` are considered to be identical numeric.

Post assignment cannot be used for view arrays (in a DDM field definition or a `DEFINE DATA` definition) if one or more dimensions of that array are smaller than the dimensions of the array in the layout.

If a length conflict occurs, an **AL**/**FL**/**NL** attribute is generated to map the field definition to the layout definition with truncation or expansion. Data is truncated when **AL**/**FL**/**NL** is specified.

17 Array and Table Definitions

- Array Definition 144
- Table Definition 148

The array definition function is used to define the occurrences and layout of an array (X-arrays are not supported).

Array definition is initiated by the field command `.A` or by issuing the line command `..E` and then marking the required field with the function code `A`.

The table definition function is used to define the occurrences and layout of more than one array at the same time. The arrays must begin in the same map line.

Table definition is invoked by the line command `..A`.

Array Definition

The upper portion of the following screen is displayed for the purpose of array definition:

```
Name #001                           Top Dim 1__________ 1__________ 1__________
-------------------------------------------------------------------------------
Dimensions                       Occurrences    Starting from     Spacing
0 . Index vertical               1__            ___________       0   Lines
0 . Index horizontal             1__            ___________       1   Columns
0 . Index (h/v) V                1__            ___________       0   Cls/Ls

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
   .AXXXXX


                Please enter starting name .AXXXXXXXXXXXXXXXXXXX




 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Mset  Exit              --    -     +           <     >     Let
```

You can specify the following:

Field	Explanation
Top Dim	Indicates the top dimension of the array; that is, the highest occurrence (from left to right) in 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.

Field	Explanation
	If you select a map array from a data definition in another Natural object, the dimensions of the map array must not exceed the dimensions shown in this field. If you do not select a map array from a data definition, the dimensions of the map array must not exceed the dimensions as defined in the Natural program.
Dimensions	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.
Occurrences	The number of occurrences to be defined for a dimension.
Starting from	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. If the variable is not defined otherwise as a field in the map, it is assumed to be of Natural data format/length N7. If so, it can be edited using PF9 in the **Field and Parameter Definition** screen. **Note:** Removing a **Starting from** value from an array implies that the variable is removed from the map, too, unless it is a map field or it is associated with any other map field as a **Starting from** value or help parameter. To edit **Starting from** values, press PF9 in the **Field and Variable Definitions - Summary** screen.
Spacing	The number of blank lines (for vertical dimensions) or blank columns (for horizontal dimensions) to be inserted between each dimension occurrence.

Modifying the Top Dimension

If you need to modify the top dimension, e.g. because you want to allow more lines on a map, proceed as follows:

- Rename each field whose size is to be modified so that there is no such declaration in any of the sources in the current library. For example, overwrite the first character by “X”.
- Now, update each top dimension. As there is no longer a relation to a definition, this field should be accessible for changes.
- Rename each array field again, as it was before step 1. That is, undo the renaming of the individual array fields.
- Adapt the programs that define the sizes of those arrays.

Examples of Array Definitions

Example 1:

A one-dimensional array consisting of 10 vertical occurrences with 2 blank lines to be inserted between neighboring occurrences.

```
Name #001                           Top Dim 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.

```
Name #001                           Top Dim 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 neighboring occurrences. The second dimension consists of 5 horizontal occurrences with 2 blank columns between neighboring occurrences.

```
Name #001                           Top Dim 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.

```
Name #001                           Top Dim 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 neighboring occurrences. The second dimension consists of 5 horizontal occurrences with 2 blank columns between neighboring occurrences. The third dimension consists of 2 occurrences, expanded vertically within each occurrence of the first dimension.

```
Name #001                           Top Dim 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 of using **Starting from**. The first dimension consists of 10 vertical occurrences starting from index `I`. `I` is defined in the map editor with Natural data format/length N7 by default. The second dimension consists of 5 horizontal occurrences starting from the index 3.

```
Name #001                           Top Dim 10_________ 5__________ 1__________
--------------------------------------------------------------------------------
Dimensions                       Occurrences    Starting from     Spacing
1 . Index vertical               10_            I_______          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.

```
Name #001                           Top Dim 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
```

Table Definition

A table of one or more arrays which all begin in the same map line is defined with the `..A` line command. When you enter the `..A` line command, the following screen is invoked:

```
14:41:47                ***** NATURAL MAP EDITOR *****               2007-10-22
                          - Array Table Definition -

Main   Index:  Vert. Occur.   1    Starting from ___________ Spacing 0   Lines
Second Index:  Direction(H/V) V                   ___________         0   Cls/Ls
Third  Index:  Direction(H/V) V                   ___________         0   Cls/Ls
-------------------------------------------------------------------------------
Name of Variable         Col           Dimension Size            Order 2.  3.
(truncated)              Pos Ind1        Ind2        Ind3        M S T Occ Occ
-------------------------------------------------------------------------------
#001                       2 1           1           1
#002                      25 1           1           1









  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Mset  Exit              --    -     +                       Let
```

Note: When applying the `..A` command to arrays which were not defined by an `..A` command but by an `.A` command, may result in a modification or even a destruction of these arrays.

The example screen above contains the following fields:

Field	Explanation
Main Index	The number of vertical occurrences, the starting position and the number of lines to be skipped between dimension occurrences.
Second Index	The direction (horizontal or vertical), the starting position and the number of lines/columns to be skipped between dimension occurrences. The second dimension only applies if one of the arrays has more than one dimension. In this case the second dimension can be displayed either horizontally (in which case there must be enough space in the line for all selected occurrences) or vertically (in which case there must be enough lines on the map to display main dimension times second dimension occurrences, including line spacing).
Third Index	The direction (horizontal or vertical), the starting position and the number of lines/columns to be skipped between dimension occurrences. The third dimension only applies if one of the arrays has more than two dimensions. In this case the third dimension can be displayed either horizontally (in which case there must be enough space in the line for all selected occurrences) or vertically (in which case there must be enough lines on the map to display main dimension times second dimension times third dimension occurrences, including line spacing).
Name of Variable	All names of field arrays contained in the table are displayed.
Col Pos	The column position in which the field is located. This is displayed for informational purposes only.
Dimension Size	The upper bounds **Ind1**, **Ind2** and **Ind3** of an array. The dimensions of the array defined in the map must not exceed the dimensions of the corresponding array defined in the Natural object that invokes the map.
Order	The order in which the dimensions are to be defined: **M**, **S** and **T** correspond to Main, Second and Third.
2. Occ.	The number of occurrences to be defined for the second index.
3. Occ.	The number of occurrences to be defined for the third index.

Example of a Table Definition

This is an example of defining map fields that correspond to the following program definition:

```
DEFINE DATA
  1 ARRAY1 (A3/1:10)
  1 ARRAY2 (A5/1:10,1:2)
  1 ARRAY3 (A7/1:10,1:2,1:3)
END-DEFINE
```

Table Definition:

```
14:41:47                ***** NATURAL MAP EDITOR *****              2006-07-24
                          - Array Table Definition -

Main   Index:  Vert. Occur.   1    Starting from ___________ Spacing 0   Lines
Second Index:  Direction(H/V) V                   ___________         0   Cls/Ls
Third  Index:  Direction(H/V) V                   ___________         0   Cls/Ls
--------------------------------------------------------------------------------
Name of Variable         Col           Dimension Size            Order 2.  3.
(truncated)              Pos Ind1        Ind2        Ind3        M S T Occ Occ
--------------------------------------------------------------------------------
  ARRAY1                  3  10          1           1           1
  ARRAY2                 32  10          2           1           1 2    2
  ARRAY3                 58  10          2           3           1 2 3  2  3









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit              --    -     +                       Let
```

`ARRAY1` is a one-dimensional array with ten occurrences. The first two occurrences are expanded in the table.

`ARRAY2` is a two-dimensional array. The first index consists of ten occurrences and the second index consists of two occurrences. The first two occurrences of the first index and both occurrences of the second index are expanded in the table.

`ARRAY3` is a three-dimensional array. The first index consists of ten occurrences, the second index consists of two occurrences and the third index consists of three occurrences. The first two occurrences of the first index, both occurrences of the second index and all three occurrences of the third index are expanded in the table.

Table Layout:

```
(*DATE                                                                  (*TIME
           Map containing an array table of multi-dimensional arrays

  ARRAY1 (1-dim.)            ARRAY2 (2-dim.)          ARRAY3 (3-dim.)

  :xxxxxxxxxx                :xxxxxxxxxx                :xxxxxxxxxx
                                                        :xxxxxxxxxx   Third Index
                                                        :xxxxxxxxxx   (3 vertical
                             Second Index               occurrences)
                             (2 vertical
                             occurrences)
                Main Index
                (2 vertical  :xxxxxxxxxx                :xxxxxxxxxx
                occurrences)                            :xxxxxxxxxx
                                                        :xxxxxxxxxx

  :xxxxxxxxxx                :xxxxxxxxxx                :xxxxxxxxxx
                                                        :xxxxxxxxxx
                                                        :xxxxxxxxxx

                             :xxxxxxxxxx                :xxxxxxxxxx
                                                        :xxxxxxxxxx
                                                        :xxxxxxxxxx
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
      Help  Mset  Exit  Test  Edit  Top   -     +     Full  <     >     Let
```

The table is defined as a collection of arrays which share the following characteristics:

- The number of occurrences of the main index must be the same for each array of the table. The main index is always expanded vertically.
- All elements of a specific index must be placed in the same line. Thus, spacing between the elements of a specific index depends on the array with the largest dimension.

18 Processing Rules

- Field-Related Processing Rules 154
- Function-Key-Related Processing Rules 155
- Processing Rule Editing 155

Field-Related Processing Rules

Three types of field-related processing rules can be defined:

- Inline processing rules
- Predict free rules
- Predict automatic 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.

Predict free rules have a name assigned and are stored in Predict.

To edit a free rule, enter the rule during map creation and assign a name to it. Inline rules can become Predict rules (and vice versa) by assigning/removing the rule name.

Predict automatic rules apply to database fields and are defined by the Predict administrator. When a field is selected from the data definitions in another Natural object, 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 settings (default is `1`).

Automatic rules cannot be modified using the map editor. They can, however, be assigned a different rank either by using the command `P=n` or by just overwriting the old rank.

Note: If a field with linked Predict processing rules is renamed, the rules are lost and must be linked again.

An ampersand (&) within the source code of a processing rule is dynamically replaced by the fully-qualified name of the field for which the rule is defined. Array indexes are not affected by the replacement; you must explicitly specify the index notation after the ampersand (&) as shown in the following example.

Examples:

```
IF &    = ' ' THEN REINPUT 'ENTER NAME' MARK *&   /* For a scalar field
IF &(1) = ' ' THEN MOVE 'X' TO &(*)               /* For an array field
```

The field name notation `&.field-name` within the source code of a processing rule allows you to have DDM-specific rules that cross-check the integrity of values between database fields, without having to explicitly qualify the fields with a view name. As `field-name` you specify the name of the database field as defined in the DDM, and at compilation time, Natural dynamically qualifies the field by replacing the ampersand (&) with the corresponding view name. This allows you to use the same processing rule for specific fields, regardless of which view the fields are taken from.

Function-Key-Related Processing Rules

Two types of function-key-related processing rules can be defined:

- Inline processing rules
- Predict free rules

Function-key-related processing rules can be used to assign activities to program sensitive function keys (PF keys) during map processing. For PF 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 Editing

Processing-rule editing is invoked by the field command `.P`, or by issuing the line command `..E` and then placing the function code `P` next to the field for which processing rule editing is to be performed. PF-key processing rule editing is invoked by the command `..P`.

A parameter can be used (`.P`*rr*) to indicate the rank (priority) of the processing rule to be defined/edited. 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

Note: When modifying or adding a processing rule, keep in mind that you must recatalog the map(s) that reference this rule so that the changes become effective.

The section below contains information on:

- Selecting a Rule for Editing
- Direct Commands
- Editor Commands for Positioning
- Line Commands for Processing Rules

Selecting a Rule for Editing

If you enter the field command `.P*` in a map field, you obtain a list of all processing rules defined for the field.

If you enter the line command `..P*` in any map line, you obtain a list of all processing rules defined for the PF keys used in this map.

On each list, the Predict rules are identified by their names, the inline rules by their first three source code lines. From each list you can select a rule for editing by entering its rank.

The screen for processing rule editing (with a processing rule example) is shown below:

```
Variables used in current map                                          MOD
MODTXT(A3)                                                               U
FVAR(A75/1:6)                                                            U
FTYP(A1/1:6)                                                             U
RULEMODE(A6)                                                             U
RULE-NAME(A32)                                                           D
FIELDAN(A5)                                                              D

Rule ________________________________ Field FULCB3.CBCOM
>                                      > + Rank 0     S 1   L 1    Struct Mode
ALL   ....+....10...+....20...+....30...+....40...+....50...+....60...+....70..
  0010 *
  0020 IF & EQ MASK('?')
  0030  REINPUT USING HELP
  0040 END-IF
  0050 *
  0060
  0070
  0080
  0090
  0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

During processing rule editing you can switch between split-screen and full-screen mode using PF9 or SPLIT/SPLIT E command. The upper half of the split screen displays the definitions of all map fields (except system variables). This display can be positioned by **split-screen commands**.

The source code used to define the processing rule is entered/edited in the same way as with the Natural program editor.

Direct Commands

While working in the processing-rule editor, processing rules can be edited by entering one of the following commands in the editor command line (>). In the table below, an underlined portion of a keyword represents an acceptable abbreviation. For further explanations of the syntax characters used in this table, refer to *System Command Syntax* in the *System Commands* documentation.

Command	Explanation
ADD[(*n*)]	Adds *n* blank lines to the source code. See the more detailed description of the ADD command in ***Editor Commands*** in the *Program Editor* documentation.
CHANGE '*string1*'*string2*'	Scans for the value entered as *string1* and replaces each such value found with the value entered as *string2*.
CHECK	Checks the rule.
CLEAR	Clears the editing area (including the line markers X and Y).
DX or DY or DX-Y	Deletes the X-marked line; or the Y-marked line; or the block of lines delimited by X and Y.
EX or EY or EX-Y	Deletes source lines from the top of the source area to, but not including, the X-marked line; or from the source line following the Y-marked line to the bottom of the source area; or all source lines in the source area excluding the block of lines delimited by X and Y.
EXIT	Terminates the rule editing function and returns to map editing.
.	
P	Positions forward to the next rule defined for the field.
P*	Selects a rule from the selection menu.
P*rr*	Selects the rule with rank *rr*.
P=*rr*	Changes the rank of a processing rule to rank *rr*.
POINT	Positions the line in which the line command .N was entered to the top of the current screen.

Command	Explanation
RESET	Deletes the current X and/or Y line markers and any marker previously set with the line command .N.
SAVE *name*	Saves a rule as copycode with the name *name*.
SCAN ['*scan-value*']	Scans for data in the source area. Entering SCAN without any parameter displays the **SCAN/REPLACE** menu. Entering SCAN '*scan-value*' results in a scan for *scan-value*.
SCAN = [+|-]	Scans for the next occurrence of the scan value. The direction of the scan operation is determined by the setting of the direction indicator. See the more detailed description of the SCAN commands in ***Editor Commands*** in the section *Program Editor*.
SHIFT [-|+*nn*]	Shifts each source line delimited by the X and Y markers to the left or right. *nn* represents the number of characters the source line is to be shifted. Comment lines are not shifted.
SHIFT --	Shifts each source line delimited by the X and Y markers to the leftmost position. Comment lines are not shifted.
SHIFT ++	Shifts each source line delimited by the X and Y markers to the rightmost position (maximum 99 positions). Comment lines are not shifted.
SPLIT [E]	Switches between split-screen mode and full-screen mode. See also SPLIT in ***Editor Commands for Positioning***.
TEST	Tests a map.
UNLINK	Unlinks an inline or Predict free rule from the field.

Note: To select a rule from all Predict free rules, enter a question mark (?) in the rule name field of the processing rule editing screen.

Editor Commands for Positioning

Editor commands for positioning are entered in the command line (>) of the rule editor. The commands available are listed in the following table; an underlined portion of a keyword represents an acceptable abbreviation.

Command	Explanation
+P	Positions forwards one page.
+	
-P	Positions backwards one page.
-	
+H	Positions forwards half a page.
-H	Positions backwards half a page.
TOP	Positions to top of rule.
-	

Command	Explanation
BOTTOM	Positions to bottom of rule.
++	
+*nnnn*	Positions forwards *nnnn* lines (maximum 4 digits).
-*nnnn*	Positions backwards *nnnn* lines (maximum 4 digits).
nnnn	Positions to line *nnnn*.
X	Positions to the line marked with X.
Y	Positions to the line marked with Y.
SPLIT[-|--|+|++]	Positions backwards (- or --) or forwards (+ or ++) in the split screen.

Line Commands for Processing Rules

In addition to the editor commands, the following line commands can be used when editing a processing rule:

Command	Explanation
.C(*nnnn*)	Copies the line in which the command was entered.
.CX(*nnnn*) or .CY(*nnnn*)	Copies the X-marked or the Y-marked line. See also the commands .X and .Y in the following section.
.CX-Y(*nnnn*)	Copies the block of lines delimited by the X and Y markers. If the direction indicator is a plus sign (+), the copied lines are placed after the line in which the command was entered. If the direction indicator is a minus sign (-), the copied lines are placed before the line in which the command was entered.
.D(*nnnn*)	Deletes line or lines. The default is 1 line.
.I(*n*)	Inserts *n* blank lines. With the next ENTER, lines that are left blank are eliminated again.
.I(*obj*,*ssss*,*nnnn*)	Inserts an object contained in the current library or in the steplib into the source. The *ssss* entry can be used to indicate the line number at which the insert operation is to begin. The *nnnn* entry can be used to indicate the number of lines to be inserted. For detailed information on the .I line commands, see ***Line Commands*** in the section *Editing a Map*.
.J	Joins the current line with the next line. If the resulting line length is greater than the length of the editor screen line, the line is marked with L and must then be separated again using the .S command (see below), before it can be modified.

Command	Explanation
.L	Ignores all modifications that have been made to the line since the last time ENTER was pressed.
.MX or .MY	Moves the X-marked or the Y-marked line. See also the commands .X and .Y below.
.MX-Y	Moves the block of lines delimited by the X and Y markers. If the direction indicator is set to a plus sign (+), the moved lines are placed after the line in which the command was entered. If the direction indicator is set to a minus sign (-), the moved lines are placed before the line in which the command was entered.
.P	Positions the line marked by this command to the top of the screen.
.S	Splits the line at the position marked by the cursor.
.W	Inserts *n* blank lines. With the next ENTER, lines that are left blank are eliminated again.
.X	Marks a line, or the beginning of a block of lines, to be processed.
.Y	Marks a line, or the end of a block of lines, to be processed. **Note:** If both the commands .X and .Y are applied to one line, it is treated as being marked with X and with Y; the line marker actually shown to reflect this status is a Z.

VIII Map Editor Tutorial

This tutorial provides a general introduction to using the Natural map editor where explanations are kept to a minimum. For a comprehensive description of all map editor functions, refer to the section *Map Editor*.

The layout of the example screens (24x80) provided in the tutorial and the behavior of Natural described here can differ from your results. For example, the command or message line may appear in a different screen position, or the execution of a Natural command can be protected by security control. The default settings in your environment depend on the system parameters set by your Natural system administrator.

Important: It is important that you work through the exercises in the sequence indicated below. Otherwise, you may not accomplish the results intended by the exercises.

Note: The map editor has been disabled in your environment by default. For more information, see *NaturalONE as the Default Development Environment* .

Opening the Map Editor	Invoking the map editor menu and initializing a map.
Creating, Positioning and Deleting Map Fields	Creating map fields and placing them in the required map position.
Testing and Saving a Map	Testing and saving a map as an object module.
Defining Processing Rules	Defining processing rules for a map field.
Naming Fields and Saving/Cataloging a Map	Giving names to map fields and saving/cataloging a map as object modules.
Defining Field Properties	Using extended field editing to define field properties.
Creating and Testing a Help Map	Creating and testing a help map for a map field.
Invoking a Map with INPUT USING MAP	Creating and executing a program that invokes a map with the `INPUT USING MAP` statement.
Creating a Map for WRITE and Copying Field Definitions	Creating a map by copying fields from another Natural object. Changing the map settings to use the `WRITE` statement.
Reusing the Layout of a Map	Creating a map from another map layout.

Invoking a Map with WRITE USING MAP Creating and executing a program that invokes a map with the `WRITE USING MAP` statement.

19 Opening the Map Editor

In this session, you will invoke the map editor, specify map settings and open the map editing screen.

To invoke the Map editor

1 In the Natural **Main Menu**, select **Development Functions** and press ENTER.

The **Development Functions** menu appears.

2 If you are working in reporting mode, change the programming mode to structured mode:

Enter an `S` in the first position of the **Mode** input field and press ENTER.

The **Mode** field now indicates `Structured`.

3 Enter an `E` (for **Edit Object**) in the **Code** field and an `M` (Map) in the **Type** field. Ignore the **Name** field.

4 Press ENTER.

The **Edit Map** menu appears:

```
14:08:08                ***** NATURAL MAP EDITOR *****              2007-12-14
User SAG                         - Edit Map -                  Library SAGTEST

                   Code    Function
                   ----    ---------------------------------
                     D     Field and Variable Definitions
                     E     Edit Map
                     I     Initialize new Map
                     H     Initialize a new Help Map
                     M     Maintenance of Profiles & Devices
                     S     Save Map
                     T     Test Map
                     W     Stow Map
                     ?     Help
                     .     Exit



             Code .. I     Name .. ________       Profile .. SYSPROF_

Command ===>









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Test  Edit
```

The **Edit Map** menu is the main menu of the map editor.

Tip: The map editor contains an extensive help system. Any time you require help, enter a question mark (?) in the field for which you want to obtain further information. This will invoke the online help for that field. If a field does not have an individual help assigned, a help menu will be displayed, from which you can select the required item of information.

5 Enter an `I` (for **Initialize new Map**) in the **Code** field and `MAP001` in the **Name** field.

6 Press ENTER.

The **Define Map Settings for MAP** screen appears:

```
14:10:19                Define Map Settings for MAP                  2007-12-14

 Delimiters            Format                          Context
 -----------------     ---------------------------     --------------------------
 Cls Att CD  Del       Page Size ...... 31             Device Check .... ________
  T   D      BLANK     Line Size ...... 79             WRITE Statement   _
  T   I      ?         Column Shift ... 0 (0/1)        INPUT Statement   X
  A   D      _         Layout ......... ________       Help  ____________________
  A   I      )          dynamic ....... N (Y/N)         as field default N (Y/N)
  A   N      ¬         Zero Print ..... N (Y/N)
  M   D      &         Case Default ... UC (UC/LC)
  M   I      :         Manual Skip .... N (Y/N)        Automatic Rule Rank  1
  O   D      +         Decimal Char ... .              Profile Name .... SYSPROF
  O   I      (         Standard Keys .. N (Y/N)
                       Justification .. L (L/R)        Filler Characters
                       Print Mode ..... __             ------------------------
                                                       Optional, Partial ....
                       Control Var .... ________       Required, Partial ....
                                                       Optional, Complete ...
                                                       Required, Complete ...

