



natural

Natural Optimizer
Compiler

Version 4.1.2

Natural Optimizer Compiler

This document applies to Natural Optimizer Compiler Version 4.1.2 and to all subsequent releases.

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

© Copyright Software AG 1979 - 2003.
All rights reserved.

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

Table of Contents

Natural Optimizer Compiler - Overview	1
Natural Optimizer Compiler - Overview	1
NOC - General Information	2
NOC - General Information	2
Natural Nucleus Optimization	2
Natural Optimizer Compiler	3
Installing the Natural Optimizer Compiler	5
Installing the Natural Optimizer Compiler	5
General Information	5
Installation Jobs	5
Using System Maintenance Aid	5
Prerequisites	5
Installation Tape - OS/390	6
Space Requirements	6
Copying the Tape Contents to Disk	6
Installation Tape - VSE/ESA	8
Copying the Tape Contents to Disk	8
Installation Tape - BS2000/OSD	10
Space Requirements	10
Copying the Tape Contents to Disk	10
Installation Tape - VM/CMS	12
Space Requirements	12
Copying the Tape Contents to Disk	12
Installation Procedure	13
Installation Verification	14
Using the Optimizer Compiler - Overview	15
Using the Optimizer Compiler - Overview	15
What is Compiled and What is Not	16
What is Compiled and What is Not	16
NOCSTAT Command	17
NOCSTAT Command	17
Invoking NOCSTAT	18
Generating Reports	19
Report Formats	21
Statement Category	21
Statement Type	24
Code Profile	25
Batch Execution	26
Displaying the Size of the Machine Code	28
Displaying the Size of the Machine Code	28
Optimizer Usage Examples	29
Optimizer Usage Examples	29
Example 1 - No Improvement	29
Example 2 - Considerable Improvement	30
Examples 3 and 4 - CPU Usage	32
Activating the Optimizer Compiler	34
Activating the Optimizer Compiler	34
Macro NTOPT	34
Dynamic Profile Parameter OPT	34
System Command NOCOPT	35
Natural Statement OPTIONS	35
Optimizer Options	36
Optimizer Options	36

List of Options	36
Example of INDEX and OVFLW	38
Optimum Code Generation	39
PGEN Option	39
Setting PGEN	39
Sub-Options of the PGEN Option	40
Output of the PGEN Option	40
Working with the PGEN Output	41
Influence of other Natural Parameters	43
Performance Considerations	44
Performance Considerations	44
Formats	44
Arrays	44
Alphanumeric Fields	45
DECIDE ON	45
Numeric Values	45
Rules of Thumb	45
Variable Positioning	45
Variable Caching	45
Example of Variable Caching	46
NODBG	46
Listing Zaps	47
Listing Zaps	47

Natural Optimizer Compiler - Overview

This documentation for Natural Optimizer Compiler describes various aspects which should be taken into consideration when the Natural Optimizer Compiler is installed at your site.

In the remainder of the Natural Optimizer Compiler documentation the Natural Optimizer Compiler is also referred to as NOC.

- **General Information** Various aspects of the Natural Optimizer Compiler and how to benefit most from the Natural Optimizer Compiler.
- **Installing the Optimizer Compiler** Installation of the Natural Optimizer Compiler.
- **Using the Optimizer Compiler** Statements and programs used for compilation.
Statistical data on programs suitable for processing by the Natural Optimizer Compiler: NOCSTAT command.
Examples of when to use the Optimizer Compiler.
- **Activating the Optimizer Compiler** How to switch on the Natural Optimizer Compiler.
- **Optimizer Options** Various options of the Natural Optimizer Compiler.
How to apply PGEN to output generated code and internal Natural structures for examination.
Influence by other Natural parameters.
- **Performance Considerations** How to achieve best performance considering data formats, arrays, alpha fields, DECIDE ON and numeric values.
- **Listing Zaps** How to receive an overview of the Zaps that have been applied to the Natural Optimizer Compiler.

NOC - General Information