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                    Let
```

7 Move the cursor to the **Filler Characters** section of the screen. Type in an underscore (_) after each of the four options as shown below:

```
14:10:19               Define Map Settings for MAP                  2007-12-14

 Delimiters           Format                          Context
 -----------------    ---------------------------     --------------------------
 Cls Att CD  Del      Page Size ...... 31             Device Check .... ________
  T   D      BLANK    Line Size ...... 79             WRITE Statement   _
  T   I      ?        Column Shift ... 0 (0/1)        INPUT Statement   X
  A   D      _        Layout ......... ________       Help  ____________________
  A   I      )         dynamic ....... N (Y/N)         as field default N (Y/N)
  A   N      ¬        Zero Print ..... N (Y/N)
  M   D      &        Case Default ... UC (UC/LC)
  M   I      :        Manual Skip .... N (Y/N)        Automatic Rule Rank  1
  O   D      +        Decimal Char ... .              Profile Name .... SYSPROF
  O   I      (        Standard Keys .. N (Y/N)
                      Justification .. L (L/R)        Filler Characters
                      Print Mode ..... __             ------------------------
                                                      Optional, Partial .... _
                      Control Var .... ________       Required, Partial .... _
                                                      Optional, Complete ... _
                                                      Required, Complete ... _









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Let
```

This will cause any blank positions within an input field on the map to be filled with the underscore (_). You can then see the exact position and length of a field which makes entering input easier.

8 Ignore the other map settings and press ENTER *twice*.

The map editing screen appears:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

The screen appears in split-screen mode: the top half displays the delimiter characters, which are valid for the map to be created, and the bottom half is the editing area where you actually design a map.

You can now proceed with ***Creating, Positioning and Deleting Map Fields***.

20 Creating, Positioning and Deleting Map Fields

- Creating and Centering Fields 170
- Moving Fields 175
- Deleting Fields and Inserting Lines 190

In this session, you will design a map.

Creating and Centering Fields

To create text fields

1 In the first line of the editing area of MAP001, enter the line command `..F*`, and, in the second line, type in the text `PERSONNEL INFORMATION` as shown below:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
..F*
PERSONNEL INFORMATION

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

2 Press ENTER.

The screen now looks as follows:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
*******************************************************************************
PERSONNEL INFORMATION

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

3 Press PF9.

The map editing screen changes to full-screen mode: the delimiter characters are hidden.

4 In the bottom line, enter the line command `..F*`.

5 Press ENTER

The screen now looks as follows:

```
******************************************************************************
PERSONNEL INFORMATION
******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

To center all fields contained in a line

1 In the first three positions of the text, enter the line command `..C` as shown below:

```
*******************************************************************************
..CSONNEL INFORMATION
*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

2 Press ENTER.

The text is centered.

To create data fields

1 Enter the following as shown on the screen below:

```
********************************************************************************
(*DATX                        PERSONNEL INFORMATION
(*TIMX

PLEASE ENTER CITY::X(20)
PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
********************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

`*DATX` and `*TIMX` are Natural system variables, which display the current date and time respectively. The opening parenthesis (`(`) is the delimiter character for intensified (highlighted) output fields. The colon (`:`) is the delimiter character for intensified modifiable fields. The number of Xs indicates the length of the field.

2 Press ENTER.

The screen now looks as follows:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX




*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

Moving Fields

To move a single field or an entire line

1 In the editing area, enter the field command `.M` and move the cursor to the position indicated by `[]` as shown below. The `[]` sign is used to show the cursor position, you must not enter this sign.

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

.MEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[]

*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

2 Press ENTER.

The text field in which the command was entered is moved to the cursor position:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

       ENTER CITY::XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE









******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

3 Enter the line command `..M` as shown below and move the cursor to the position indicated by `[]` as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

..M    ENTER CITY::XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[]LEASE



*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

4 Press ENTER.

The line in which the command was entered is moved to the line after the one in which the cursor is positioned:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE
       ENTER CITY::XXXXXXXXXXXXXXXXXXXX




*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

To join lines

1 Enter the line command `..J` as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
..JASE
       ENER CITY::XXXXXXXXXXXXXXXXXXXX




*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

2 Press ENTER.

The line in which the command was entered and the line below it are joined:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

To move a block of fields or a block of lines

1 Type in additional text in the same order and position as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION OF TEXT IS
FOR FURTHER DEMONSTRATION
OF THE MOVE
COMMANDS









*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

2 Press ENTER.

3 Enter the field command `.M` twice and move the cursor to the position indicated by `[]` as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

.MIS PORTION OF TEXT IS
FOR FURTHER DEMONSTRATION
OF THE MOVE
.MMMANDS




                                        []






*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

The block of fields to be moved is delimited by the `.M` commands and the widths of the fields to which these commands apply. In this example, the block starts with the top left field `THIS` and ends at the last field `COMMANDS`. The widest field `COMMANDS` (which extends into two or three fields contained in the previous lines) determines which fields within the marked block are moved.

4 Press ENTER.

The marked block of fields is moved to the cursor position with the top left field being placed at the cursor:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

            OF TEXT IS
           DEMONSTRATION





                                     THIS PORTION
                                     FOR FURTHER
                                     OF THE MOVE
                                     COMMANDS









******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

5 Enter the field command `.M` twice as shown below and move the cursor to the position indicated by `[]` as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

             .M TEXT IS
            .MMONSTRATION





                                      THIS PORTION[]
                                     FOR FURTHER
                                     OF THE MOVE
                                     COMMANDS









*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

6 Press ENTER.

The block of fields delimited by the commands is moved to the cursor position:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                                    THIS PORTION OF TEXT IS
                                    FOR FURTHER DEMONSTRATION
                                    OF THE MOVE
                                    COMMANDS




*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

7 Enter the field command `.M` three times and move the cursor to the position indicated by `[]` as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

                            []





                                      .MIS PORTION OF TEXT IS
                                      FOR FURTHER .MMONSTRATION
                                     OF THE MOVE
                                      .MMMANDS


*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

8 Press ENTER.

The entire block of fields delimited by the commands is moved to the cursor position:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

                           THIS PORTION OF TEXT IS
                           FOR FURTHER DEMONSTRATION
                           OF THE MOVE
                           COMMANDS









******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

9 Enter the line command `..M` twice and move the cursor to the position indicated by `[]` as shown below:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

..M                          THIS PORTION OF TEXT IS
                            FOR FURTHER DEMONSTRATION
                            OF THE MOVE
..M                          COMMANDS


[]









******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

10 Press ENTER.

The block of lines delimited by the commands is inserted below the line in which the cursor is positioned. (The old block of lines is deleted from its previous location.)

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                           THIS PORTION OF TEXT IS
                           FOR FURTHER DEMONSTRATION
                           OF THE MOVE
                           COMMANDS
*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

Deleting Fields and Inserting Lines

To delete fields

1 In the editing area, enter the field command `.T` as shown below:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                          THIS PORTION .T TEXT IS
                          FOR FURTHER DEMONSTRATION
                          OF THE MOVE
                          COMMANDS



******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

2 Press ENTER.

The field in which the command was entered and the fields in the rest of the line are deleted:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                           THIS PORTION
                           FOR FURTHER DEMONSTRATION
                           OF THE MOVE
                           COMMANDS


******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

3 Enter the field command `.D` as shown below:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                           THIS PORTION
                           FOR .DRTHER DEMONSTRATION
                           OF THE MOVE
                           COMMANDS



******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

4 Press ENTER.

The field in which the command was entered is deleted:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                          THIS PORTION
                          FOR         DEMONSTRATION
                          OF THE MOVE
                          COMMANDS



*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

5 Enter the field command `.M` and move the cursor to the position indicated by `[]` as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                           THIS PORTION
                           FOR []      .MMONSTRATION
                           OF THE MOVE
                           COMMANDS



*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

6 Press ENTER.

The field in which the command was entered is moved to the cursor position:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




                           THIS PORTION
                           FOR DEMONSTRATION
                           OF THE MOVE
                           COMMANDS


******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

7 Enter the line command `..D` twice as shown below:

```
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX




..D                         THIS PORTION
                            FOR DEMONSTRATION
                            OF THE MOVE
..D                         COMMANDS



*******************************************************************************
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

8 Press ENTER.

The block of lines delimited by the commands is deleted:

```
******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

******************************************************************************

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

To insert lines

1 Enter the line command `..I4` as shown below:

```
*****************************************************************************
(XXXXXXXX                   PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX
..I4




















*****************************************************************************




001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

2 Press ENTER.

Four blank lines are inserted, and the bottom line with the asterisks is moved four lines down.

You can now proceed with ***Testing and Saving a Map***.

21 Testing and Saving a Map

In this session, you will test the layout of a map, leave the map editing area, and save a map as a source object.

To test a map and save it as a source object

1 In the editing area, press PF4.

The map is shown in the layout view in which it will appear on the screen when executing the program that references the map:

```
*******************************************************************************
 07-12-14                    PERSONNEL INFORMATION
 15:26:05

PLEASE ENTER NAME: ________________________________________
PLEASE ENTER CITY: ____________________

*******************************************************************************
```

2 Press PF3.

The test is terminated and the map editing area appears.

3 Press PF3.

The **Field and Variable Definitions - Summary** screen appears. This screen will be discussed in a later session of this tutorial.

4 Press ENTER.

The **Edit Map** menu appears with the **Name** field set to `MAP001`.

5 In the **Code** field, enter an `S` and press ENTER.

The map is stored as a source object with the name MAP001 in the current Natural library in the current system file.

You can now proceed with ***Defining Processing Rules***.

22 Defining Processing Rules

In this session, you will define processing rules for a map field.

To define processing rules

1 In the **Code** field of the **Edit Map** menu, enter an `E` and, in the **Name** field, enter `MAP001`.

The map editing screen appears in split-screen mode and map MAP001 is displayed in the editing area.

2 Enter the field command `.P` as shown below:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEAS E ENTER NAME:.PXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

3 Press ENTER.

The processing rule editor for the field in which the command was entered appears:

```
Variables used in current map                                          Mod
#001(A40)
#002(A20)




Rule ________________________________ Field #001
>                                     > + Rank 0    S      L 1     Struct Mode
ALL    ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
  0010
  0020
  0030
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

4 Type in the following processing rule:

```
Rule ________________________________ Field #001
>                                     > + Rank 0    S      L 1     Struct Mode
ALL    ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
  0010 *
  0020 IF & = ' ' REINPUT 'PLEASE TYPE IN A NAME'
  0030            MARK *&
  0040 END-IF
  0050 *
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

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

5 Press ENTER and then PF3.

The map editing screen appears.

6 Press PF4 to test the map.

The test screen appears.

7 Press ENTER to test the processing rule.

The processing rule is executed and the text entered in the rule appears:

```
********************************************************************************
 07-12-14                    PERSONNEL INFORMATION
 16:04:15

PLEASE ENTER NAME: ________________________________________
PLEASE ENTER CITY: ____________________

********************************************************************************
PLEASE TYPE IN A NAME
```

Note: The text `PLEASE TYPE IN A NAME` may not necessarily appear at the bottom of the screen (as shown above) but on another line, depending on the position of the message line as set by the Natural administrator.

8 In the first position of the input field next to `PLEASE ENTER NAME:`, enter any character and press ENTER.

The test is terminated and the map editing screen appears.

9 Enter the field command `.P` in the same position as before and press ENTER.

The processing rule for rank (priority) 0 of the field where the command was entered is displayed again.

10 Enter the command `P=5` as shown below:

```
Rule ________________________________ Field #001
> P=5                                  > + Rank 0    S 5    L 1     Struct Mode
ALL    ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
  0010 *
  0020 IF & = ' ' REINPUT 'PLEASE TYPE IN A NAME'
  0030            MARK *&
  0040 END-IF
  0050 *
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

11 Press ENTER.

The processing rule which was previously assigned to rank 0 is now assigned to rank 5 (processing rules are processed in ascending order of rank, starting with rank 0).

12 Enter the command `P0` as shown below:

```
Rule ________________________________ Field #001
> P0                                   > + Rank 5     S 5   L 1     Struct Mode
ALL   ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
  0010 *
  0020 IF & = ' ' REINPUT 'PLEASE TYPE IN A NAME'
  0030            MARK *&
  0040 END-IF
  0050 *
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

13 Press ENTER.

An empty processing rule editor screen is displayed, because there is no longer any processing rule assigned to rank 0.

```
Rule ________________________________ Field #001
>                                     > + Rank 0    S      L 1     Struct Mode
ALL    ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
  0010
  0020
  0030
  0040
  0050
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

14 Type in the following processing rule:

```
Rule ________________________________ Field #001
>                                     > + Rank 0     S 0   L 1     Struct Mode
ALL   ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
  0010 *
  0020 IF & = MASK ('.') STOP
  0030 END-IF
  0040 *
  0050
  0060
  0070
  0080
  0090
  0100
  0110
  0120
  0130
  0140
  0150
  0160
  0170
  0180
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

15 Press ENTER and then PF3.

The map editing screen appears.

You can now proceed with ***Naming Fields and Saving/Cataloging a Map***.

23 Naming Fields and Saving/Cataloging a Map

In this session, you will give names to the user-defined fields in a map and save and catalog a map.

To name fields and save/catalog a map

1 On the map editing screen of MAP001, press PF3.

The Field and Variable Definitions - Summary screen appears:

```
16:10:49          Field and Variable Definitions - Summary         2007-12-14

Cmd Name (truncated)                              Mod Format       Ar Ru Lin Col
___ *DATX_________________________________________ S   D                   2   2
___ *TIMX_________________________________________ S   T                   3   2
___ #001__________________________________________     A40             2   5  20
___ #002__________________________________________     A20                 6  20

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit              --                Parm  Local       Let
```

The fields contained in the map are listed in the order in which they appear on the map. The two user-defined fields are assigned dummy names, which are preceded by a number sign (#). These dummy names must be replaced to catalog the map.

2 Replace the dummy names as shown below:

```
16:10:49          Field and Variable Definitions - Summary          2007-12-14

Cmd Name (truncated)                              Mod Format       Ar Ru Lin Col
___ *DATX_____________________________________ S   D                    2   2
___ *TIMX_____________________________________ S   T                    3   2
___ #NAME_____________________________________     A40              2   5  20
___ #CITY_____________________________________     A20                  6  20

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit              --                Parm  Local       Let
```

3 Press ENTER *twice*.

The **Edit Map** menu appears with the **Name** field set to `MAP001`.

4 In the **Code** field, enter a `W` (for **Stow Map**) and press ENTER.

The **Stow Map** function executes the `STOW` command, which checks the syntax of MAP001 and stores the map definition as a source object *and* a cataloged object in the current Natural library in the current system file.

You can now proceed with *Defining Field Properties*.

24 Defining Field Properties

In this session, you will define the properties for a field.

To define the properties for a field

1 In the **Code** field of the **Edit Map** menu, enter an `E` and, in the **Name** field, enter `MAP001` (if it is not already entered).

The map editing screen appears and map MAP001 is displayed in the editing area.

2 Type in additional text as shown below:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

TYPE IN?. TO STOP OR?? FOR HELP.









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

The question mark (?) shown in boldface is the delimiter character for intensified text fields.

3 Enter the field command `.E` as shown below:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
*******************************************************************************
(XXXXXXXX                   PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME:.EXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

TYPE IN?. TO STOP OR?? FOR HELP.









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

4 Press ENTER.

The extended field editing area for the field in which the command was entered appears:

```
Fld #NAME                                                        Fmt A40
--------------------------------------------------------------------------------
AD= MIT'_'____     ZP=         SG=        HE= ______________________   Rls 2
AL= _____          CD= __      CV= ________________________________   Mod User
PM= __  DF=                    DY= ______________________________
EM= ___________________________ SB= ________________________________

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
********************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME:.EXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

TYPE IN?. TO STOP OR?? FOR HELP.









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  <---  --->  --    -     +           <     >     Let
```

5 In the **Fmt** field, replace the current value by `A20`, and, in the **HE=** field, enter `'HELP001'` (within apostrophes) as shown below:

```
Fld #NAME                                                        Fmt A20
--------------------------------------------------------------------------------
AD= MIT'_'____      ZP=         SG=        HE= 'HELP001'_____________   Rls 2
AL= _____           CD= __      CV= ________________________________   Mod User
PM= __  DF=                     DY= ______________________________
EM= ___________________________ SB= ________________________________

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
********************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME:.EXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

TYPE IN?. TO STOP OR?? FOR HELP.









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  <---  --->  --    -     +           <     >     Let
```

6 Press ENTER.

The extended field editing area disappears. HELP001 (which is yet to be created) is assigned as the help routine/help map to the field and the field length is reduced to 20:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .     A  D    _       A  I    )
.                                       .     A  N    ¬       M  D    &
.                                       .     M  I    :       O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
*******************************************************************************
(XXXXXXXX                    PERSONNEL INFORMATION
(XXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

TYPE IN?. TO STOP OR?? FOR HELP.









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

7 Press PF3 *twice*.

The **Edit Map** menu appears.

8 STOW MAP001 by using the appropriate menu function.

You can now proceed with *Creating and Testing a Help Map*.

25 Creating and Testing a Help Map

In this session, you will create and test a help routine/help map for a map field.

To create a help map

1 In the **Edit Map** menu, enter an `H` in the **Code** field and `HELP001` in the **Name** field.

2 Press ENTER.

The **Define Map Settings for HELPMAP** screen appears.

3 In the **Page Size** field, enter `15`, and, in the **Line Size** field, enter `25` as shown below:

```
11:36:53               Define Map Settings for HELPMAP              2007-12-14

 Delimiters           Format                          Context
 -----------------    ----------------------------    --------------------------
 Cls Att CD  Del      Page Size ...... 15             Device Check .... ________
  T   D      BLANK    Line Size ...... 25             WRITE Statement   _
  T   I      ?        Column Shift ... 0 (0/1)        INPUT Statement   X
  A   D      _        Layout ......... ________
  A   I      )         dynamic ....... N (Y/N)
  A   N      ¬        Zero Print ..... N (Y/N)        Position Line    Col
  M   D      &        Case Default ... UC (UC/LC)
  M   I      :        Manual Skip .... N (Y/N)        Automatic Rule Rank  1
  O   D      +        Decimal Char ... .              Profile Name .... SYSPROF
  O   I      (        Standard Keys .. N (Y/N)
                      Justification .. L (L/R)        Filler Characters
                      Print Mode ..... __             ------------------------
                                                      Optional, Partial ....
                      Control Var .... ________       Required, Partial ....
                                                      Optional, Complete ...
                                                      Required, Complete ...









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Let
```

4 Press ENTER *twice*.

The map editing screen appears.

5 Press PF9.

The screen appears in full-screen mode:

```
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
                          ..................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
..........................................................................
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Split <     >     Let
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

The portion of the screen not to be used is filled with periods.

6 Enter text as shown below:

```
                            ..................................................
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.               ..................................................
                            ..................................................
For a list of employees     ..................................................
of a certain name who       ..................................................
live in a certain city,     ..................................................
type in a name in the       ..................................................
first field and a city      ..................................................
in the second field         ..................................................
and press ENTER.            ..................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
.......................................................................................
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Split <     >     Let
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

7 Press PF3.

The **Edit Map** menu appears with the **Name** field set to `HELP001`.

8 `STOW` help map HELP001 by using the appropriate menu function.

9 In the **Code** field, enter a `T`, and, in the **Name** field, enter `MAP001`.

10 Press ENTER.

The test screen for MAP001 appears.

11 In the first position of the input field next to `PLEASE ENTER NAME:`, enter a question mark (?) and press ENTER.

Help map HELP001 appears:

```
*******************************************************************************
 07-12-14                    PERSONNEL INFORMATION
 16:58:37

PLEASE ENTER NAME: ?____________________
PLEASE ENTER CITY: +---------------------------+
                   |                           |
TYPE IN . TO STOP  | 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.             |
                   |                           |
                   | For a list of employees   |
                   | of a certain name who     |
                   | live in a certain city,   |
                   | type in a name in the     |
                   | first field and a city    |
                   | in the second field       |
                   | and press ENTER.          |
                   |                           |
                   +---------------------------+
*******************************************************************************
```

12 Press ENTER *twice*.

The processing rule for the first field (#NAME) is tested and the following message is displayed: PLEASE TYPE IN A NAME.

13 In the first position of the first field, enter any character and press ENTER.

The test is terminated and the **Edit Map** menu appears.

You can now proceed with *Invoking a Map with INPUT USING MAP*.

26 Invoking a Map with INPUT USING MAP

In this session, you will create and execute the example program PROG001 to test the effect of the `INPUT USING MAP` statement, which invokes map MAP001.

PROG001 is provided in the Natural system library SYSEXPG; ask your Natural system administrator for details.

To create and execute PROG001

1 If you have access to a copy of PROG001, in the command line of the **Edit Map** menu, enter the following:

```
EDIT PROG001
```

The program editor is invoked and the source code of PROG001 is displayed in the editing area. Make sure that the program is identical to the one shown below.

Or:

If you do not have access to a copy of PROG001, in the command line of the **Edit Map** menu, enter the following:

```
EDIT PROGRAM
```

The program editor is invoked. If necessary, clear the editing area by entering the command `CLEAR` at the program editor's command prompt (>). Then type in the following program:

PROG001:

```
** Example 'PROG001': Example program for the Map Tutorial
************************************************************************
 DEFINE DATA LOCAL
 01 PERS-VIEW VIEW OF EMPLOYEES
   02 NAME
   02 FIRST-NAME
   02 CITY
 *
 01 #NAME (A20)
 01 #CITY (A20)
 END-DEFINE
 *
 REPEAT
   /*
   INPUT USING MAP 'MAP001'
   /*
   IF #CITY NE ' ' AND #NAME NE ' '
     FIND PERS-VIEW WITH NAME = #NAME AND CITY = #CITY
       IF NO RECORDS FOUND
         REINPUT 'NO ONE BY THIS NAME LIVING IN THIS CITY.'
                 MARK *#CITY
       END-NOREC
       /*
       DISPLAY NOTITLE NAME FIRST-NAME CITY
       /*
     END-FIND
   ELSE
     IF #NAME NE ' '
       FIND PERS-VIEW WITH NAME = #NAME
         IF NO RECORDS FOUND
           REINPUT 'PLEASE TRY ANOTHER NAME.'
         END-NOREC
         /*
         DISPLAY NOTITLE NAME FIRST-NAME CITY
         /*
       END-FIND
     END-IF
   END-IF
   /*
 END-REPEAT
 END
```

2 If no program name is displayed in the top line of the editor, at the program editor's command prompt, enter the command `STOW PROG001`.

Or:

If the program name PROG001 is displayed in the top line of the editor, at the program editor's command prompt, enter the command `STOW`.

If required, correct any syntax errors and repeat the STOW.

After the STOW completed successfully, the program is stored as a source object and a cataloged object in the current Natural library in the current system file.

3 At the program editor's command prompt, enter the following:

```
RUN
```

PROG001 is executed and map MAP001 appears.

To check whether MAP001 works as intended

1 Press ENTER without typing in anything.

The following message is displayed: PLEASE TYPE IN A NAME.

2 In the first input field, enter a question mark (?) and press ENTER.

Help map HELP001 appears.

3 Press ENTER.

The help map disappears.

4 In the first input field, enter the name MCKENNA and press ENTER.

The following message is displayed: PLEASE TRY ANOTHER NAME.

5 Replace MCKENNA by JONES and press ENTER.

The program produces the following list:

```
        NAME               FIRST-NAME              CITY
-------------------- -------------------- --------------------

JONES                VIRGINIA             TULSA
JONES                MARSHA               MOBILE
JONES                ROBERT               MILWAUKEE
JONES                LILLY                BEVERLEY HILLS
JONES                EDWARD               CAMDEN
JONES                MARTHA               KALAMAZOO
JONES                LAUREL               BALTIMORE
JONES                KEVIN                DERBY
JONES                GREGORY              NOTTINGHAM
```

6 Press ENTER.

MAP001 appears.

7 In the first input field of the map, enter the name `JONES` and, in the second input field, enter the name `DUNFERMLINE`.

8 Press ENTER.

The following message is displayed: `NO ONE BY THIS NAME LIVING IN THIS CITY`.

9 In the first input field of the map, enter the name `JONES` and, in the second input field, enter the name `TULSA`.

10 Press ENTER.

The program produces the following list:

```
        NAME               FIRST-NAME              CITY
-------------------- -------------------- --------------------

JONES                VIRGINIA             TULSA
```

11 Press ENTER.

MAP001 appears.

12 In the first input field, enter a period (.) and press ENTER.

The editing area of the program editor appears with the source code of PROG001.

You can now proceed with *Creating a Map for WRITE and Copying Field Definitions*.

27 Creating a Map for WRITE and Copying Field Definitions

In this session, you will create a map by copying the field definitions from a DDM (data definition module). In addition, you will change the map setting so that you can use the `WRITE USING MAP` statement.

To create a map from DDM fields

1 At the program editor's command prompt, enter the following:

```
EDIT MAP
```

The **Edit Map** menu appears.

2 Execute the **Initialize new Map** function to create a map with the name `MAP002`.

The **Define Map Settings For MAP** screen appears.

3 Change **Page Size** to `60`. Mark **WRITE Statement** by typing in an `X` in the input field next to it. Unmark **INPUT Statement** by removing the `X` next to it with SPACEBAR.

4 Press ENTER.

The **WRITE Statement** option is marked, which means that this map can only be invoked from a program with a `WRITE USING MAP` statement.

5 Press ENTER *twice*.

The map editing screen appears.

6 Next to the **Ob** input fields at the top of the screen, enter the following:

```
V EMPLOYEES
```

7 Press ENTER.

The fields definitions of the DDM EMPLOYEES are listed:

```
Ob V EMPLOYEES                          Ob D CLS ATT  DEL    CLS ATT  DEL
1 PERSONNEL-ID              A8          .     T  D    Blnk    T  I    ?
. FULL-NAME                 *G1         .     A  D    _       A  I    )
2 FIRST-NAME                A20         .     A  N    ¬       M  D    &
3 MIDDLE-I                  A1          .     M  I    :       O  D    +
4 NAME                      A20         .     O  I    (
5 MIDDLE-NAME               A20         .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

8 In the editing area, enter the following:

```
Ob V EMPLOYEES                          Ob D CLS ATT  DEL    CLS ATT  DEL
1 PERSONNEL-ID                    A8    .     T  D    Blnk    T  I    ?
. FULL-NAME                       *G1   .
2 FIRST-NAME                      A20   .
3 MIDDLE-I                        A1    .                     O  D    +
4 NAME                            A20   .     O  I    (
5 MIDDLE-NAME                     A20   .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(4
(2

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

This defines two fields for the map, whose definitions are to be copied from the DDM: the sequential number 4 entered will copy the definition of the corresponding field `NAME`, and number 2 will copy the definition of the corresponding field `FIRST-NAME`.

9 Press ENTER.

The field definitions in the DDM are copied into the map:

```
Ob V EMPLOYEES                          Ob D CLS ATT  DEL    CLS ATT  DEL
1 PERSONNEL-ID                    A8    .     T  D    Blnk    T  I    ?
. FULL-NAME                       *G1   .
2 FIRST-NAME                      A20   .
3 MIDDLE-I                        A1    .                     O  D    +
4 NAME                            A20   .     O  I    (
5 MIDDLE-NAME                     A20   .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(XXXXXXXXXXXXXXXXXXXX
(XXXXXXXXXXXXXXXXXXXX




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

10 Use the field command `.M` to move the field positioned in the second line of the editing area to the position shown below:

```
Ob V EMPLOYEES                          Ob D CLS ATT  DEL    CLS ATT  DEL
1 PERSONNEL-ID              A8          .     T  D    Blnk    T  I    ?
. FULL-NAME                 *G1         .
2 FIRST-NAME                A20         .
3 MIDDLE-I                  A1          .                     O  D    +
4 NAME                      A20         .     O  I    (
5 MIDDLE-NAME               A20         .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(XXXXXXXXXXXXXXXXXXXX(XXXXXXXXXXXXXXXXXXXX)
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

11 In the **Ob** input fields, replace the `V` by a plus sign (+) and press ENTER.

The next page of field definitions is displayed.

12 Repeat the previous step until the field `CITY` appears in the list. Replace the `V` by a minus sign (-) if you need to scroll up one page in the list.

13 Enter `CITY:(2` as shown below:

```
Ob V EMPLOYEES                          Ob D CLS ATT  DEL    CLS ATT  DEL
. FULL-ADDRESS              *G1         .     T  D    Blnk    T  I    ?
1 ADDRESS-LINE              A20         .     A  D    _       A  I    )
2 CITY                      A20         .     A  N    ¬       M  D    &
3 ZIP                       A10         .     M  I    :       O  D    +
4 POST-CODE                 A10         .     O  I    (
5 COUNTRY                   A3          .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(XXXXXXXXXXXXXXXXXXX(XXXXXXXXXXXXXXXXXXX CITY:(2

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

14 Press ENTER.

The screen now look as follows:

```
Ob V EMPLOYEES                          Ob D CLS ATT  DEL    CLS ATT  DEL
. FULL-ADDRESS              *G1         .     T  D    Blnk    T  I    ?
1 ADDRESS-LINE              A20         .     A  D    _       A  I    )
2 CITY                      A20         .     A  N    ¬       M  D    &
3 ZIP                       A10         .     M  I    :       O  D    +
4 POST-CODE                 A10         .     O  I    (
5 COUNTRY                   A3          .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(XXXXXXXXXXXXXXXXXXXX(XXXXXXXXXXXXXXXXXXXX CITY:(XXXXXXXXXXXXXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

15 Press PF3.

The **Edit Map** menu appears.

16 STOW map MAP002 by using the appropriate menu function.

You can now proceed with *Reusing the Layout of a Map*.

28 Reusing the Layout of a Map

In this session, you will create a map by using the layout of another map.

To create a map from a map layout

1 From the **Edit Map** menu, execute the **Initialize new Map** function to create a map with the name `MAP003`.

2 On the **Define Map Settings for MAP** screen, change **Page Size** to `60`, mark **WRITE Statement**, unmark **INPUT Statement**, and enter `MAP002` next to **Layout**.

3 Press ENTER.

The map settings now look as follows:

```
16:57:39               Define Map Settings for MAP                  2007-12-14

 Delimiters           Format                          Context
 -----------------    ---------------------------     --------------------------
 Cls Att CD  Del      Page Size ...... 60             Device Check .... ________
  T   D      BLANK    Line Size ...... 79             WRITE Statement   X
  T   I      ?        Column Shift ... 0 (0/1)        INPUT Statement   _
                      Layout ......... MAP002__       Help  ____________________
                       dynamic ....... N (Y/N)         as field default N (Y/N)
                      Zero Print ..... N (Y/N)
                      Case Default ... UC (UC/LC)
                      Manual Skip .... N (Y/N)        Automatic Rule Rank  1
  O   D      +        Decimal Char ... .              Profile Name .... SYSPROF
  O   I      (        Standard Keys .. N (Y/N)
                      Justification .. L (L/R)        Filler Characters
                      Print Mode ..... __             ------------------------
                                                      Optional, Partial ....
                      Control Var .... ________       Required, Partial ....
                                                      Optional, Complete ...
                                                       Required, Complete ...









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Let
```

4 Press ENTER.

The map editing screen appears with the layout of map MAP002.

5 Delete `CITY:(XXXXXXXXXXXXXXXXXXXX` by using the field command `.T`.

6 Move the second of the remaining output fields to the position shown below by using the field command `.M`.

The screen then looks as follows:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .
.                                       .
.                                       .                     O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(XXXXXXXXXXXXXXXXXXXX             (XXXXXXXXXXXXXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

7 Insert the text `FIRST NAME:` into the line as shown below:

```
Ob _                                    Ob D CLS ATT  DEL    CLS ATT  DEL
.                                       .     T  D    Blnk    T  I    ?
.                                       .
.                                       .
.                                       .                     O  D    +
.                                       .     O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
NAME:(XXXXXXXXXXXXXXXXXXXX FIRST NAME: (XXXXXXXXXXXXXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

8 Press PF3.

The **Edit Map** menu appears.

9 STOW map MAP003 by using the appropriate menu function.

You can now proceed with *Invoking a Map with WRITE USING MAP*.

29 Invoking a Map with WRITE USING MAP

In this session, you will create and execute the example program PROG002 to test the effects of the `WRITE USING MAP` statements, which invoke the maps MAP002 and MAP003.

To create PROG002

1 In the command line of the **Edit Map** menu, enter the following:

```
EDIT PROG002
```

The program editor is invoked and the source code of PROG002 is displayed in the editing area.

2 At the program editor's command prompt, enter the following:

```
SAVE PROG002
```

A copy of program PROG001 is saved as a source object under the name PROG002 in the current Natural library in the current system file.

3 At the program editor's command prompt, enter the following:

```
READ PROG002
```

The source code of program PROG002 is displayed in the editing area of the program editor.

4 Replace the `DISPLAY` statements with the source code lines shown in boldface below:

PROG002:

```
** Example 'PROG002': Example program for the Map Tutorial
************************************************************************
DEFINE DATA LOCAL
01 PERS-VIEW VIEW OF EMPLOYEES
  02 NAME
  02 FIRST-NAME
  02 CITY
*
01 #NAME (A20)
01 #CITY (A20)
END-DEFINE
*
REPEAT
  /*
  INPUT USING MAP 'MAP001'
  /*
  IF #CITY NE ' ' AND #NAME NE ' '
    FIND PERS-VIEW WITH NAME = #NAME AND CITY = #CITY
      IF NO RECORDS FOUND
        REINPUT 'NO ONE BY THIS NAME LIVING IN THIS CITY.'
                MARK *#CITY
      END-NOREC
      /*
      AT START OF DATA
        WRITE 'THE FOLLOWING EMPLOYEES LIVE IN' CITY
      END-START
      WRITE USING MAP 'MAP003'
      /*
    END-FIND
  ELSE
    IF #NAME NE ' '
      FIND PERS-VIEW WITH NAME = #NAME
        IF NO RECORDS FOUND
          REINPUT 'PLEASE TRY ANOTHER NAME.'
        END-NOREC
        /*
        WRITE USING MAP 'MAP002'
        /*
      END-FIND
    END-IF
  END-IF
  /*
END-REPEAT
END
```

5 When you have made all changes, at the program editor's prompt, enter the following:

```
STOW
```

The source of PROG002 is updated and a cataloged object is created and stored in the current Natural library in the current system file.

To execute PROG002

1 At the program editor's command prompt, enter the following:

```
PROG002
```

2 The program starts to execute and MAP001 appears.

3 Enter the name `JONES` and leave the second input field empty.

4 Press ENTER.

The list produced by the program now uses MAP002:

```
Page      1                                                 07-12-14  17:12:41

NAME: JONES                VIRGINIA             CITY: TULSA
NAME: JONES                MARSHA               CITY: MOBILE
NAME: JONES                ROBERT               CITY: MILWAUKEE
NAME: JONES                LILLY                CITY: BEVERLEY HILLS
NAME: JONES                EDWARD               CITY: CAMDEN
NAME: JONES                MARTHA               CITY: KALAMAZOO
NAME: JONES                LAUREL               CITY: BALTIMORE
NAME: JONES                KEVIN                CITY: DERBY
NAME: JONES                GREGORY              CITY: NOTTINGHAM
```

5 Press ENTER.

MAP001 appears.

6 Leave the name `JONES` and enter the city `DERBY`.

7 Press ENTER.

The list produced by the program now uses MAP003:

```
Page      2                                                  07-12-14  17:15:18

THE FOLLOWING EMPLOYEES LIVE IN DERBY
NAME: JONES               FIRST NAME:  KEVIN
```

8 Press ENTER.

MAP001 appears.

9 In the first position of the NAME input field, enter a period (.).

The **Edit Map** menu appears.

You have successfully completed this tutorial.

IX SYSDDM Utility

The SYSDDM utility is used to create, maintain and delete Natural data definition modules (DDMs).

This documentation is organized in the following parts:

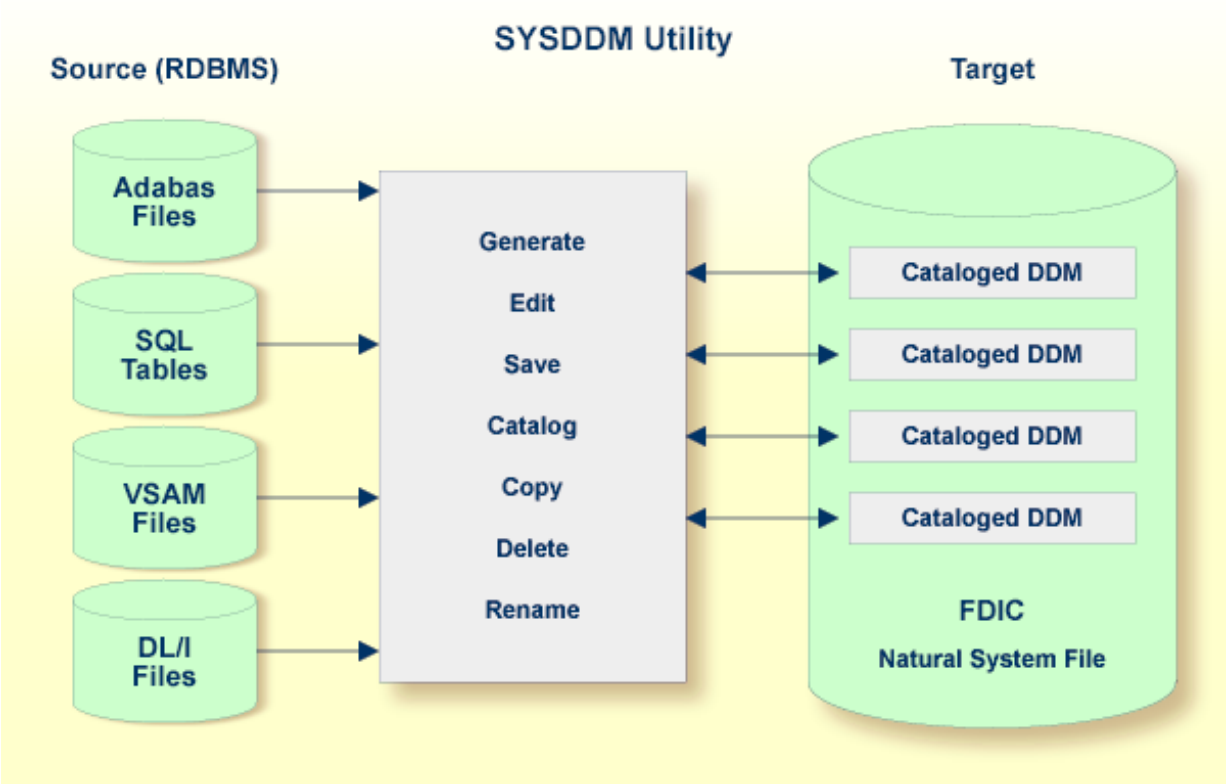
Principles of Operation
Invoking and Terminating SYSDDM
Using SYSDDM Maintenance and Service Functions
Creating DDMs
Invoking and Terminating the DDM Editor
Using the DDM Editor
Cataloging a DDM
Listing DDMs
Maintaining DDMs in Different Environments

30 Principles of Operation

- Storing DDMs 254
- Restrictions of Use 255

The SYSDDM utility is used to create a Natural DDM from a database file or from another DDM. A Natural object (for example, a program or a data area) can only access a database file if a corresponding DDM has been created for this file and saved as a cataloged object. For further details on Natural DDMs, see *Data Definition Modules - DDMs* in the *Programming Guide*.

The following graphic illustrates the main features and basic principles of operation when processing DDMs with the SYSDDM utility:



Storing DDMs

DDMs are stored in the Natural system file FDIC. The FDIC system file collects all DDMs available in your Natural system environment. You can only store and access DDMs in the FDIC system file.

DDMs are not restricted to files stored in an Adabas database. Some options described in the *SYSDDM Utility* documentation only apply to Adabas and can be ignored if a different database management system is used.

Restrictions of Use

The section below describes the restrictions that can apply to using SYSDDM utility functions under:

- Predict
- Natural Security

Predict

To guarantee data integrity for DDMs defined in Predict, the Predict administrator can restrict the use of SYSDDM utility functions for editing, copying, creating, deleting, renaming and cataloging, for which equivalent functions are provided by Predict.

In principle, we strongly recommend that you do not use *any* SYSDDM utility functions that can alternatively be performed by Predict.

For further information, contact your Predict system administrator.

Natural Security

Access to DDMs can be restricted when Natural Security has been installed. Within the DDM security profile, there may be a definition of whether a DDM may be modified only by specific users (DDM modifiers) or the owners of the security profile.

For further information, see *Protecting DDMs On Mainframes* in the *Natural Security* documentation.

31 Invoking and Terminating SYSDDM

This section provides instructions for invoking and terminating the utility SYSDDM.

To invoke the SYSDDM utility

- Enter the following Natural system command:

  ```
  SYSDDM
  ```

  Or:

  1. From the Natural main menu, choose **Maintenance and Transfer Utilities** to display the **Maintenance and Transfer Utilities** menu.
  2. From the **Maintenance and Transfer Utilities** menu, choose **Maintain DDMs**.

  The menu of the SYSDDM utility appears as shown in the example below:

```
10:01:44              ***** NATURAL SYSDDM UTILITY *****             2019-07-24
User SAG                            - Menu -                       FDIC (10,460)
                                                                 Work area empty
             DDM Maintenance         List/Copy Services

             E  Edit DDM             L  List DDMs
             R  Read DDM             X  List DDMs with Additional Information
             C  Catalog DDM          S  Show Defined DBIDs and Used FNRs
             U  Delete DDM           M  Copy DDM to Another FDIC File
             ?  Help
             .  Exit                 Other Services

                                     G  Generate DDM from Adabas FDT
                                      B  SQL Services (NDB)
                                     D  DL/I Services
                                     Z  SQL Services (NSB)
 Code ...... _                                FDIC Type ........ A
 DDM Name .. ________________________________  DDM Type ......... _
 FNR ....... 0     DBID .. 0                  Adabas Password ..
 Replace ... N                                DBID Type ........ 7

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                    Canc
```

The fields and functions provided on the SYSDDM utility menu are explained in the section *Using SYSDDM Maintenance and Service Functions*.

To terminate the SYSDDM utility

- In the SYSDDM utility menu, press PF3 (Exit).

  Or:

  In the command line, enter a period (.).

32 Using SYSDDM Maintenance and Service Functions

- Help on Functions 260
- Performing a Function 260
- Description of Functions 261
- DDM Specification 264

The functions provided in the SYSDDM utility menu are used to create, display, edit, rename or delete a DDM.

This section describes the fields and functions available in the SYSDDM utility menu and provides information on maintaining DDMs in different environments.

Help on Functions

This section provides instructions for obtaining information on the fields and functions provided in the SYSDDM utility menu.

To display SYSDDM help information

1 On the SYSDDM utility screen, press PF1.

Or:

In the command line or in any input field, enter a question mark (?).

The **Help Main Menu** of the SYSDDM utility appears.

2 Position the cursor next to the greater than (>) sign of the help topic required and press PF1.

The SYSDDM help screen appears for the topic selected.

- To scroll forward, press ENTER.
- To return to the **Help Main Menu**, press PF3.
- To terminate the help function and return to the SYSDDM utility menu, press PF12.

3 To exit the **Help Main Menu**, enter a period (.) or press PF3.

The SYSDDM utility menu appears.

Performing a Function

This section provides instructions for performing a function from the SYSDDM utility menu.

To perform a function

- In the SYSDDM utility menu:

- From the section **DDM Maintenance**, **List/Copy Services** or **Other Services** choose the one-digit code listed next to the function required and enter it in the **Code** field.

  For example: to edit a DDM, enter the function code `E`.

  The functions are explained in *Description of Functions*.

- In the input fields next to **Code**, enter a valid value to specify the DDM(s) to be processed as described in *DDM Specification*.

Description of Functions

This section describes the functions available on the SYSDDM utility menu and lists their corresponding function codes:

Function Code	Field	Function
E	**Edit DDM**	Invokes the DDM editor and reads a DDM source from the FDIC system file into the source area.
R	**Read DDM**	Reads a DDM source from the FDIC system file into the source area but does *not* invoke the DDM editor.
C	**Catalog DDM**	Saves the DDM source currently contained in the source area as a cataloged object in the current FDIC system file. For the naming conventions that apply when cataloging a DDM, refer to *Object Naming Conventions* in the *Using Natural* documentation. If the source area is empty, use the function **Generate DDM from Adabas FDT** or **Edit DDM** to load a source into the source area. For a DDM from VSAM (**DDM Type** set to `V`; see also **DDM Type** in *DDM Specification*), SYSDDM prompts you for additional information. For details, see *Natural File Access* in the *Database Management System Interfaces* documentation.
U	**Delete DDM**	Deletes one or more cataloged DDM from the FDIC system file. The contents of the source area are not affected by the deletion. When you delete a DDM with SYSDDM, the corresponding Natural Security file profile is deleted too.
L	**List DDMs**	Displays a single DDM source (DDM editor *not* invoked) or a list of DDMs stored in the specified FDIC system file similar to the Natural system command `LIST DDM` described in the *System Commands* documentation. However, unlike the system command `LIST DDM`, the **List DDMs** function additionally displays

Function Code	Field	Function
		all DDMs for which no read or update access right is defined in Natural Security (if installed). For details, see *Protecting DDMs On z/OS* in the *Natural Security* documentation. From the list of DDMs displayed on the **LIST DDMs** screen, you can select a DDM for further processing by entering the line command that corresponds to the action required in the **Cmd** column. For information on possible commands, enter a question mark (?) in the **Cmd** column.
X	**List DDMs with Additional Information**	Displays a list of DDMs stored in the specified FDIC system file. Displays a list of DDMs stored in the specified FDIC system file. When you enter a DDM name, the system may take a few moments to retrieve the complete list. The list contains the following fields of information: **DBID** — The database ID: see **DBID** in *DDM Specification*. **FNR** — The file number: see **FNR** in *DDM Specification*. **DDM Typ** — The DDM type: see **DDM Type** in *DDM Specification*. **Bytes** — The size of the DDM in bytes. **Secur. Type** — Only applies if Natural Security is installed. The security type: `Public`, `Private`, `Access` or `Undef` (undefined). See also the status of a DDM in the section *Components of a File Profile* in the *Natural Security* documentation). **File Type** — Only applies to a DDM created from VSAM, or a PersonalDB file from Super Natural. The VSAM file type: `Log.View` (logical view), `Phy.File` (physical file) or `Log.File` (logical file). Super Natural: `Userfile` indicates that the DDM was created in Super Natural. **File Name** — Only applies to DDMs from VSAM files. The name of a logical or physical file.

Function Code	Field	Function
		Remark — A remark such as `SupNat` (for a `Userfile` from Super Natural) or the VSAM file organization (`KSDS`, `RRDS`, `ESDS` or `VRDS`). From the list displayed on the screen **List DDMs with Additional Information**, you can select a DDM for further processing by entering the line command that corresponds to the action required in the **Cmd** column. For information on possible commands, enter a question mark (?) in the **Cmd** column. The **Refresh List (Refr)** option is available on the **PF5** key. Use this option to retrieve the latest list of DDMs in real time. For example, if a DDM has been updated or deleted by another user or process, press **PF5** to refresh the list and display the current DDM information.
S	**Show Defined DBIDs and Used FNRs**	See ***Show Defined DBIDs and Used FNRs***.
M	**Copy DDM to Another FDIC File**	Copies one or more DDMs from one FDIC system file to another. This can be required, for example, when a Natural application is transferred from test to production status. This function invokes the **Copy DDMs** window of the Natural utility SYSMAIN where you can specify source and target environment of the DDM(s) to be copied; see also the section *Processing DDMs* in the *SYSMAIN Utility* documentation.
G	**Generate DDM from Adabas FDT**	Generates a DDM from an Adabas field definition table (FDT) and places it in the source area for further processing. If `0` (zero) is entered as DBID, the default DBID specified with the `UDB` profile parameter in the Natural parameter module is used. The generated DDM is placed in the source area for further processing.
B	**SQL Services (NDB)**	Only available if Natural for Db2 (NDB) is installed. **SQL Services (NDB)** are used to generate DDMs from Db2 tables as described in *SQL Services (NDB)* (Natural for Db2) in the *Database Management System Interfaces* documentation.
Z	**SQL Services (NSB)**	Not applicable.

Show Defined DBIDs and Used FNRs

This function shows you which database IDs (DBIDs) are defined, as well as all file numbers (FNRs) of a given DBID for which DDMs have been defined.

When you invoke this function, a menu appears from which you can select the subordinate functions described in the following section.

- Database IDs Defined in Natural
- File Numbers of Existing DDMs for a Database

Database IDs Defined in Natural

This function lists all DBIDs and appropriate database types specified with the Natural profile parameter `DB` (see also *DB - Database Types and Options* in the *Parameter Reference* documentation). The list does not contain the DBIDs of the default database type, which is shown at the top of the screen.

File Numbers of Existing DDMs for a Database

This function lists for a given DBID all file numbers for which DDMs have been defined.

To invoke this function, choose either of the following methods:

- In the menu **Show Defined DBIDs and Used FNRs**, in the **Code** field, enter an `F` and, if required, modify the DBID entered by default.

  Or:

  On the screen **Database IDs Defined in Natural**, in the command line, enter a DBID (valid values are `1` - `65535`) and press PF5.

DDM Specification

For each function provided in the SYSDDM utility menu, you can specify one or more parameters that determine which DDM(s) are processed for which database. The section below describes the fields where you enter these parameters.

Field	Explanation
DDM Name	The name of the DDM to be processed. You can also specify a range of names: use asterisk (*) to process all DDMs or use asterisk (*) notation to process particular DDMs. For example: `EMP*` selects all DDMs with names that start with EMP. Specifying ranges with the function **Delete DDMs** or **Copy DDM to Another FDIC File** invokes the SYSMAIN utility where you can further process the DDMs; see also the section *Processing DDMs* in the *SYSMAIN Utility* documentation.
FNR	The file number (FNR) of the database file for which the DDM is (to be) defined. The file number corresponds to the type of database. For example, if an Adabas file is used, the Adabas file number must be entered. For VSAM files, see *Natural for VSAM* in the *Database Management System Interfaces* documentation. Valid values are `0` - `65535`.
DBID	The database ID (DBID) which contains the database file referenced by the DDM. Valid values are `1` - `65535`, except `255`. If `0` (zero) is specified, the database ID specified with the Natural profile parameter `UDB` in the Natural parameter module is used. See also `UDB` in the *Parameter Reference* documentation. If no DBID is entered, it is generated dynamically at execution time based on the DBID of the system file FUSER.
Replace	Specifies whether to replace a DDM: `Y` — Yes. A DDM which is being copied or cataloged replaces an existing DDM with the same name. `N` — No. Existing DDMs with the same name are not replaced. This is the default setting. This function corresponds to the `REPLACE` option of the command `CATALOG` described in *Editor and System Commands*.
FDIC Type	The database type of the system file. Possible types are the same as for **DDM Type** (see below). This is an output field only.
DDM Type	The type of DDM, for example:

Field	Explanation
	A — Adabas. This is the default type. V — VSAM 2 — Db2 or SQL P — PROCESS (Entire System Server) C — Command processor S — Super Natural M — Natural Messaging The type of DDM corresponds to the type of database referenced by the DDM.
Adabas Password	The password required by Adabas if installed.
DBID Type	The database type of the database specified in the **DBID** field. Possible types are the same as for **DDM Type** (see above). Exception: for an Adabas database, the Adabas version (for example, 8) is displayed. This is an output field only.

33 Creating DDMs

This section describes how to create a DDM by either copying DDMs or creating DDMs directly from the field definitions in a database.

To create a DDM from an Adabas database

1 In the SYSDDM utility menu, enter the following:

In the **Code** field, enter either function code E (**Edit DDM**) or G (**Generate DDM from Adabas FDT**).

Leave the **Name** field empty.

In the **DBID** and **FNR** fields, enter the ID and file number of the database to be accessed by the DDM you want to create (see also *DDM Specification*).

For function E: An empty **Edit DDM** screen appears where you can enter field definitions as described in *Using the DDM Editor*.

For function G: The DDM is generated automatically from the FDT (field definition table) defined for the specified database and the **Edit DDM** screen appears with all relevant field definitions.

2 Depending on the function used, on the **Edit DDM** screen either enter the field definitions required (see *Using the DDM Editor*) or modify the field definitions generated from the FDT.

When you are finished editing the DDM, check the source and press PF3.

The SYSDDM utility menu appears.

3 In the SYSDDM utility menu, enter the following:

In the **Name** field, enter a valid DDM name (see also *Object Naming Conventions* in the *Using Natural* documentation). For function G, the **Name** field is preset to the default name

`FILE-file-number` where `file-number` corresponds to the file number entered in the **FNR** field.

In the **Code** field, enter function code `C` (**Catalog DDM**).

The DDM is saved as a cataloged object in the specified database ID and system file. See also *Cataloging a DDM*.

To create a DDM from a VSAM file

1 In the SYSDDM utility menu, specify the following:

In the **Code** field, enter an `E` (see also *Description of Functions*).

In the **DDM Type** field, enter a `V`.

In the **DBID** and **FNR** fields, enter the ID and file number of the database assigned to VSAM, which is to be accessed by the DDM you want to create.

For valid field entries, see *DDM Specification*.

An empty **Edit DDM** screen appears.

2 Enter the field definitions as described in *Using the DDM Editor*.

3 Catalog the DDM as described in *Catalog DDM* in *Natural for VSAM* in the *Database Management System Interfaces* documentation.

To create a DDM from a Db2 table

1 For Db2, proceed as described in *Generate DDM from an SQL Table* in the *Database Management System Interfaces* documentation.

2 Catalog the DDM as described in *Cataloging a DDM*.

To copy single or multiple DDMs

1 In the SYSDDM utility menu, specify the following:

In the **Code** field, enter an `M` (see also *Description of Functions*).

In the **Name** field, enter the name of the DDM (or a range of names) to be copied.

In the **DBID** and **FNR** fields, enter the database ID and file number of the FDIC system file that contains the required DDM(s).

For valid field entries, see *DDM Specification*.

The **Copy DDMs (via SYSMAIN)** window appears.

2 Specify the source and target environments for the DDM(s) to be copied as described in *Processing DDMs* in the *SYSMAIN Utility* documentation.

If you want to copy DDMs between different system files and/or hardware platforms, see also ***Maintaining DDMs in Different Environments***.

34 Invoking and Terminating the DDM Editor

- Invoking the Editor 272
- Terminating the Editor 273

The DDM editor is used to edit the source of a DDM.

This section describes how to invoke and terminate the DDM editor by using the SYSDDM menu.

Invoking the Editor

This section provides instructions for invoking the DDM editor from the SYSDDM menu for modifying an existing DDM or creating a new one.

To invoke the DDM editor for an existing DDM

- In the **Code** field of the SYSDDM menu, enter an `E` and, in the **DDM Name** field, enter the name of a DDM.

  If the specified DDM exists, the source code of the DDM is read into the source area and the **Edit DDM** screen similar to the example below appears:

```
16:03:29                  ***** Edit DDM (ADA) *****                 2004-11-22
DDM Name EMPLOYEES                        Def.Seq.        DBID     0 FNR   316
Command
I T L DB Name                             F       Leng S D Remark
- - - -- -------------- top ------------- - ---------- - - -------------------
    1 AA PERSONNEL-ID                     A          8   D
  *      CNNNNNNN
  *      C=COUNTRY
  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
    1 AF MAR-STAT                         A          1 F
  *      M=MARRIED
  *      S=SINGLE
  *      D=DIVORCED
  *      W=WIDOWED
    1 AG SEX                              A          1 F
    1 AH BIRTH                            D          6   D
    1 AH NJBIRTH                          I          2   D
  G 1 A1 FULL-ADDRESS

DDM EMPLOYEES read into source area.
```

To invoke the DDM editor for a new DDM, use one of the following methods

- With Adabas, use the function **Generate DDM from Adabas FDT** described in *Using SYSDDM Maintenance and Service Functions*.

  With Db2, use the function **Generate DDM from an SQL Table** described in the *Database Management System Interfaces* documentation.

  Or:

  In the **Code** field of the SYSDDM menu, enter an `E` and leave the **DDM Name** field empty.

  The **Edit DDM** screen similar to the example above appears.

  You can clear the source area by entering `CLEAR` in the command line.

Terminating the Editor

This section describes how to terminate an editor session and return to the SYSDDM utility menu.

To leave the DDM editor

- After editing, checking and cataloging the DDM source (see ***Using the DDM Editor*** and ***Cataloging a DDM***) contained in the source area, in the command line of the **Edit DDM** screen, enter a period (.) or press PF3.

  The SYSDDM utility menu appears.

Note: The DDM editor uses the editor profile option **Leave Editor with Unlock** to unlock source code when leaving the DDM editor. This option is described in ***General Defaults*** in *Editor Profile* in the section *General Information*.

35 Using the DDM Editor

- DDM Header Information 276
- Columns of Field Attributes 277
- Commands for Editing and Function Execution 282
- Specifying Extended Field Attributes 288

The DDM editor screen (**Edit DDM** screen) is organized in a table where the field definitions data is contained in rows and columns. All attributes that belong to a field defined for a DDM are contained in one row (that is, source-code line), separated by tabs.

This section describes the columns contained on the DDM Editor screen and the commands provided to create or modify a DDM field, navigate in the screen, or catalog a DDM source, for example.

DDM Header Information

This section describes the fields contained in the header at the top of the **Edit DDM** screen.

- Explanation of DDM Header Fields

Explanation of DDM Header Fields

Header Field	Description
Edit DDM (*DDM-type*)	The value displayed in parentheses next to the screen title **Edit DDM** denotes the type of DDM, for example: `ADA` — Adabas `VSAM` — VSAM `DB2` — Db2 `PROCESS` — Entire System Server `CMD-PROC` — Command processor `SNAT` — Super Natural `IBM MQ` — Natural Messaging
DBID	The database ID (DBID) as described for **DBID** in the section *DDM Specification*.
FNR	The number of the file being referenced in the database as described for **FNR** in the section *DDM Specification*.
DDM Name	The name of the DDM currently contained in the work area of the DDM editor.
Def.Seq.	The default sequence by which the file is read when it is accessed with a `READ LOGICAL` statement in a Natural program. See also the `READ` statement described in the *Statements* documentation. The default sequence is specified with the two-character field short name. The system validates the short name based on the selected file number. If the database is accessible, the short name is checked against the corresponding field in the database file. If such a field does not exist in the database, a selection list of valid short names is displayed. If the database cannot be accessed, no selection list is generated.

Header Field	Description
	The contents of this field are modifiable.

Columns of Field Attributes

This section describes the field attributes that can be defined in the rows and columns of the **Edit DDM** screen.

Column Heading	Field Attribute	
I	The line indicator. This column displays any of the following letters next to a line:	
	E	Line contains an error detected during execution of a CHECK command. See also CHECK in *Editor and System Commands*.
	S	Line contains a scanned value. See also SCAN in *Editor and System Commands*.
	X	Line is marked for a copy or move operation as described in ***Line Commands***.
	Y	Line is marked for a copy or move operation as described in ***Line Commands***.
T	The type of field:	
	blank	Elementary field. This type of field can hold data and does not contain any other fields. It can have only one value within a record.
	C	Only applies to a DDM that refers to an Adabas file. Specifies that a file is physically coupled to this DDM. Files are coupled by using Adabas descriptors. For further information on file coupling, refer to the *Adabas* documentation.
	G	Group. A group is a number of fields defined under one common group name. This allows you to reference several fields collectively by using the group name instead of the names of all the individual fields. Such fields cannot hold any data, but are only containers for other fields.

Column Heading	Field Attribute	
		Note: Groups defined in a DDM need not necessarily be defined as groups in the Natural object(s) that reference this DDM.
	M	Multiple-value field. This type of field can have more than one value within a record. See also *Multiple-Value Fields* in the *Programming Guide*.
	P	Periodic group. A group of fields that can have more than one value within a record. See also *Periodic Groups* in the *Programming Guide*.
	*	Comment line.
L	The level number assigned to the field. Levels are used to indicate the structure and grouping of the field definitions. This is relevant with view definitions, redefinitions and field groups (see the relevant sections in the *Programming Guide*). Valid level numbers are `1` - `7`. Level numbers must be specified in consecutive ascending order.	
DB	For Adabas files, the **DB** column displays the two-character short name of the corresponding field in the database file. For VSAM files, see *Natural for VSAM* in the *Database Management System Interfaces* documentation. For fields of the type `C` (see the **attribute T**), this column contains the short name of the Adabas descriptor used for file coupling.	
Name	The name of the field. It can be 3 - 32 characters long for Adabas fields and SQL columns. When generating a field from an Adabas FDT, the DDM editor assigns a default *`prefix`*`-FIELD` field name, for example, `AA-FIELD` for **DB** short name `AA`, or `AUAL-FIELD` for short name `Aa`. The rules to create a name comply with the naming conventions for user-defined variables (see the *Using Natural* documentation), except that the first character of the name must always be a Latin capital letter (A - Z). In addition, the name must not start with `L@`, `N@` or `O@`, where `@` is the character with hexadecimal value H'7C'. These prefixes identify **indicator fields** as explained in the following section. The field name is the name used in other Natural objects (for example, in a program) to reference the field. The field name is unique across the whole DDM. For fields of the type `C` (see the **attribute T**), this column contains the short name of the Adabas descriptor used for file coupling.	

Column Heading	Field Attribute
F	The Natural data format of an elementary field, such as `A` (alphanumeric), `P` (packed numeric) or `L` (logical). For valid Natural data formats, refer to *Format and Length of User-Defined Variables* in the *Programming Guide*.
Leng	The standard length of an elementary field. This length can be overridden by the user in a Natural program. For numeric fields (Natural data format `N`), the length is specified as *nn*`.`*m*, where *nn* is the number of digits before the decimal point an *m* is the number of digits after the decimal point. In the **Leng** 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 fields of the type `C` (see the **attribute T**), this column contains the name of the DDM used for file coupling.
S	Not applicable to VSAM. Null-value suppression option: *blank* — Indicates that standard Adabas suppression is used; that is, trailing blanks in alphanumeric fields and leading zeros in numeric fields are suppressed. `F` — Indicates that the field is defined with the Adabas fixed storage option; that is, no suppression is used and the field is stored without compression. `N` — 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 are not returned when the field is used in the `WITH` clause of a `FIND` statement, or in a `HISTOGRAM` or `READ LOGICAL` statement. If the **Remark** column contains `NC` (not counted), an `N` in this column indicates that the field is defined with the SQL null-value option. Below this field, the corresponding null-indicator field is listed. `M` — Indicates that the field is defined with the SQL null-value option `not null`. The **Remark** field (see *Specifying Extended Field Attributes*) for this field contains `NN NC` (`not null`, `not counted`). Below this field, the corresponding null-indicator field is listed.
D	The Adabas descriptor type of an elementary field that is not an array.

Column Heading	Field Attribute
	A descriptor can be used as the basis of a database search performed with the `READ` or the `FIND` statement. For example: a field from an Adabas database that has a `D` or an `S` in the **D** column can be used in the `BY` clause of the `READ` statement. Once a record has been read from the database using the `READ` statement, a `DISPLAY` statement can reference any field that has either a `D` or an `S` in this column. Descriptors types are:

blank	No descriptor. This field is not a descriptor.
`A`	Indicates that the field is an alternate index for a VSAM file.
`D`	Elementary descriptor. Value lists are created and maintained for this field by Adabas, so that this field can 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.
`H`	Hyperdescriptor. A hyperdescriptor is a user exit in Adabas. For Natural, it provides the same functionality as a phonetic descriptor (see below).
`N`	Non-descriptor. A non-descriptor is not a descriptor, but can be used as a search field for a non-descriptor search.
`P`	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.
`S`	Superdescriptor. If a superdescriptor contains a multiple-value field or a field from a periodic group (or part of such a field), the superdescriptor is marked with an `M` or a `P` in the field type column **T**; this enables Natural to create the correct search algorithms for this superdescriptor.
`U`	Subdescriptor or collation descriptor. If a subdescriptor contains a multiple-value field or a field from a periodic group (or part of such a field), you have to mark the subdescriptor with an `M` in the field type column **T**. This enables Natural to create the correct search algorithms for this subdescriptor. A collation descriptor is used to sort (collate) descriptor field values in a non-standard sequence. If a field is a collation descriptor, the **Remark** column (see below) reads:

Column Heading	Field Attribute	
		Collation, the number of the Adabas user exit that contains the collation sequence (1-8) and the short name of the parent field to which the collation sequence applies, for example, Collation 5 on AA.
	X	Alternate subdescriptor or superdescriptor; that is, an alternate index for a VSAM file.
	For VSAM files, see *Natural for VSAM* in the *Database Management System Interfaces* documentation. For fields of the type C (see the **attribute T**), this column contains the name of the DDM used for file coupling.	
Remark	A comment which applies to a field and/or the DDM.	

Indicator Fields

An indicator field is used to retrieve the length of a variable length field or information about the data significance (NULL value indicator) of a database field. An indicator field does *not* provide the contents of a database field.

A database field name starting with L@, N@ or O@ (where @ is the character with hexadecimal value H'7C') is interpreted as an indicator field. Therefore, a database field name must not start with any of these character strings unless it represents an indicator field.

The following happens when a DDM is initially generated.

- An L@*xxxxx* field is automatically added for every variable length field, where *xxxxx* is the name of the related field.

  This applies to long alpha (LA) and large object (LB) fields in an Adabas file, and VARCHAR and LOB fields in a Db2 table.

  If the length indicator relates to an LA, LB or LOB field, the Natural data format/length must be I4. For a VARCHAR field, the format/length must be I2.
- An N@*xxxxx* field is automatically added for a field that may contain a NULL value, where *xxxxx* is the name of the related field.

  This applies to Adabas fields defined with the SQL Null Value Option and Db2 fields which may have a NULL value by definition. The Natural data format/length of a NULL indicator field must be I2.
- An O@*xxxxx* is currently not assigned a particular retrieval function but is reserved for future extension.

An `0@xxxxx` is automatically added for a locator field of a Db2 LOB field. The Natural data format/length of a locator field must be I4.

Commands for Editing and Function Execution

This section provides information on the commands provided on the **Edit DDM** screen.

Line commands are used to copy, delete, insert or move single or multiple source-code lines. Additionally, they are used for invoking the extended field editing function (see *Specifying Extended Field Attributes*).

Editor or system commands, for example, are used to execute particular line commands, navigate in the DDM source or execute a SYSDDM function directly from the **Edit DDM** screen.

- Help on Commands
- Line Commands
- Editor and System Commands

Help on Commands

This section provides instructions for obtaining help information on the commands provided on the **Edit DDM** screen.

To display help information on commands

1 In the command line of the **Edit DDM** screen, enter `HELP`.

Or:

In the command line of the **Edit DDM** screen, enter a question mark (?).

The **Editor Help Info** screen appears.

2 Press ENTER to scroll down the help text and to exit the **Editor Help Info** screen.

The **Edit DDM** screen appears.

Line Commands

This section describes all line commands available on the **Edit DDM** screen and provides instructions for executing a line command.

To execute a line command

1 On the **Edit DDM** screen, next to the source line(s) to which the command applies, position the cursor in the column **T** and type in a line command by overriding any existing values in the column **T**, **L**, **DB** or **Name**.

2 Press ENTER.

Line Command	Explanation
.C(*nn*)	Copies a line once or *nn* times below the line in which the command was entered.
.CX(*nn*)	Copies the line marked with .X once or *nn* times below the line in which the command was entered.
.CY(*nn*)	Copies the line marked with .Y once or *nn* times below the line in which the command was entered.
.CX-Y(*nn*)	Copies a block of lines once or *nn* times as described in ***To copy or move a block of lines***.
.D(*nnnn*)	Deletes the line in which the command was entered or deletes *nnnn* lines starting with the line in which the command was entered. If *nnnn* is not specified, one line is deleted by default.
.E*nn*	Invokes the extended field attribute editing function as described in ***Specifying Extended Field Attributes***.
.I(*nn*)	Inserts *nn* blank lines below the line in which the command was entered, where *nn* can be in the range from 1 to 10. (With the next ENTER, lines that are left blank are eliminated again.) If *nn* is not (or not correctly) specified, 10 lines are inserted by default. To append lines to the source code, use the editor command ADD.
.MX	Moves the line marked with .X below the line in which the command was entered.
.MY	Moves the line marked with .Y below the line in which the command was entered.
.MX-Y	Moves a block of lines as described in ***To copy or move a block of lines***.
.X	Marks a single line or the first line of a block of lines to be copied or moved. A marked line is indicated by an X in the column **I**. See also ***To copy or move a block of lines***.
.Y	Marks the last line of a block of lines to be copied or moved. A marked line is indicated by a Y in the column **I**. See also ***To copy or move a block of lines***.

To copy or move a block of lines

1 In the first line of the block of lines to be copied or moved, enter the following line command:

```
.X
```

In the last line of the block of lines to be copied or moved, enter the following line command:

```
.Y
```

2 Press ENTER.

The block of lines is delimited as indicated by an `X` and a `Y` in the column **I**.

3 In the line below which you want to copy or move the marked block, enter one of the following line commands:

```
.CX-Y (nn)
```

or

```
.MX-Y
```

where `C` denotes copy and `M` denotes move. *nn* indicates the number of times the marked block is to be copied (if *nn* is not specified, the block is copied once by default).

4 Press ENTER.

The marked block is copied (once or *nn* times) or moved below the line in which the command was entered.

Editor and System Commands

This section describes the editor commands and Natural system commands available on the **Edit DDM** screen and lists equivalent PF keys (if relevant).

To execute an editor or a system command

- At the top of the **Edit DDM** screen, in the command line, enter an editor or a system command.

  Or:

  On the **Edit DDM** screen, press a PF key if assigned to an editor or system command.

  For example, to catalog a DDM you can either enter the command `CATALOG` or press PF11.

For an explanation of the symbols used in the syntax diagrams in the following tables, refer to *System Command Syntax* in the *System Commands* documentation. An underlined portion of a

command denotes a valid abbreviation. Note that the editor commands used to navigate in the DDM source are described in a separate table under *Editor Commands for Positioning*.

Command	Explanation
ADD	Appends 10 blank lines to the source code. (With the next ENTER, lines that are left blank are eliminated again.) To insert lines, see the line command .I.
CATALOG	Performs a syntax check and saves the DDM source currently contained in the source area as a cataloged object. CATALOG [*DDM-name*] [REPLACE] If the DDM source has already been cataloged, the REPLACE option must be used. **Equivalent PF key: PF11**
CHECK	Validates the DDM source in the source area against the Adabas FDT referenced by the DDM. Should any inconsistency occur, the source line of the field definition that caused the error is marked for correction as indicated by an E in the column **I**. **Equivalent PF key: PF10**
CLEAR	Clears the source area as described for the corresponding Natural system command CLEAR in the *System Commands* documentation.
CODEPAGE	Displays the code of the DDM currently in the editor and the session code page. The code page of the DDM currently in the editor may be different to the code page of this DDM on the system file, because a DDM may have been converted during reading it into the DDM editor.
DX	Deletes the line marked with the line command .X.
DY	Deletes the line marked with the line command .Y.
DX-Y	Deletes a block of lines delimited with the line commands .X and .Y.
EX	Deletes all lines above the line marked with the line command .X.
EY	Deletes all lines below the line marked with the line command .Y.
EX-Y	Deletes all lines except for the block of lines delimited with the line commands .X and .Y.
HELP or ?	Invokes the **Editor Help Info** screen with help information on editor commands.
LENGTH or SIZE	LENGTH [*from-field* *to-field*] or SIZE [*from-field* *to-field*]
LIST DDM	LIST DDM [*DDM-name*]

Command	Explanation
or LIST VIEW	or LIST View [*DDM-name*] Displays a single DDM source or a list of DDMs as specified with *DDM-name* as described for the corresponding Natural system command LIST in the *System Commands* documentation.
LOWERCASE or LC	Only applies to DDMs of the types A (Adabas) and 2 (Db2 or SQL). Disables automatic uppercase conversion of lowercase characters. If disabled, Lower ON is displayed in the upper right of the editor screen. For DDMs that contain lowercase characters, Lower ON is always set when they are opened for editing. See also UPPERCASE.
QUIT or .	Terminates the DDM editor and displays the SYSDDM utility menu. The DDM source is retained in the source area until another source is read into the source area (by any Natural editor) or until the Natural session is terminated. The DDM editor uses the editor profile option **Leave Editor with Unlock** to unlock source code when leaving the DDM editor. This option is described in ***General Defaults*** in *Editor Profile* in the section *General Information*. **Equivalent PF key: PF3**
READ	READ [*DDM-name*] Reads a DDM source into the source area. Any DDM source currently contained in the source area is overwritten.
RESET	Removes all marks from lines marked with an X (see the line command .X), a Y (see the line command .Y) or an E (error during CHECK) indicated in the column **I**.
SCAN	SCAN [*scan-value*] Scans for the search string specified with *scan-value*, for example: SCAN ABC or SCAN ABC D. If found, the line(s) that contain *scan-value* are marked with an S displayed in the column **I**. (Press ENTER to remove the marks.)
UNCATALOG	UNCATALOG [*DDM-name*] Deletes one or more DDMs from the current FDIC system file if *DDM-name* is specified (see also **DDM Name** in the section *DDM Specification*). Deletes the DDM source currently contained in the source area if *DDM-name* is not specified. This command corresponds to the Natural system command UNCATALOG described in the *System Commands* documentation.

Command	Explanation
UPPERCASE or UC	Only applies to DDMs of the types A (Adabas) and 2 (Db2 or SQL). Enables automatic uppercase conversion of lowercase characters. This is the default conversion mode. See also LOWERCASE.

Editor Commands for Positioning:

Editor Command	PF Key	Explanation
+ or +P	PF8 or ENTER	Scrolls down one page (20 lines).
- or -P	PF7	Scrolls up one page (20 lines).
+H	PF5	Scrolls down half a page (10 lines).
-H	PF4	Scrolls up half a page (10 lines).
X or Y		Positions in the line marked with the line command .X or .Y.
B or ++	PF9	Scrolls down to the last page.
T or --	PF6	Scrolls up to the first page.
+*nn*		Scrolls down *nn* lines.
-*nn*		Scrolls up *nn* lines.

Specifying Extended Field Attributes

The extended field editing function can be used to specify default field attributes for headers and edit masks, a field comment (remark) and a format option to be applied when the field is used in another Natural object (for example, in a program). In addition, for a DDM generated from a VSAM file you can display and edit VSAM-specific field attributes.

The header attribute specifies the default column header to be displayed above the field when it is output, for example, with a `DISPLAY` statement. The header corresponds to the text specified with the `HD` parameter within single quotation marks (`HD='text'`) as described in the *Parameter Reference* documentation. If no header is specified, the field name is used as column header.

The edit mask attribute specifies the default edit mask to be used when the field is output, for example, with a `DISPLAY` statement. The edit mask must conform with Natural syntax rules and be valid for the Natural data format and length of the field.

The remark attribute specifies a comment about the field.

The format option can be used to define variable length fields: when set to `LA`, the field is defined as Long Alpha (LA), when set to `LB`, the field is defined as Large Object (LOB). A Long Alpha field can be of format `A` or `U`, a Large Object field can be of format `A`, `U` or `B`.

Related Topics:

- `DISPLAY` and `INPUT` in the *Statements* documentation
- *EM - Edit Mask* in the *Parameter Reference* documentation

The section below covers the following topic:

- Editing Extended Field Attributes

Editing Extended Field Attributes

This section provides instructions for invoking and terminating extended field attribute editing for a single field or a range of consecutive fields.

To invoke extended field editing

1 For a single field:

Next to the field required, position the cursor in the column **T** column and type in the line command `.E` over the values in the columns **T** and **L**.

The **Extended Field Editing** screen for the field marked with the command is displayed as shown in the example of a DDM from Adabas below:

```
12:07:49                  ***** Edit DDM (ADA) *****                 2006-02-08
                          - Extended Field Editing -
DDM Name DDM-TEST                          Def.Seq.        DBID     0 FNR   316

I T L DB Name                              F       Leng S D
- - - -- -------------- top ------------- - ---------- - -
    1 AF LA-FIELD                          A      16381 F
-------------------------------------------------------------

Remark ........... LA____________________
Field Header ..... ______________________________________________________
Field Edit Mask .. ______________________________________________________

Format Option .... LA Long Alpha      (LA = LA field, LB = LOB field)
```

On the **Extended Field Editing** screen, as described earlier, you can specify a remark (comment), a field header, an edit mask and a format option.

For extended field editing in DDMs from VSAM, see *Extended Editing at Field Level* in the *Natural for VSAM* documentation).

2 For a range of fields:

1. Next to the first field to be selected, position the cursor in the column **T** and type in the following line command over the values in the columns **T** and **L**:

   ```
   .Enn
   ```

   where *nn* is the number of fields to be selected including the current one.

   The **Extended Field Editing** screen appears for the first field selected.

2. Enter or modify the field attributes required and press ENTER or PF3.

   The **Extended Field Editing** screen for the next field in sequence appears.

To terminate extended field editing

- Press ENTER or PF3.

  Any field modifications are saved and the **Edit DDM** screen appears.

36 Cataloging a DDM

- Instructions 292

You can save a DDM as a cataloged object in the specified Natural system file (see also ***Storing DDMs***).

For the naming conventions that apply to an object, refer to *Object Naming Conventions* in the *Using Natural* documentation.

Instructions

To catalog a DDM

- In the SYSDDM utility menu, specify the DDM you want to catalog (see also ***DDM Specification***) and execute the **Catalog DDM** function (see also *Description of Functions*).

  Or:

  From the **Edit DDM** screen, execute the `CATALOG` command as described in *Editor and System Commands*.

37 Listing DDMs

This section provides instructions for displaying a list of all DDMs available in an FDIC system file by using either the SYSDDM utility or the Natural system command `LIST`.

To list all DDMs with SYSDDM

- In the SYSDDM utility menu, specify the following:

  In the **Code** field, enter an `L` or `X` (see also *Description of Functions*).

  In the **Name** field, enter an asterisk (*), and, in the **DBID** and **FNR** fields, enter the ID and file number of the database to be accessed by the DDMs.

  For details on valid field entries, see *DDM Specification*.

  The **LIST DDMs** screen appears with a selection list of all DDMs contained in the specified FDIC system file.

To list all DDMs with `LIST`

- In the command line, enter the following system command:

  ```
  LIST DDM *
  ```

  The **LIST DDMs** screen appears with a selection list of all DDMs contained in the current FDIC system file.

For information on all options available with `LIST`, see the relevant section in the *System Commands* documentation.

38 Maintaining DDMs in Different Environments

To transfer DDMs (for example, copy or move) between different FDIC system files, and to perform a DDM operation (for example, delete or find) in a different environment, you can use the Natural utility SYSMAIN (see the *Utilities* documentation).

To transfer DDMs between different hardware platforms (z/OS, Linux, and Windows), you can use the Object Handler (see the *Utilities* documentation).

X Software AG Editor

The Software AG Editor is used to edit objects in several Natural utilities, Natural add-on products and other products.

General Information on the Software AG Editor

Invoking the Software AG Editor

Using the Editor Screen

Using Commands

Creating and Modifying Data

Setting the Editor Profile

Storing Data and Leaving the Software AG Editor

Summary of Line Commands

Summary of Main Commands

Related Topics:

- *Installing Software AG Editor* described in the Natural *Installation* documentation for the appropriate operating system.
- *Operating the Software AG Editor* described in the *Operations* documentation
- *Editors in the SPoD Environment* described in the *Unicode and Code Page Support* documentation

39 General Information on the Software AG Editor

The Software AG Editor is used to edit objects in Natural (including utilities such as SYSRPC and SYSBPM) and the following other products:

- Natural ISPF
- Entire Operations
- Entire Output Management
- Predict

Each of these products uses a subset of the complete Software AG Editor functionality.

If you want to use the Software AG Editor as an alternative to the Natural program editor, set the editor profile appropriately (see also ***General Defaults*** in ***Editor Profile*** in *Editors - General Information*).

The Software AG Editor provides Natural ISPF-like functionality to display and/or edit objects such as:

- Natural objects of the type program, subprogram, subroutine, helproutine, copycode, text, class or function
- PDS members and sequential files
- PANVALET members
- CA-LIBRARIAN members
- Job SYSOUT (browse mode only)
- Output in the user work pool
- List of system objects (browse mode only)
- Any text objects or tables

40 Invoking the Software AG Editor

The screen from which you enter the Software AG Editor depends on the application you are using and the type of object you want to edit or display.

For example, if you want to edit a Natural object in Natural ISPF, you might enter the Software AG Editor from the Natural object entry screen.

The ways of entering the Software AG Editor are described under the appropriate section headings in the documentation for the application you are using.

The commands available to you depend on the application you are using.

Examples of defined objects using the Software AG Editor are given in the appropriate sections of the documentation for the application you are using.

41 Using the Editor Screen

- Using the Input Areas 304
- Scrolling the Data in the Editing Area 305
- Locating a Line 307
- Showing or Hiding Lines 308
- Displaying Boundaries, Tab and Column Positions 309

If you specify a new object name for editing, an editor screen similar to the following appears:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-78K ------------ Columns 001 072
 COMMAND===>                                                   SCROLL===> CSR
****** ****************************** top of data ******************************
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
****** ***************************** bottom of data *****************************
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The layout of this screen can vary depending on the product you use.

The first line on the screen identifies the object to be edited according to the environment from which the Software AG Editor is called. In the above example, `JOB1JCL` is the name of the new object and `NATLIB1` identifies the object library which is a Natural library.

The input areas contained on the editor screen are explained in the following section.

Using the Input Areas

The Software AG Editor screen contains four input areas:

- **Command Line**

  The command line is indicated by **COMMAND===>**.

  It is used for entering an editor main command described in *Summary of Main Commands*.

- **Scroll Field**

  The scroll field is indicated by **SCROLL===>**.

  It is used for specifying default scrolling amounts described in *Settings for Scroll Field*.

- **Prefix Area**

  The leftmost six columns of the editor screen are referred to as the prefix area. The prefix area can contain a series of apostrophes (`' ' ' ' ' '`) or asterisks (`******`), a source-line number or text. The prefix area is used for entering line commands described in *Summary of Line Commands*.

- **Editing Area**

  The editing area is to the right of the prefix area. It is used for entering data.

  You can start your editing session by entering data into the Software AG Editor screen and using editor main commands (*Summary of Main Commands*) and line commands (*Summary of Line Commands*).

Scrolling the Data in the Editing Area

This section describes the commands and settings that can be used for scrolling the data contained in the editing area of the editor screen.

- PF Keys
- Settings for Scroll Field
- Main Commands for Scrolling

PF Keys

Commands for scrolling data are often assigned to the following PF keys:

- PF7 (main command `UP`) to scroll toward top of data.
- PF8 (main command `DOWN`) to scroll toward bottom of data.
- PF10 (main command `LEFT`) to scroll data to the left
- PF11 (main command `RIGHT`) to scroll to the right.

Settings for Scroll Field

In the scroll field (**SCROLL===>**), you can enter scroll settings. These settings are used to set the scroll amount for the PF keys, above, and some are also used with the scrolling main commands on the following page.

Possible settings for the scroll field are:

Scroll Setting	Explanation
number	Scrolls up or down a specified *number* of lines. Scroll right or left a specified *number* of columns.
CSR (default)	Scrolls down to cursor position if cursor is on a line of text. Cursor line becomes first line of text. When scrolling up, cursor line becomes last line of text. Scroll a page length, if cursor is in the command line. Scroll right or left to cursor position.
DATA	Scrolls a page length minus one line. When scrolling down, the bottom line becomes the top line. When scrolling up, the top line becomes the bottom line. When scrolling right, the last column becomes the first column. When scrolling left, the first column becomes the last column.
HALF	Scrolls half a page in any direction.
LINE	Scrolls up to beginning of line or down to end of line.
MAX	Scrolls to top or bottom of data. Scroll to extreme right or left of data.
PAGE	Scrolls a page length in any direction.
PARA	Scrolls up or down to first character of next paragraph.
SENT	Scrolls up to first character of current sentence or down to first character of following sentence. When scrolling up, if cursor is on first character of sentence, scroll to first character of previous sentence.
WORD	Scrolls up to first character of next word or down to first character of following word.

Main Commands for Scrolling

Apart from the LOCATE main command which scrolls data to a specified line, several main commands are available for vertical and horizontal scrolling.

The following table shows all possible scrolling commands and their meaning:

Main Command	Explanation
BOTTOM or ++	Scrolls to the end of the object being edited.
TOP or --	Scrolls to the beginning of the object being edited.
DOWN	Scrolls forward by the amount specified in the scroll field.
DOWN *n*	Scrolls forward by *n* lines.
+*n*	Scrolls forward by *n* lines.
UP	Scrolls backwards by the amount specified in the scroll field.
UP *n*	Scrolls backwards by *n* lines.
-*n*	Scrolls backwards by *n* lines.
LEFT	Scrolls to the left by the amount specified in the scroll field.
LEFT *n*	Scrolls to the left by *n* columns.
RIGHT	Scrolls to the right by the amount specified in the scroll field.
RIGHT *n*	Scrolls to the right by *n* columns.
FIX *n*	Specifies the number of columns *n*, starting with column 1, to remain in display when scrolling to the right.

Locating a Line

If you want to display a specific line at the top of your editor screen (that is, make it the current line), use the LOCATE main command with a parameter describing the line you want to become the current line.

Examples:

```
L 32
```

Makes line 32 the current line.

```
32
```

Same as above.

```
L .X
```

Makes the line labeled `.X` the current line.

```
L 'ABC'
```

Makes the first line that starts with the string `ABC` the current line (useful when browsing sorted data such as directory lists).

Differences between the LOCATE and FIND Commands

Note the following differences between the `LOCATE` and `FIND` commands:

- If you issue the `LOCATE` command with a character string (`L 'ABC'`), the string is only found if it starts in column 1. The `FIND` command searches the whole source work area;
- With the `LOCATE` command, it is assumed that the data to be searched is sorted in ascending alphabetical order;
- When a line is located with the `LOCATE` command, the cursor is placed in the prefix area; with the `FIND` command, the cursor is placed on the found string and the line is not necessarily made the current line.

Showing or Hiding Lines

You can exclude specific lines from the display by using the `EXCLUDE` main command. For example, the command:

```
EXC 'ABC' .X .Y ALL
```

excludes all lines with the string `ABC` within the block labeled `.X` and `.Y` from display. An unqualified `EXCLUDE` command excludes the current line. Each excluded line or block of lines is replaced by a line of dashes and a message informing you how many lines are excluded.

To recall excluded lines to display, use the `INCLUDE` main command. For example, the command:

```
IN C'Abc' ALL
```

includes all excluded lines containing the string `Abc` exactly as entered here. An unqualified `INCLUDE` command recalls the first line in the excluded block.

The `EXCLUDE` and `INCLUDE` main commands can be issued with the same string and search operands as described for the `FIND` command, except that the `ALL` search direction operand means exclude or include all lines with the given string.

Lines can also be excluded or recalled to display by using any of the line commands listed below.

Line Command	Explanation
X	Excludes this line from display.
X*n*	Excludes the next *n* lines from display.
XX	Marks the first line of a block of data to be excluded from display. A second XX line command is required to delineate the block. The exclusion is performed after the second XX is entered.
F	Recalls this line to display or recall the first line of the excluded block to display.
F*n*	Recalls the first *n* lines of the excluded block to display.
L*n*	Recalls the last *n* lines of the excluded block to display.

You can issue the main command XSWAP to exchange excluded lines with displayed lines.

Displaying Boundaries, Tab and Column Positions

You can display the positions of your boundaries (set with a BNDS main command) and tabs (set by a TABS main command), as well as the editing area column positions on any line by using the appropriate line command as listed below:

Line Command	Explanation
BNDS	Displays the boundary positions on this line.
COLS	Displays the column positions on this line.
TABS	Displays the tab positions on this line.

For detailed instructions and examples of setting boundaries and tabs, see the relevant sections in *Creating and Modifying Data*.

42 Using Commands

- Executing a Command 312

The Software AG Editor provides two types of command for controlling your editing session:

- **Main Commands**

  Main commands refer to the entire data currently contained in the source work area of the editor. The main commands available to you depend on the application you are using.

  The command format in the examples provided in the *Software AG Editor* documentation show the abbreviated form of each main command. These examples are by no means exclusive. For a complete summary of command syntax, see the section ***Summary of Main Commands***.

- **Line Commands**

  Line commands refer to the source line on which they are entered or to a block of data delineated by line commands.

  For a complete summary of line commands, see the section ***Summary of Line Commands***.

Commands are processed in the following order:

- Data updates
- Line commands
- Main commands

Executing a Command

Depending on the configuration of your installation, main commands and line commands can be entered in lower case. In the command descriptions provided in the *Software AG Editor* documentation, all commands appear in upper case to distinguish them as commands.

To execute a main command

- In the command line (**COMMAND===>**), enter a main command and press ENTER.

  You can enter several commands in the same input operation if you separate them with a semicolon (;).

To execute a line command

- In the leftmost column of the **prefix area**, next to the required source line, type a line command over the characters contained in these columns, and press ENTER.

  When you press ENTER, apostrophes (' ' ' ' ' ') or asterisks (******) are replaced by line numbers on used lines. Unused lines are automatically deleted.

Or:

In the first column of the editing area, enter a line command preceded by the **escape** character and press ENTER. The default escape character is the period (.).

Or:

In the command line, enter a line command preceded by a colon (:).

The cursor then marks the line to be addressed. Press ENTER.

43 Creating and Modifying Data

- Inserting and Deleting Lines 316
- Copying, Moving, Overlaying and Repeating Lines 317
- Copying or Moving a Window with Data 321
- Setting Horizontal and Vertical Boundaries 328
- Defining a Mask Line 330
- Ordering Data between Boundaries 331
- Centering Data 332
- Justifying Data 332
- Using the Physical or Logical Tabulator 335
- Sorting Lines in Alphabetical Order 340
- Finding a Character String 340
- Replacing a Character String 344

This section describes the functions that are available for all object types using main commands and/or line commands.

Inserting and Deleting Lines

This section describes the line commands and main commands available to insert and delete lines.

- Line Commands
- POWER Command
- DELETE Command

Line Commands

The following line commands are available to insert and delete lines on your editor screen.

Line Command	Explanation
D	Deletes this line.
D*n*	Deletes the next *n* lines.
DD	Marks the first line of a block to be deleted. A second DD line command is required to delineate the block. The deletion is performed after the second DD is entered.
DX	Deletes the line labeled .X.
DY	Deletes the line labeled .Y.
DX-Y	Deletes the block of lines from the line labeled .X to the line labeled .Y.
I	Inserts one line after this one. The editor switches to insert mode. This means if you type data or enter a blank in the new line and press ENTER, a new line is automatically inserted and the cursor placed in it. If you enter no new data in an inserted line and press ENTER, the editor leaves insert mode and the blank line is deleted. If you have defined a mask line and the MASK setting is ON, the mask line is inserted when you issue this command.
I*n*	Inserts *n* lines after this one. You can 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).
TE	Switches the editor to text enter mode. This means that beginning with this line the editor screen is blank (without line numbers) and you can enter data. When you press ENTER, any remaining blank lines are deleted, the line numbers are re-displayed and the text is reformatted within the set margins and with the specified justification. See also ***POWER Command***.
W	Opens a window of one line. No new line is inserted if you enter data in the window line and press ENTER.

Line Command	Explanation
W*n*	Opens a window of *n* lines. When you press ENTER, all unused lines are deleted.

POWER Command

You can also use the POWER main command to switch to text enter mode.

When you issue the POWER main command, you are presented with a blank editor screen (without line numbers) and you can enter data. When you press ENTER, any remaining blank lines are deleted, the line numbers are redisplayed and the text is reformatted within the set margins and with the specified justification.

DELETE Command

Line deletion can also be performed with the DELETE main command.

For example, the command:

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

deletes all lines containing the string Abc exactly as entered here between columns 10 to 30 within the block delineated by the labels .X and .Y.

You can specify all operands described for the FIND command above, except that the ALL direction operand specifies deletion of all lines with the given string. An unqualified DELETE command deletes the current line.

Copying, Moving, Overlaying and Repeating Lines

With the following line commands, you can copy, move or repeat lines or blocks of data.

Line Command	Explanation
A	Marks the target line for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. Data is inserted *after* this line.
B	Marks the target line for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. Data is inserted *before* this line.
C	Copies this line to the position marked by an A, B or O line command.
C*n*	Copies the next *n* lines to the position marked by an A, B or O line command.
CC	Marks the first line of the block to be copied. A second CC is required to the delineate block. Copying is performed after target has been marked by an A, B or O line command.
CX	Copies the line labeled .X. Inserts data after this line.
CY	Copies the line labeled .Y. Inserts data after this line.

Line Command	Explanation
CX-Y	Copies the block of lines from the line labeled .X to the line labeled .Y. Inserts data after this line.
M	Moves this line to the position marked by an A, B or O line command.
M*n*	Moves the next *n* lines to the position marked by an A or B line command.
MM	Marks the first line of the block to be moved. A second MM is required to delineate the block. The move is performed after the target has been marked by an A, B or O line command.
MX	Moves the line labeled .X. Inserts data after this line.
MY	Moves the line labeled .Y. Inserts data after this line.
MX-Y	Moves the block of lines from the line labeled .X to the line labeled .Y. Inserts data after this line.
O	Marks the target line for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. Data is merged with this line. (Only blank characters *within* boundaries are changed). See also ***Example of Overlaying Data***.
O*n*	Marks this line and the next *n* lines as the target lines for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. The moved or copied lines are merged with these lines, that is, blank characters in the lines are overlaid. See also ***Example of Overlaying Data***.
OO	Marks the first line of a block of target lines for a copy (C, C*n*, CC) or move (M, M*n*, MM) line command. A second OO command is required to mark the last line of the block of target lines. The moved or copied line(s) are merged with these lines, that is, blank characters in the lines are overlaid. See also ***Example of Overlaying Data***.
R	Repeat this line once.
R*n*	Repeat this line *n* times.
RR	Marks the first line of a block to be repeated. A second RR is required to delineate the block. The repeat is performed after second RR is entered.
RR*n*	Repeat block delineated by two RR*n* line commands *n* times. The) (parenthesis) line commands (see below), move text the full number of columns specified, but only *within* the set boundaries, therefore, part of the moved text could disappear.
)	Moves this line right by one column beginning with left boundary.
)*n*	Moves this line right by *n* columns.
))*n*	Marks first line of a block to be moved right by *n* columns. A second))*n* is required to delineate the block. The move is performed after the second))*n* command is entered. Two unqualified)) line commands move the block right by 1 column.
(	Moves this line left by one column.
(*n*	Moves this line left by *n* columns.

Line Command	Explanation
((*n*	Marks first line of a block to be moved left by *n* columns. A second ((*n* is required to delineate the block. The move is performed after the second ((*n* command is entered. Two unqualified ((line commands move the block left by 1 column. If you use the > (greater than) or < (less than) symbols (see below) to move data, the maximum move possible is up to the next non-blank character *within* the set boundaries.
>	Moves this line right by one column.
>*n*	Moves this line right by *n* columns.
>>*n*	Marks first line of a block to be moved right by *n* columns. A second >> is required to delineate the block. The move is performed after the second >> command is entered. Two unqualified >> line commands move the block right by 1 column.
<	Moves this line left by one column.
<*n*	Moves this line left by *n* columns.
<<*n*	Marks first line of a block to be moved left by *n* columns. A second << is required to delineate the block. The move is performed after the second << command is entered. Two unqualified << line commands move the block left by 1 column.

Example of Overlaying Data

An overlay line command (O, O*n* or OO) allows you to merge single-column lists into multi-column format (that is, tabular form). You can use an overlay line command in conjunction with a copy (C, C*n* or CC) or move (M, M*n* or MM) line command.

The following two figures illustrate this function:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
000090 //JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 // 'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
OO0140 //STEPLIB DD
000150 //        DD
000160 //        DD
OO0170 //        DD
MM0180                     DISP=SHR,DSN=SPF.SYSF.ADALOAD
000190                     DISP=SHR,DSN=SPF.SYSF.LOAD
000200                     DISP=SHR,DSN=SPF.SYSF.PROD.INST   * OPS INSTALL
MM0210                     DISP=SHR,DSN=SPF.SYSF.SOURCE      * OPS DOCUMENTS
000220 //DDCARD  DD *
000230 ADARUN DA=9,DE=3380,SVC=249
000240 //CMPRINT DD SYSOUT=X
000250 //CMPRT01 DD SYSOUT=X
000260 //CMWKF01 DD DUMMY
000270 //CMSYNIN DD *
000280 LOGON SYSMAIN2
000290 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000300 FIN
000310 /*
****** **************************** bottom of data ***************************




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

In the figure above, lines 180 to 210 are marked by the `MM` (Move) line command. They are to be overlaid on lines 140 to 170, which are marked by the `OO` (Overlay) line command.

This figure shows the result of the line commands displayed in the previous figure. Lines 180 to 210 have been overlaid on lines 140 to 170:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-76K ------------ Checkpoint done
 COMMAND===>                                                    SCROLL===> CSR
000090 //JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 // 'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB DD        DISP=SHR,DSN=SPF.SYSF.ADALOAD
000150 //        DD        DISP=SHR,DSN=SPF.SYSF.LOAD
000160 //        DD        DISP=SHR,DSN=SPF.SYSF.PROD.INST   * OPS INSTALL
000170 //        DD        DISP=SHR,DSN=SPF.SYSF.SOURCE      * OPS DOCUMENTS
000180 //DDCARD  DD *
000190 ADARUN DA=9,DE=3380,SVC=249
000200 //CMPRINT DD SYSOUT=X
000210 //CMPRT01 DD SYSOUT=X
000220 //CMWKF01 DD DUMMY
000230 //CMSYNIN DD *
000240 LOGON SYSMAIN2
000250 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000260 FIN
000270 /*
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

Copying or Moving a Window with Data

You can specify a window with data for move or copy operations. This allows you to copy or move data that does not start or end at the beginning or end of a line. This function can be performed by using editor line commands and/or main commands.

When you define a window, all data on your screen between start and end of the window becomes part of the window.

This section covers the following topics:

- Line Commands Available
- Main Commands Available

- Example of Using a Data Window

Line Commands Available

Line Command	Explanation
WS	Marks start of data window. Cursor position marks the column from which data is read. If the cursor is not in the line for which the command is entered, the window starts in column 1.
WS*n*	Data window starts in column *n* of this line.
WE	Marks end of data window. Works in the same way as WS. If the window is to start and end in the same line, type the WS command over the WE command. The editor acknowledges the set window with messages WS (WS*n*) and WE (WE*n*) in the prefix area, or with WW if the start and end of the window are in the same line. Before defining a new window, reset the old window with the RESET command to avoid a command conflict.
WE*n*	Data window ends in column *n* of this line.
WC	Copies the data window. The cursor position marks the column at which this line is to be split to insert the copied data.
WC*n*	Splits this line in column *n*, and copies the data between the two parts of the line.
WM	Moves the data window. Works in the same way as WC, but the original data is deleted after the copy operation.
WM*n*	Splits this line in column *n*, and moves the data between the two parts of the line.

Main Commands Available

Main Command	Explanation
WINDOW	Defines a window. The starting line and column, and the end line column are specified in the command parameters. At least one parameter is required. Examples: WINDOW 5 10 24 13 Defines a window starting in line 5 / column 24, and ending in line 10, column 13. WINDOW 5 10 24 Defines a window starting in line 5 / column 24, and ending in line 10 / last column. WINDOW 5 10 Defines a window starting in line 5 / first column, and ending in line 10 / last column. WINDOW 5 5 Defines a window starting in line 5 / first column, and ending in line 5 / last column.

Main Command	Explanation

`CWINDOW`	Copies a window defined with the `WINDOW` command. Optional parameters specify the line at which the window is to be inserted. Examples:
	
	`CWINDOW 5`
	
	Copies the window after line 5.
	
	`CWINDOW 5 24`
	
	Splits line 5 at column 24 and copies the window in between the two parts.

`DWINDOW`	Deletes a window of data defined by the `WINDOW` command.
`MWINDOW`	Moves window defined by the `WINDOW` command. Works like the `CWINDOW` command, but data in the original window is deleted after the copy operation.

Example of Using a Data Window

This section provides an example of defining and moving text with a data window by using either line commands or corresponding main commands.

The example refers to the text shown in Step 1 below and assumes that you want to move the whole sentence starting `Note that when...` (line 8) to follow the first sentence of the displayed text ending `...copy operations` (line 3).

To define and move a window using line commands

1 Type in text as shown below:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ---------------- Columns 001 072
 COMMAND===>                                                   SCROLL===> CSR
****** ****************************** top of data ******************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations. This
000004 allows you to copy or move data that does not start or end at the
000005 beginning or end of a line. This function can be performed using
000006 editor line commands and/or main commands.
000007
000008 Below are some examples of copying windows with data. Note that when
000009 you define a window, all data on your screen between start and end of
000010 the window become part of the window. Available line commands are:
****** ***************************** bottom of data ****************************



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

2 Type the line command `WS` in line 8, the first line of data to be moved, place the cursor in the required column (`N` of the word `Note`) and press ENTER.

The message `WS55` appears in the prefix area of line 8, indicating the column number selected:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K --------------- Block is pending
 COMMAND===>                                                   SCROLL===> CSR
****** ****************************** top of data ******************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations. This
000004 allows you to copy or move data that does not start or end at the
000005 beginning or end of a line. This function can be performed using
000006 editor line commands and/or main commands.
000007
WS55   Below are some examples of copying windows with data. Note that when
000009 you define a window, all data on your screen between start and end of
000010 the window become part of the window. Available line commands are:
****** ***************************** bottom of data ****************************
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

3 Type the line command `WE` in line 10, the last line of data to be moved, move the cursor to the last column to be moved (full stop (`.`) after `window`) and press ENTER.

The message `WE37` appears in the prefix area of line 10:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ---------------- Block is pending
 COMMAND===>                                                    SCROLL===> CSR
****** ****************************** top of data ******************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations. This
000004 allows you to copy or move data that does not start or end at the
000005 beginning or end of a line. This function can be performed using
000006 editor line commands and/or main commands.
000007
WS55   Below are some examples of copying windows with data. Note that when
000009 you define a window, all data on your screen between start and end of
WE37   the window become part of the window. Available line commands are:
****** **************************** bottom of data ****************************



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

4 Type the line command `WM` in line 3 (the data will be moved to the following line, line 4), and move the cursor to the column at which line 3 is to be split (the blank before the word `This`). Press ENTER.

The specified text section is moved:

```
EDIT-NAT:NATLIB1(WINEX)-Text->Struct-Free-78K ----------------- Checkpoint done
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ****************************
000001 Copy a Window with Data
000002
000003 You can specify a window with data for move or copy operations.
000004 Note that when
000005 you define a window, all data on your screen between start and end of
000006 the window become part of the window.
000007 This
000008 allows you to copy or move data that does not start or end at the
000009 beginning or end of a line. This function can be performed using
000010 editor line commands and/or main commands.
000011
000012 Below are some examples of copying windows with data.
000013                                      Available line commands are:
****** *************************** bottom of data ***************************

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

You can achieve the same result described in the instructions above when using the command sequence indicated below.

To define and move a window using main commands

1 Type in the text shown in **Step 1** above.

2 In the command line, enter the following:

```
WINDOW 8 10 55 37;MWINDOW 3 64
```

The specified text section is moved as illustrated in **Step 4** above.

Setting Horizontal and Vertical Boundaries

The editor provides commands which allow you to set horizontal and vertical boundaries within which certain functions can be performed such as the main commands `FIND`, `CHANGE`, `CENTER`, `ORDER`, `JLEFT` and `JRIGHT`, as well as their corresponding line commands (for example, `TC`, `TO`, `LJ` or `RJ`).

This section covers the following topics:

- Setting Boundaries
- Labeling Single or Multiple Lines

Setting Boundaries

With the `BNDS` main command, you can define horizontal boundaries as described in the following example instructions.

To set and display boundaries

1 Issue the following main command:

```
BNDS 20 50
```

Horizontal limits are set at columns 20 and 50.

2 To display the current boundary settings, issue the following line command:

```
BNDS
```

The following figure shows the result of the `BNDS 20 50` main command followed by a `BNDS` line command in line 2:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===>                                                   SCROLL===> CSR
****** ****************************** top of data ******************************
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
=bnds>                    <                            >
000010   RESET #JOBNAME(A8)
000020   RESET #FD(N3) #FL(A8) #FF(N3)
000030   RESET #TD(N3) #TL(A8) #TF(N3)
000040   COMPRESS *INIT-USER 'SM' INTO #JOBNAME LEAVING NO SPACE
000050   SET CONTROL 'WL60C6B005/010F'
000060   INPUT  'ENTER PARAMETERS FOR LIBRARY COPY:'
000070    /     'FROM:   DBID:' #FD 'FNR:' #FF 'LIB:' #FL
000080    /     'TO  :   DBID:' #TD 'FNR:' #TF 'LIB:' #TL
000090 // #JOBNAME JOB JWO,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT21,REGION=2000K,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB  DD  DISP=SHR,DSN=OPS.SYSF.V5.ADALOAD
000150 //         DD  DISP=SHR,DSN=OPS.SYSF.PROD.LOAD
000160 //DDCARD   DD  *
000170 ADARUN DA=9,DE=3380,SVC=249
000180 //CMPRINT  DD  SYSOUT=X
000190 //CMPRT01 DD SYSOUT=X
000200 //CMWKF01 DD DUMMY
000210 //CMSYNIN DD *
000220 LOGON SYSMAIN2
000230 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000240 FIN
000250 /*
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

Labeling Single or Multiple Lines

You can label the current line (line currently at the top of editing area) or a block of lines by using either the `LABEL` main commands or corresponding line commands.

To label current lines using `LABEL`

1 Issue the following main command:

```
LABEL .X
```

The current line is labeled `.X`.

2 To delineate a block of lines, you can now scroll to the next required delimiting line and issue the following main command:

```
LABEL .Y
```

The new current line is labeled `.Y` denoting the last line of a block of lines.

To label specific lines using line commands

1 Next to the first line to be labeled, issue the following line command:

```
.X
```

The specified line is labeled `.X`.

2 Next to the last line to be labeled, issue the following line command:

```
.Y
```

The specified line is labeled `.Y` delimiting a block of lines that starts with the line labeled `.X`.

Tip: You can use any string to label lines, for example `.START` and `.END`.

For examples of using labeled lines, see the sections *Finding a Character String* and *Replacing a Character String*.

Defining a Mask Line

You can define data that is automatically placed in a line added through a line insertion operation (for example, by using the line command `I` or `W`). Such a line is referred to as a mask line. A mask line is useful when you must write several lines of code which are identical or very similar.

Note: You can only have one mask line during an editing session. If you define a new mask line, any existing mask line definition is automatically updated with the new value.

To define and use a mask line

1 Issue the following line command:

```
MASK
```

A blank line indicated by `=mask>` appears above the line in which you entered the command.

2 In the blank line, enter the data you want to define as a mask line and press ENTER.

The mask line is now available for the current source until you update the mask with a new mask line or until you deactivate the mask function.

3 Issue the following main command:

```
MASK ON
```

The mask function is activated. The defined mask line now appears in all lines added through a line insert operation.

4 Issue an insert line command, for example:

```
I2
```

Two new lines are inserted into the source with the text of the mask line. The text of a mask line appears in all lines added with an insert command.

5 Modify the text in the new lines. If you do not modify the text, any inserted line is deleted the next time you press ENTER.

6 If required, deactivate the mask function with the following main command:

```
MASK OFF
```

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

See also `MASK` in *Summary of Main Commands*.

Ordering Data between Boundaries

You can change the indentation of specified lines by using the `ORDER` main command together with a boundary setting. For example, the command sequence:

```
BNDS 3;ORDER 5 20
```

moves lines 5 to 20 right to start in column 3.

Note: If the end of any ordered line traverses the right boundary, it is automatically split.

To re-justify the shifted data to the left, use a `JLEFT` command.

You can also change the indentation of lines or of a block of data by using line commands. Here, too, if the end of any line traverses the right boundary, it is automatically split.

Line Command	Explanation
`TF`	Orders data from the line on which it was entered to the end of the paragraph or to the next blank line with the right boundary. This line command can be entered with a numerical value specifying the right boundary, for example, the line command `TF50` orders data with column 50.
`TO`	Marks one line to be ordered.
`TOO`	Marks the first line of a block of data to be ordered. Requires a second `TOO` line command to delineate the block. Ordering is performed after the second `TOO` is issued.

Data can be ordered within set boundaries and justified to the left boundary, right boundary or both by using the `JUSTIFY` command. For example, the command:

```
BNDS 6 60;JUSTIFY BOTH
```

activates justification to columns 5 and 60. To perform the ordering, mark a block of data with two `TOO` line commands.

The editor also provides a line command with which you can split a single line into two. Type the line command `S` in the prefix area of the line you want to split, move the cursor to the position where the split is to occur and press ENTER.

Centering Data

The editor also provides commands with which you can center specified data within set boundaries. For example, the sequence:

```
BNDS 5 60;CENTER 5 15
```

centers data in lines 5 to 15 between columns 5 and 60.

Note: Only text already within the boundaries is centered. Text to the left and right of the boundaries is not affected.

Alternatively, you can use line commands to perform the centering function:

Line Command	Explanation
`TC`	Centers this line within the set boundaries.
`TCC`	Marks the first line in a block of data to be centered. Requires a second `TCC` line command to delineate the block. The centering is performed after the second `TCC` command is issued.

Justifying Data

A number of main commands and line commands are available to rearrange lines or blocks of data on your screen, depending on the setting of your horizontal boundaries (`BNDS` main command); see the section *Setting Horizontal and Vertical Boundaries*.

The `JLEFT` and `JRIGHT` main commands justifies the specified data with the left and right boundaries respectively. For example, the sequence:

```
BNDS 16 80;JLEFT 140 170
```

justifies the data between columns 16 to 80 in lines 140 to 170 with column 16.

The figure below illustrates this example:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===> BNDS 16 80;JLEFT 140 170                            SCROLL===> CSR
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB  DD          DISP=SHR,DSN=SPF.SYSF.ADALOAD
000150 //         DD          DISP=SHR,DSN=SPF.SYSF.LOAD
000160 //         DD          DISP=SHR,DSN=SPF.SYSF.PROD.INST  * OPS INSTALL
000170 //         DD          DISP=SHR,DSN=SPF.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
000190 ADARUN DA=9,DE=3380,SVC=249
000200 //CMPRINT  DD  SYSOUT=X
000210 //CMPRT01 DD SYSOUT=X
000220 //CMWKF01 DD DUMMY
000230 //CMSYNIN DD *
000240 LOGON SYSMAIN2
000250 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000260 FIN
000270 /*
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The following screen shows the result of the command displayed in the command line of the previous screen:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------ File has been ordered
 COMMAND===>                                                  SCROLL===> CSR
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
000140 //STEPLIB  DD  DISP=SHR,DSN=SPF.SYSF.ADALOAD
000150 //         DD  DISP=SHR,DSN=SPF.SYSF.LOAD
000160 //         DD  DISP=SHR,DSN=SPF.SYSF.PROD.INST  * OPS INSTALL
000170 //         DD  DISP=SHR,DSN=SPF.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
000190 ADARUN DA=9,DE=3380,SVC=249
000200 //CMPRINT  DD  SYSOUT=X
000210 //CMPRT01 DD SYSOUT=X
000220 //CMWKF01 DD DUMMY
000230 //CMSYNIN DD *
000240 LOGON SYSMAIN2
000250 CMD C C * FM  #FL DBID  #FD FNR  #FF TO  #TL DBID  #TD FNR  #TF REP
000260 FIN
000270 /*
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The sequence:

```
BNDS 10;JRIGHT 15
```

justifies the data on the right of column 10 in line 15 to the end of the screen with the last column of your editor screen (column 88 of your terminal screen).

Alternatively, you can justify lines or blocks of data by using any of the line commands listed in the following table:

Line Command	Explanation
LJ	Justifies the data within the set boundaries in this line with the left boundary.
LJJ	Marks the first line of a block of data within the boundaries (set with the BNDS main command) to be justified to the left. A second LJJ line command is required to delineate the block. Justification is performed after the second LJJ is issued.
RJ	Justifies the data within the set boundaries in this line with the right boundary.
RJJ	Marks the first line of a block of data within the boundaries (set with the BNDS main command) to be justified to the right. A second RJJ line command is required to delineate the block. Justification is performed after the second RJJ is issued.

You can also justify data to the left boundary, or order data between left and right boundary in conjunction with the JUSTIFY command.

For example, the command sequence:

```
BNDS 10 60;JUSTIFY LEFT
```

enables justification to the left boundary. Mark a block of data with two TOO line commands (explained below) to reformat the data between columns 10 and 60, justified to column 10.

Using the Physical or Logical Tabulator

This section provides instructions for using the TABS main command and/or the TABS line command to control tabulator settings.

In the examples of tabulation provided in this section, the ampersand (&) is assumed to be the tabulation character; the COLS line command has been issued to display column positions.

This section covers the following topics:

- Using the TABS Main Command
- Using the TABS Line Command
- Example 1 - Tab Positions
- Example 2 - TABS RIGHT
- Example 3 - TABS DECIMAL
- Example 4 - Mixed Justification

- Example 5 - Using a Blank as Tabulation Symbol

Using the TABS Main Command

When you issue the `TABS ON` main command, the standard tab positions set in your editor profile are turned on and `tabs on std` appears in your profile. Issue the `TABS OFF` main command to turn tabulation off again.

For detailed information on the command syntax that applies to `TABS`, see the relevant section in *Summary of Main Commands*.

This section covers the following topics:

- Setting Standard Tab Positions
- Setting the Logical Tab Character
- Setting Justification Parameters

Setting Standard Tab Positions

To turn tabulation on and set the standard tab positions for your profile to columns 10, 20, 30, 40 and 50, for example, issue the main command `TABS 10 20 30 40 50`.

Setting the Logical Tab Character

To turn tabulation on and set the logical tab character to & (ampersand), for example, issue the main command `TABS &`.

You can enter data and automatically move it to a specific tab position by preceding it with a logical tab character. One tab character moves the data to the next tab position, two tab characters moves the data to the second tab position, and so on.

Setting Justification Parameters

Apart from tab positions, you can specify the following parameters with the `TABS` main command:

Parameter	Explanation
`DECIMAL`	Orders data with the decimal point in the data at the tab position. See also ***Example 3 - TABS DECIMAL***.
`LEFT`	Orders data to the left of the tab position. See also ***Example 1 - Tab Positions***.
`RIGHT`	Orders data to the right of the tab position. See also ***Example 2 - TABS RIGHT***.

To display the current logical tab character and shift parameter (excluding tab positions), issue the `PROFILE` main command.

Using the TABS Line Command

When you issue the `TABS` line command in any line, the current tab positions set in your editor profile are displayed in that line and marked with asterisks (*) if no logical tab character has been set. This command does not turn tabulation on.

For example, issue the `TABS` line command to display the positions set with the main command `TABS 10 20 30 40 50`.

This displays the current tab positions as follows:

```
=tabs>            *        *        *        *        *        *
```

Setting Multiple Logical Tab Characters and Mixed Justification Parameters

To tabulate data in a specific column and with a specific shift, multiple logical tab characters and mixed justification parameters are possible.

To set the multiple logical tab characters, issue the `TABS` line command and type a special character over each asterisk (*). Any data typed in preceded by any of these logical tab characters are tabulated in the corresponding column.

To set the mixed justification parameters, type `L` (Left), `R` (Right) or `D` (Decimal) to the right of each logical tab character for left, right or decimal ordering.

See also ***Example 4 - Mixed Justification***.

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 EUR &13 IS
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
              15.27$    16.3 EUR             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 - 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
```

Sorting Lines in Alphabetical Order

You can sort data lines in ascending or descending alphabetical order according to sorting criteria. For example, the command:

```
SORT 10 15
```

sorts all lines in the source in ascending order according to the characters beginning in column 10 and ending in column 15.

To sort only a block of lines, for example, label the lines where the block is to start and end with `.X` and `.Y` respectively. The command:

```
SORT .X .Y D
```

sorts all lines in the block marked with `.X` and `.Y` in descending order.

To sort a block of lines according to the characters beginning in column 5 and ending in column 20, for example, label the lines where the block is to start and end with `.X` and `.Y` respectively. The command:

```
SORT 5 20 .X .Y
```

sorts all lines in the block marked by `.X` and `.Y` in ascending order according to the characters beginning in column 5 and ending in column 20.

Finding a Character String

To locate a specific character string, you can use the `FIND` main command with operands defining the string, the area to be searched and the direction of search. The cursor is placed on the first character of the string. If the line containing the string was excluded from display, it is now included in the display.

The following sections describe the possible command operands.

String Definition Operand

The string operand defines the character string to be located.

You can specify any of the following:

Operand	Explanation
`*`	Finds the string specified in previous `FIND` command.
`'abc'`	Finds the string `abc` regardless of whether the string is upper case or lower case.
`C'Abc'`	Finds the string exactly as entered here.
`P'a`*`(char)`*`c'`	Finds the string whose first character is `a` and third character is `c`. *`(char)`* stands for a special character acting as a wildcard character with the following meaning: `=` any character `§` alphabetic character `#` numeric character `$` special character `^` non-blank character `-` non-numeric character `<` lower-case character `>` upper-case character
`T'abc'`	Finds the string `abc` regardless of whether the string is upper case or lower case.
`X'D4A8'`	Finds the string that corresponds to hexadecimal `D4A8`.

String-Matching Operand

The string-matching operand specifies whether any special occurrence of the string is to be located. The following options are possible:

Operand	Explanation
`CHARS`	No restrictions (any occurrence of the string).
`PREFIX`	Only those occurrences which are the prefix of a word.
`SUFFIX`	Only those occurrences which are the suffix of a word.
`WORD`	Only those occurrences which form a word.

Default is `CHARS`.

Direction Operand

The direction operand specifies the direction of the search operation.

The following options are possible:

Operand	Explanation
ALL	Any occurrence of the string (search all directions).
FIRST	First occurrence of the string.
LAST	Last occurrence of the string.
NEXT	Next occurrence of the string starting from the cursor position.
PREV	Previous occurrence of the string.

Default is NEXT.

Line-Type Operand

This line-type operand specifies whether excluded or included lines only are to be searched. The following options are possible:

Operand	Explanation
X	Search excluded lines only.
NX	Search non-excluded lines only.

If this operand is omitted, the editor searches all data for the given string, included and excluded lines. If the string is found in an excluded line, it is returned to display.

Block Operand

If you have labeled lines or a block of lines, you can use the block operand to restrict the search area for the FIND command.

Two examples of the block operand follow:

Operand	Explanation
.X	Search from line labeled .X to end of data.
.X .Y	Search from line labeled .X to line labeled .Y.

where X and Y can be any alphabetic character or four-character string.

Columns Operand

The column operand allows you to restrict the search for the given string between certain columns. Below are two examples of the columns operand.

Operand	Explanation
`20`	Locate given string starting in column 20 (the first character of the string must be in column 20).
`20 40`	Locate given string anywhere between columns 20 to 40.

Examples of the FIND Command

```
F C'HILITE' X PREV
```

Find the previous occurrence of the string `HILITE` exactly as entered here; search excluded lines only.

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

Find the string starting `RCV` with a numeric fourth character within the block `.X .Z` and between columns 20 to 30.

```
F X'6C' SUFFIX NX
```

Find the character corresponding to the hexadecimal `6C` in non-excluded lines only. The character must end a word.

Command	Explanation
`F C'HILITE' X PREV`	Find the previous occurrence of the string `HILITE` exactly as entered here; search excluded lines only.
`F P'RCV#' .X .Z 20 30`	Find the string starting `RCV` with a numeric fourth character within the block `.X .Z` and between columns 20 to 30.
`F X'6C' SUFFIX NX`	Find the character corresponding to the hexadecimal `6C` in non-excluded lines only. The character must end a word.

If single quotation marks are part of the string to be found, you must use a different separator in the `FIND` command, for example double quotation marks:

```
FIND C"'string'"
```

You can repeat a previous `FIND` command with the `RFIND` main command.

Replacing a Character String

You can find and replace a given character string by another character string by using the CHANGE main command.

If apostrophes are part of the string to be replaced, you must use a different separator in the CHANGE command, for example quotation marks:

```
CHANGE "'string1'" "'string2'"
```

The same operands as for the FIND command can be used with the CHANGE command. For the CHANGE command, however, the ALL directions operand means change all occurrences of the specified string.

After the replace operation is performed, the message ==chg> appears in the prefix area of the changed line.

This section covers the following topics:

- Examples of the CHANGE Command
- RFIND and RCHANGE Commands

Examples of the CHANGE Command

Command	Explanation
CHG 'LOW' 'HIGH'	Replaces the first occurrence of LOW by HIGH (upper and lower case ignored).
CHG C'OPS' 'SPF' .X .Y 28 32 all	Replaces OPS (exactly as entered here) by SPF ; replace all occurrences in the block labeled .X and .Y and between the columns 28 and 32. See also the example screens below.
Repeated CHANGE **commands:**	
CHG * 'NEW'	Replaces the next occurrence of the string specified in the last CHANGE command by the new string NEW.
CHG 'OLD' *	Replaces OLD by the same new string as specified in the last CHANGE command.

The screen below illustrates the second example *before* the command is executed with ENTER:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------------ Columns 001 072
 COMMAND===> CHG C'OPS' 'SPF' .X .Y 28 32 ALL                   SCROLL===> CSR
****** ***************************** top of data ****************************
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000010   RESET #JOBNAME(A8)
000020   RESET #FD(N3) #FL(A8) #FF(N3)
000030   RESET #TD(N3) #TL(A8) #TF(N3)
000040   COMPRESS *INIT-USER 'SM' INTO #JOBNAME LEAVING NO SPACE
000050   SET CONTROL 'WL60C6B005/010F'
000060   INPUT  'ENTER PARAMETERS FOR LIBRARY COPY:'
000070    /     'FROM:   DBID:' #FD 'FNR:' #FF 'LIB:' #FL
000080    /     'TO  :   DBID:' #TD 'FNR:' #TF 'LIB:' #TL
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
.X     //STEPLIB  DD  DISP=SHR,DSN=OPS.SYSF.ADALOAD
000150 //         DD  DISP=SHR,DSN=OPS.SYSF.LOAD
000160 //         DD  DISP=SHR,DSN=OPS.SYSF.PROD.INST  * OPS INSTALL
.Y     //         DD  DISP=SHR,DSN=OPS.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The screen below illustrates the second example *after* the command is executed with ENTER:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-77K ------ 4 char  'OPS' changed
 COMMAND===>                                                   SCROLL===> CSR
****** ****************************** top of data ****************************
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000010   RESET #JOBNAME(A8)
000020   RESET #FD(N3) #FL(A8) #FF(N3)
000030   RESET #TD(N3) #TL(A8) #TF(N3)
000040   COMPRESS *INIT-USER 'SM' INTO #JOBNAME LEAVING NO SPACE
000050   SET CONTROL 'WL60C6B005/010F'
000060   INPUT  'ENTER PARAMETERS FOR LIBRARY COPY:'
000070    /     'FROM:   DBID:' #FD 'FNR:' #FF 'LIB:' #FL
000080    /     'TO  :   DBID:' #TD 'FNR:' #TF 'LIB:' #TL
000090 // #JOBNAME JOB NAT,MSGCLASS=X,CLASS=G,TIME=1400
000100 /*JOBPARM LINES=2000
000110 //COPY EXEC PGM=NATBAT,TIME=60,
000120 // PARM=('DBID=9,FNR=33,FNAT=(,15),FSIZE=19',
000130 //       'EJ=OFF,IM=D,ID='';'',MAINPR=1,INTENS=1')
.X     //STEPLIB  DD  DISP=SHR,DSN=SPF.SYSF.ADALOAD
==chg> //         DD  DISP=SHR,DSN=SPF.SYSF.LOAD
==chg> //         DD  DISP=SHR,DSN=SPF.SYSF.PROD.INST  * OPS INSTALL
.Y     //         DD  DISP=SHR,DSN=SPF.SYSF.SOURCE     * OPS DOCUMENTS
000180 //DDCARD   DD  *
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

All occurrences of the string `OPS` have been replaced by the string `SPF` between lines 140 and 170 and within the columns 28 to 32.

RFIND and RCHANGE Commands

When changing character strings, good use can be made of the `RFIND` (repeated `FIND`) and the `RCHANGE` (repeated `CHANGE`) commands, for example, the sequence:

```
FIND 'abc'
  CHANGE 'abc' 'def'
  RFIND
  RCHANGE
```

allows you to find occurrences of a certain string and optionally change them with relatively small effort.

44 Setting the Editor Profile

- Editor Profile Items 349

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.

You can modify single settings in your editor profile using appropriate commands. The new settings are valid for the remainder of the editing session or until you change them again using the appropriate commands.

To display the current settings of your editor profile

- Issue the following main command:

```
PROFILE
```

For details, see PROFILE in *Summary of Main Commands*.

The following lines appear at the top of the editor screen:

```
EDIT-NAT:NATLIB1(JOB1JCL)-Program->Struct-Free-78K ------------ Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
****** ****************************** top of data ******************************
=prof> date: 16/04/08 15:37:17 user: SAG      init size: 0 size: 0
=prof> var   - 88,..recovery on  (7 0)...autosave off... empty line off
=prof> mask off.caps on .hex off  nulls on  std.autoren off .auto order off
=prof> log on  1.mso on .fix off .escape off   tabs off
=prof> advance on .protect off.limit off
****** **************************** bottom of data ****************************
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

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

Editor Profile Items

This section describes the editor profile items and the appropriate main commands that can be used to change these items.

For explanations of the syntax symbols used in the main commands, refer to *System Command Syntax* in the *System Commands* documentation.

For detailed information on the main commands listed, refer to ***Summary of Main Commands***.

Profile Item	Description	Main Command
`advance`	Specifies whether the cursor moves to the next line automatically after a line update.	`ADVANCE {ON|OFF|PAGE}`
`auto order`	Automatically justifies text within defined boundaries.	`AORDER ON|OFF`
`autoren`	Specifies whether the editor activates or deactivates the `RENUMBER` function.	`AUTOREN ON|OFF`
`autosave`	Activates or deactivates automatic save when `END` command is issued.	`AUTOSAVE ON|OFF`
`caps`	Specifies whether data is to be translated into upper case.	`CAPS ON|OFF|PGM`
`date`	Current date and time. Non-modifiable item.	None
`empty line`	Specifies if lines containing only space characters are to be deleted automatically.	`EMPTY ON|OFF`
`escape`	Specifies whether escape character is used to precede line commands.	`ESCAPE ON|OFF [char]`
`fix`	Specifies whether fixed number of columns are displayed and how many columns are to be fixed.	`FIX ON|OFF n`
`hex`	Specifies whether data is to be displayed in hexadecimal format.	`HEX ON|OFF`
`init size`	Number of lines in object when editor was invoked. Non-modifiable item.	None
`limit`	Specifies the maximum number of lines to be searched by a `FIND` or `RFIND` command.	`LIMIT n`
`log`	Enables or disables log file. When enabled, `UNDO` command can be used to backout last changes.	`LOG ON|OFF`
`mask`	Activates or deactivates the mask line function.	`MASK ON|OFF`
`mso`	Indicates that multiple-session operations are allowed. A multiple-session operation is an operation in which data is exchanged between two editing sessions, for example, when copying data from one object to another.	None

Profile Item	Description	Main Command
	Non-modifiable item.	
`nulls`	Specifies whether the end of each source-code line is to be filled with null characters.	`NULLS ON|OFF`
`protect`	Specifies protection of the **prefix area**.	`PROTECT ON|OFF|INS`
`recovery`	Activates or deactivates recovery function, specifies how often checkpoint save is made.	`RECOVERY ON|OFF` *n*
`size`	Current number of lines in the object, excluding information lines (for example profile lines and message lines). Non-modifiable item.	None
`tabs`	Activates or deactivates tabulation.	`TABS ON|OFF`
`var`	Specifies current line length.	`BNDS` *n m*
`user`	Current logon user. Non-modifiable item.	None

45 Storing Data and Leaving the Software AG Editor

You can store data and/or leave the editor by using any of the following main commands:

Main Command	Explanation
CANCEL	Leaves the editor. Any changes made during this editing session do not take effect.
END	If AUTOSAVE is ON, stores data including any changes, and leaves the editor. If AUTOSAVE is OFF, the END main command acts as a CANCEL command if no data was modified. If changes were made, a message asks you to issue a SAVE or CANCEL command.
SAVE	Stores the data (including any changes) currently contained in the source work area of the editor. The editing session continues.

46 Summary of Line Commands

This section gives a short description of each line command provided by the Software AG Editor.

For general information on using line commands, see ***Using Commands***.

Line Command	Explanation
)	Moves this line right by one column.
)*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 one column.
(*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 one column.
>	Moves data in this line right by one column.
>*n*	Moves data in this line right by *n* columns (or up to last non-blank character: no data is 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, M*n*, MM) or copy (C, C*n*, CC) line command. The moved or copied line(s) are inserted *after* this line.
B	Marks the target line for a move (M, M*n*, MM) or copy (C, C*n*, CC) line command. The moved or copied line(s) are inserted *before* this line.

Line Command	Explanation
BNDS	Displays the boundary positions in this line.
C	Copies this line to the position indicated by an A, B or O line command.
C*n*	Copies the next *n* 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.
CX	Copies the line labeled .X. Inserts data after this line.
CY	Copies the line labeled .Y. Inserts data after this line.
CX-Y	Copies the block of lines from the line labeled .X to the line labeled .Y. Inserts data after this line.
COLS	Displays the column positions in this line.
D	Deletes this line.
D*n*	Deletes the next *n* lines.
DD	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.
DX	Deletes the line labeled .X.
DY	Deletes the line labeled .Y.
DX-Y	Deletes the block of lines from the line labeled .X to the line labeled .Y.
F	Includes the first excluded line.
F*n*	Includes the first *n* excluded lines.
I	Inserts one line. The editor switches to insert mode. This means if you type data or enter a blank in the new line and press ENTER, a new line is automatically inserted and the cursor placed in it. 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 main command EMPTY). You can also fill an inserted line with a predefined content (see the main command MASK).
I*n*	Inserts *n* lines. You can 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).
J	Joins next line with this one. Identical to TJ line command.
L*n*	Includes the last *n* excluded lines.
LC	Changes this line to lower case.
LC*n*	Changes the following *n* lines to lower case.
LCC	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.
LJ	Justifies the data within the set boundaries in this line with the left boundary.

Line Command	Explanation
LJJ	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.
M	Moves this line to the position indicated by an A, B or O line command.
M*n*	Moves the next *n* lines to the position indicated by an A, B or O line command.
MM	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.
MASK	Inserts a blank line in the editor into which you can create a mask. This line is inserted whenever the insert (I*n*) line command is used to create one or more new lines (see also the main command MASK).
MX	Moves the line labeled .X. Inserts it after this line.
MY	Moves the line labeled .Y. Inserts it after this line.
MX-Y	Moves the block of lines from the line labeled .X to the line labeled .Y. Inserts it after this line.
N	Modifications made in this line do not take effect when ENTER is pressed.
O	Marks this line as target line for a move (M, M*n*, MM) or copy (C, C*n*, CC) line command. The moved or copied line(s) are merged with this line, that is, blank characters in the line are overlaid.
O*n*	Marks the following *n* lines as target lines for a move (M, M*n*, MM) or copy (C, C*n*, CC) line command. The moved or copied lines are merged with these lines, that is, blank characters in the lines are overlaid.
OO	Marks the first line of a block of target lines for a move (M, M*n*, MM) or copy (C, C*n*, CC) line command. A second OO command is required to mark the last line of the block of target lines. The moved or copied line(s) are merged with these lines, that is, blank characters in the lines are overlaid.
R	Repeats this line once.
R*n*	Repeats this line *n* times.
RR	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.
RR*n*	Repeats the block of lines *n* times.
RJ	Justifies the data within the set boundaries in this line with the right boundary.
RJJ	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.
S	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.
T	Scrolls the data to make the marked line the top line.
TABS	Displays the tab positions in this line.

Line Command	Explanation
TC	Centers the data within the set boundaries in this line.
TCC	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.
TE	Switches editor to text enter mode (blank screen to end of screen).
TF	Joins this line with the following lines until the next blank line.
TF*n*	Joins this line with the following lines until the next blank line, ignoring data that is to the right of column *n*.
TI	Inverts sequence of all characters in the current line and within the set boundaries.
TII	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.
TJ	Joins next line with this one (identical to J line command).
TO	Joins this line with the next one.
TOO	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.
TS	Splits this line into two lines at the cursor position; an empty line is also automatically inserted, but deleted if unused (identical to S line command).
UC	Changes this line to upper case.
UC*n*	Changes the following *n* lines to upper case.
UCC	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.
W	Opens window with one line.
W*n*	Opens window with *n* lines.
WC	Copies the data window. The cursor position marks the column at which this line is to be split to insert the copied data.
WC*n*	Splits this line in column *n*, and copies the data between the two parts of the line.
WE	Marks end of data window. Works in the same way as WS. If the window is to start and end in the same line, replace the WS command by the WE command. The editor acknowledges the set window with message WW in the prefix area.
WM	Moves the data window. Works in the same way as WC, but the original data is deleted after the copy operation.
WM*n*	Splits this line in column *n*, and moves the data between the two parts of the line.
WS	Marks start of data window. The cursor position marks the column from which data is read. If the cursor is not in the line for which the command is entered, column 1 is taken.
WS*n*	Data window starts in column *n* of this line.
.X	Marks this line .X.
X	Excludes this line.
X*n*	Excludes the following *n* lines.

Line Command	Explanation
XX	Marks the first line of the block to be excluded. A second XX is required to mark the second line of the block.
.*label*	Marks this line with .*label*. The *label* can be any string of 1 to 4 alphabetic characters. See also the main command LABEL.
.Y	Marks this line .Y.

47 Summary of Main Commands

- ADVANCE 361
- AORDER 361
- AUTOREN 362
- AUTOSAVE 362
- BNDS 362
- BOTTOM 363
- CANCEL 363
- CAPS 363
- CENTER 364
- CHANGE 364
- COLS 367
- CURSOR 368
- CWINDOW 368
- DELETE 368
- DOWN 370
- DWINDOW 370
- EMPTY 371
- END 371
- ESCAPE 371
- EXCLUDE 372
- FIND 373
- FIX 376
- HEX 376
- INCLUDE 377
- JLEFT 377
- JRIGHT 378
- JUSTIFY 379
- LABEL 379
- LC 380
- LEFT 382
- LIMIT 382
- LOCATE 382

- LOG 383
- MASK 384
- MWINDOW 384
- NULLS 384
- ORDER 385
- POWER 385
- PROFILE 386
- PROTECT 386
- RCHANGE 387
- RECOVERY 387
- RENUMBER 387
- RESET 388
- RFIND 388
- RIGHT 388
- SORT 389
- TABS 389
- TOP 390
- UC 391
- UNDO 391
- UNREN 392
- UP 392
- WINDOW 392
- XSWAP 393
- Common Command Options 393

This section gives a short description of each main command provided by the Software AG Editor and a complete overview of the command syntax.

For general information on using main commands, see ***Using Commands***.

For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Note: If line numbers appear in the prefix area, column 8 on the screen corresponds to column 1 of the editing area. This is important when specifying columns in main commands.

ADVANCE

```
ADVANCE [ ON  ]
        [ OFF ]
        [ PAGE ]
```

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

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

If an unqualified `ADVANCE` command is issued, it is interpreted as `ADVANCE ON`. The default setting is `ADVANCE ON` and can be changed by editing your profile; see ***Setting the Editor Profile***.

AORDER

```
AORDER [ ON  ]
       [ OFF ]
```

This command is used to specify whether 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; see ***Setting the Editor Profile***.

AUTOREN

```
AUTOREN [ ON  ]
        [ OFF ]
```

For PDS members and sequential data sets only. This command is used to specify whether the editor automatically activates the RENUMBER function.

If an unqualified AUTOREN command is issued, it is interpreted as an AUTOREN ON command. The base setting can be changed by editing your profile; see ***Setting the Editor Profile***.

AUTOSAVE

```
{ AUTOSAVE } [ ON  ]
{ ASAVE    } [ OFF ]
```

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

If an unqualified AUTOSAVE command is issued, it is interpreted as an AUTOSAVE ON command. Default setting is AUTOSAVE ON and can be changed by editing your profile; see ***Setting the Editor Profile***.

BNDS

```
BNDS [ n m ]
     [ n   ]
```

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

These boundaries apply to the main commands FIND, CHANGE, CENTER, ORDER, JLEFT and JRIGHT, and their corresponding **line commands** (for example, TC, TO, LJ or RJ).

n	The number of the column at which the left boundary is to be placed.
m	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 editing area.

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

BOTTOM

```
BOTTOM
```

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

CANCEL

```
CANCEL
```

Backs out all changes to data made during the current editing session and leaves the editor. Any changes made since the last time you saved the data are lost.

CAPS

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

This command is used to switch upper-case translation on and off.

`ON`	The data is translated to upper case.
`OFF`	The data is not translated; that is, it remains as entered.
`PGM`	The data 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`. The default is `CAPS ON`. Edit your profile to change this; see *Setting the Editor Profile*.

CENTER

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

This command is used to center data.

ALL	Centers the data of all lines.
n	Centers the data from line *n* to the last line.
n m	Centers the data from line *n* to line *m*.

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

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

CHANGE

```
{ CHANGE } [ *              ] { *            }
{ CHG    } [ [T]'string1'   ] { [X]'string2' }
           [ C'string1'     ]
           [ X'string1'     ]
           [ P'string1'     ]

[ .X    ] [ n   ] [ ALL   ] [ CHARS  ] [ NX ]
[ .X .Y ] [ n m ] [ NEXT  ] [ WORD   ] [ X  ]
                  [ PREV  ] [ PREFIX ]
                  [ FIRST ] [ SUFFIX ]
                  [ LAST  ]
```

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

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

You can specify the string to be replaced (*string1*) as described in the following section.

T'*string1*'	Replaces *string1* irrespective of whether it occurs in lower case or upper case. This is the default.
'*string1*'	Same as T'*string1*'.
C'*string1*'	Replaces *string1* only if it occurs exactly as specified.
X'*string1*'	Replaces the string that corresponds to the specified hexadecimal character string *string1*. Replace it by the hexadecimal string *string2*.
P'*string1*'	Replaces *string1* which includes the following wildcard characters: = any character § alphabetic character /# numeric character $ special character ^ non-blank character - non-numeric character < lower-case character > upper-case character
*	Use the character string specified in a previous command (for example, FIND, CHANGE, EXCLUDE).
.X	See ***Line Specifications*** for an explanation.
.X .Y	
n *n m*	See ***Column Specifications*** for an explanation.
ALL NEXT PREV FIRST LAST	See ***Direction of Operation*** for an explanation.
CHARS WORD PREFIX SUFFIX	See ***Special Occurrences*** for an explanation.
NX	See ***Displayed or Non-Displayed Lines*** for an explanation.

X	

This section covers the following topics:

- Using CHANGE Together with Other Commands
- Examples of the CHANGE Command

Using CHANGE Together with Other Commands

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

To search the entire data 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 labeled with `.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 labeled with `.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 editing area showing column positions.

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

CURSOR

```
CURSOR
```

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

CWINDOW

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

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

n	The number of the line in which the data window is to be inserted.
m	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
                       .X .Y ] [ n
                                 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 as described in the following section.

T'*string*'	Deletes lines that contain the *string* irrespective of whether it is in lower case or upper case. This is the default.
'*string*'	Same as T'*string*'.
C'*string*'	Deletes lines that contain the *string* exactly as specified.
X'*string*'	Deletes lines that contain the string which corresponds to the specified hexadecimal character *string*.
P'*string*'	Deletes lines that contain the *string* which includes the following wildcard characters: = any character § alphabetic character # numeric character $ special character ^ non-blank character - non-numeric character < lower-case character > upper-case character
*	Uses the search string specified in a previous command (for example, FIND, CHANGE, EXCLUDE).
.X	See ***Line Specifications*** for an explanation.
.X .Y	
n *n m*	See ***Column Specifications*** for an explanation.
ALL NEXT PREV FIRST LAST	See ***Direction of Operation*** for an explanation.
CHARS WORD PREFIX SUFFIX	See ***Special Occurrences*** for an explanation.
NX X	See ***Displayed or Non-Displayed Lines*** for an explanation.

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

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

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 labeled with .X and .Y

DOWN

```
DOWN [n]
```

This command is used to scroll downwards in the data.

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

```
DWINDOW
```

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

EMPTY

```
EMPTY [ ON ]
      [ OFF ]
```

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

OFF	Blank lines are not deleted.
ON	Blank lines are deleted.

If you enter EMPTY without any parameter, it is interpreted as EMPTY ON. The default setting is EMPTY OFF (no suppression) and can be changed by editing your profile; see ***Setting the Editor Profile***.

END

Stores the data including all changes and leaves the Software AG Editor.

The command format is:

```
END
```

If AUTOSAVE is set to OFF and you have changed data, the editor asks you to issue a SAVE or CANCEL command.

ESCAPE

```
ESCAPE [ ON ]  [character]
       [ OFF ]
```

This command activates or deactivates the escape character to precede line commands entered in the first column of the data.

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`. Default is `ESCAPE OFF`. Can be changed by editing profile; see ***Setting the Editor Profile***.

EXCLUDE

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

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 as described in the following section.

`T'`*string*`'`	Excludes lines that contain the *string* irrespective of whether it is in lower case or upper case. This is the default.
`'`*string*`'`	Same as `T'`*string*`'`.
`C'`*string*`'`	Excludes lines that contain the *string* exactly as specified.
`X'`*string*`'`	Excludes lines that contain the string which corresponds to the specified hexadecimal character *string*.
`P'`*string*`'`	Excludes lines that contain the *string* which includes the following wildcard characters: = any character § alphabetic character # numeric character $ special character ^ non-blank character - non-numeric character < lower-case character > upper-case character
`*`	Uses the search string specified in a previous command (for example, `FIND`, `CHANGE`, `EXCLUDE`).
`.X`	See ***Line Specifications*** for an explanation.
`.X .Y`	
n	See ***Column Specifications*** for an explanation.

n m	
ALL NEXT PREV FIRST LAST	See *Direction of Operation* for an explanation.
CHARS WORD PREFIX SUFFIX	See *Special Occurrences* for an explanation.

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

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

Example 1:

```
EX 10
```

This command excludes line 10 from display.

Example 2:

```
EX C'NAME' .X ALL PREFIX
```

This command excludes from display all lines which contain `NAME` (in upper case as entered here) as a prefix to a word, starting from the line labeled with `.X`.

FIND

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

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.

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

You can specify the *string* as described in the following section.

T'*string*'	Searches for the *string* irrespective of whether it is in lower case or upper case. This is the default.
'*string*'	Same as T'*string*'.
C'*string*'	Searches for the *string* exactly as specified.
X'*string*'	Searches for the string that corresponds to the specified hexadecimal character *string*.
P'*string*'	Searches for a *string* which includes the following wildcard characters: = any character § alphabetic character # numeric character $ special character ^ non-blank character - non-numeric character < lower-case character > upper-case character
*	Searches for the *string* specified in the previous FIND command.
.X	See ***Line Specifications*** for an explanation.
.X .Y	
n *n m*	See ***Column Specifications*** for an explanation.
ALL NEXT PREV FIRST LAST	See ***Direction of Operation*** for an explanation.
CHARS WORD	See ***Special Occurrences*** for an explanation.

PREFIX SUFFIX	
NX X	See *Displayed or Non-Displayed Lines* for an explanation.

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.

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 with hexadecimal representation 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.

FIX

```
FIX [ ON
      OFF ] 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. The default setting is `FIX OFF 000` and can be changed by editing your profile; see ***Setting the Editor Profile***.

HEX

```
HEX [ ON
      OFF ]
```

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

The default setting is `HEX ON` and can be changed by editing your profile; see ***Setting the Editor Profile***.

INCLUDE

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

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.

Example:

```
IN C'NAME' .X ALL PREFIX
```

This command recalls all excluded lines with `NAME` as a prefix to a word exactly as entered here, starting from the line labeled `.X`.

JLEFT

```
      { ALL }
JLEFT { n   }
      { n m }
```

This command is used to align data left-justified.

`ALL`	Aligns the data of all lines.
n	Aligns the data from line *n* to the last line.
n m	Aligns the data from line *n* to line *m*.

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

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

See also the main command JRIGHT.

Example:

```
BNDS 10;JLEFT 15 20
```

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

JRIGHT

```
JRIGHT { ALL }
       { n   }
       { n m }
```

This command is used to align data right-justified.

ALL	Aligns the data of all lines.
n	Aligns the data from line *n* to the last line.
n m	Aligns the data from line *n* to line *m*.

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

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

See also the main command JLEFT.

Example 1:

```
BNDS 4 40;JRIGHT 6 18
```

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

Example 2:

```
BNDS 10;JRIGHT 15
```

The data to the right of column 10 in line 15 is right-aligned to the rightmost column of your editing screen.

JUSTIFY

```
JUSTIFY { LEFT  }
        { RIGHT }
        { BOTH  }
```

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

`TO` and `TOO` are used to join data lines with subsequent lines. Both commands apply only within the horizontal boundaries as set with the main command `BNDS`.

`LEFT`	The data is aligned to the left boundary.
`RIGHT`	The data is aligned to the right boundary.
`BOTH`	The data 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 data 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 top of the editing area) with the specified `.label`.

The *label* may be a string of 1 to 4 alphabetic 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`; see the section ***Summary of Line Commands***.

LC

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

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

You can specify the *string* as described in the following section.

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

`.X`	See *Line Specifications* for an explanation.
`.X .Y`	
n *n m*	See *Column Specifications* for an explanation.

`ALL`
`NEXT`
`PREV`
`FIRST`
`LAST`
	See *Direction of Operation* for an explanation.

`CHARS`
`WORD`
`PREFIX`
`SUFFIX`
	See *Special Occurrences* for an explanation.

`NX` `X`	See *Displayed or Non-Displayed Lines* for an explanation.

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 labeled with `.X` and `.Y` if they contain the string `NAME` (in upper case as entered here) as prefix to a word.

LEFT

```
LEFT [ n    ]
     [ LEFT ]
```

This command scrolls the data to the left.

n	Scrolls *n* number of columns to the left.
LEFT	Scrolls the maximum amount to the left.

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

LIMIT

```
LIMIT [n]
```

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. This setting can be changed by editing your profile; see ***Setting the Editor Profile***.

LOCATE

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

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

The command provides the following options:

0	Makes the first line of the data current.
n	Makes line *n* current.
.label	Makes the line labeled *.label* current.

The LOCATE command differs from the FIND 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 editing area; with the FIND command, the cursor is placed on the string searched and the line is not necessarily placed at the top of the editing area.

Examples:

```
LOC 32
```

Places line number 32 at the top of the editing area.

```
32
```

Same as above.

```
LOC .X
```

Places the line labeled .X at the top of the editing 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. When using the UNDO command you can consecutively back out changes made since the beginning of the editing session. Edit your profile to change the setting; see *Setting the Editor Profile*.

Important: You must ensure the LOG is activated before starting to edit.

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 by using the `MASK` line command. The mask function is useful when you must write several lines of code which are identical or very similar.

For detailed instructions on using the mask function, see ***To define and use a mask line***.

The default setting is `MASK OFF` and can be changed by editing your profile; see ***Setting the Editor Profile***.

MWINDOW

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

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

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

NULLS

```
NULLS [ ON  ]
      [ OFF ]
```

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

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

The default setting is NULLS ON and can be changed by editing your profile; see *Setting the Editor Profile*.

ORDER

```
ORDER { ALL }
      { n   }
      { n m }
```

This command is used to join data lines.

ALL	Joins all lines.
n	Joins the lines from line *n* to the last line.
n m	Joins lines from line *n* to line *m*.

The ORDER command applies only within the horizontal boundaries as set with the main 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 data lines, you can also use the line commands TF, TO and TOO.

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 editing area.

PROFILE

```
PROFILE [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).

PROTECT

```
PROTECT [ INS ]
        [ ON  ]
        [ OFF ]
```

This command is used to protect the **prefix area**. To enter line commands with the prefix area protected, type the line command in column 1 of the editing area preceded by the escape character.

INS	Protects the prefix area of lines added when using the insert line command.
ON	Activates protection.
OFF	Deactivates protection.

The default setting is PROTECT ON and can be changed by editing your profile; see ***Setting the Editor Profile***.

RCHANGE

```
RCHANGE
```

This command repeats the last `CHANGE` command.

RECOVERY

```
RECOVERY [ ON  ] [n]
         [ OFF ]
```

This command is used to activate or deactivate the recovery feature for the current editing session. You can also specify the number of updates to be performed before a checkpoint save is performed.

When using parameter *n*, you specify the number of updated lines after which a checkpoint save is performed.

The default setting is `RECOVERY ON` and can be changed by editing your profile; see ***Setting the Editor Profile***.

RENUMBER

```
RENUMBER [ ON          ]
         [ OFF         ]
         [ n1  n2  n3  ]
```

For PDS members and sequential data sets only. Specifies renumbering of the lines in the editing area according to the parameters.

`ON`	Activates renumbering.
`OFF`	Deactivates renumbering.
n1	Increment of numbering (default is in your edit profile).
n2	Starting column for the new line number (default: 73).
n3	End column for the new line number (default: 80).

To deactivate line renumbering, see the UNREN 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 [ n
        RIGHT ]
```

This command scrolls data to the right.

n	Scrolls *n* number of columns to the right.
RIGHT	Scrolls the maximum amount to the right.

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

SORT

```
SORT [n m] [ .X     ] [ A ]
           [ .X .Y  ] [ D ]
```

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.

n m	Sorts from column *n* to column *m*.
.X	Sorts from line labeled `.X` to end of object.
.X .Y	Sorts from line labeled `.X` to line labeled `.Y` (where `.X` and `.Y` are any string of up to four characters).
A	Sorts data in ascending order (A to Z).
D	Sorts data in descending order (Z to A).

Note: The maximum key length for sorting is 80 characters. All characters beyond that are ignored as sorting criteria.

TABS

```
TABS [ ON [tab-character]                                  ]
     [ OFF                                                 ]
     [ [ LEFT  ]                                           ]
     [ [ RIGHT ] [tab-character] [column...]               ]
     [ [ DECIMAL ]                                         ]
```

This command is used to control tabulator settings.

You can enable or disable logical or physical tabulation by using the command `TABS ON` or `TABS 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 by 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 it to a specific tab position by preceding it 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, and so on.

To display the current TABS command settings, issue the main command PROFILE.

To display the current tab positions, issue the line command TABS.

The default setting is TABS OFF blank and can be changed by editing your profile; see *Setting the Editor Profile*.

Apart from tab positions, you can specify the following parameters with the TABS command:

LEFT	Places the data left-justified at the tab position.
RIGHT	Places the data right-justified at the tab position.
DECIMAL	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 are tabulated in the corresponding column. You can type an L (for LEFT), an R (for RIGHT) or a D (for DECIMAL) after each tabulation character to specify placement of data for the tab position.

Note: For further instructions and examples of using tabulation, see also *Using the Physical or Logical Tabulator*.

TOP

TOP

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

UC

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

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 source to its original status at session begin.

You can specify the following parameters with the UNDO command:

ALL	All modifications made in the current editing session are backed out.
n	The last *n* modifications are backed out.

UNREN

```
UNREN [n m]
```

Deactivates the renumbering of lines.

n	Specifies the starting column of the line numbers (default: 73).
m	Specifies the end column of the line numbers (default: 80).

To activate line renumbering, see the RENUMBER command.

UP

```
UP [n]
```

This command scrolls upwards in the data.

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 l1 l2 [ n
               n m ]
```

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 *l1* and *l2* are required.

l1 l2	Defines a window starting at column 1 of line *l1* and ending in the last column of line *l2*.
l1 l2 n	Defines a window starting at column *n* of line *l1* and ending at the last column of line *l2*.
l1 l2 n m	Defines a window starting at column *n* of line *l1* and ending at column *m* of line *l2*.

Note that all data in the source work area within the specified points becomes part of the window. For an example, see the section *Copying a Window with Data*.

XSWAP

```
XSWAP
```

The command is used to exchange displayed lines with excluded lines. Lines are excluded by using the `EXCLUDE` command.

Common Command Options

There are some options which are available with several main commands. These options are described in the following section.

- Redisplay Feature
- Line Specifications
- Column Specifications
- Displayed or Non-Displayed Lines
- Direction of Operation
- Special Occurrences

Redisplay Feature

The editor provides a command redisplay feature: if you precede a command with two ampersands (&&), it remains displayed in the command line and is executed every time you press ENTER until you delete the command or replace it.

Line Specifications

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

`.X`	The command affects only the lines from the line labeled `.X` to the last line.
`.X .Y`	The command affects only the lines from the line labeled `.X` to the line labeled `.Y`.

`X` and `Y` can also be any label of 1 to 4 alphabetic characters (see the `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 data columns; the line numbers preceding the data 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 data you are editing.

n	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*).
n m	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:

NX	The command affects only non-excluded lines; that is, lines which are currently being displayed.
X	The command affects only excluded lines; that is, lines which are currently *not* being displayed as specified by the EXCLUDE command. An excluded line remains excluded from display if a main command function is performed on it.

Direction of Operation

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

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

Special Occurrences

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

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

Index

C

catalog
- DDM, 291
- source in program editor, 57

command
- cursor-sensitive commands
  - in program editor, 56
- editor commands in program editor, 42
- line commands in program editor, 51

copy
- DDM, 267

create
- DDM, 267

cursor position
- assign to PF key
  - in program editor, 53

customize
- features of program/data area editor, 25

D

data area editor
- customize editor features, 25

DDM
- catalog, 291
- copy, 267
- create, 267
- define field for, 277
- exit from editor, 273
- header information of, 276
- indicator field, 281
- list
  - , 293
- open in editor, 272
- save, 291
- specify field header/edit mask/remark, 288

disabled
- editors, 7

E

editor screen
- split to show two sources, 22

editors
- overview of, ix

exit
- from DDM editor, 273
- from program editor, 58

F

field
- define edit mask in DDM, 288
- define for DDM, 277
- define header in DDM, 288
- specify remark in DDM, 288

format
- settings in map editor, 112

H

header information
- of DDM, 276

help
- for SYSDDM utility functions, 260

I

indicator
- for field in DDM, 281

L

list
- DDMs, 293

lock
- activate/deactivate for object, 15
  - (see also unlock)

lower case
- change in program editor, 41

M

map editor
- default map settings, 106
- format settings, 112

O

open
- DDM editor, 272
- program editor, 38

P

profile

default map settings
map editor, 106
set for program/data area editor, 25
program editor
change lower/upper case, 41
cursor-sensitive commands, 56
customize editor features, 25
editor commands, 42
exit from, 58
line commands, 51
open, 38
PF key for cursor position, 53
save/catalog source, 57
use, 37

S

save
DDM, 291
source in program editor, 57
source code
save/catalog in program editor, 57
split
editor screen, 22
SYSDDM utility
help information for, 260
open DDM editor, 272

T

terminate
program editor, 58

U

unlock
object, 17
(see also lock)
upper case
change in program editor, 41
use
program editor, 37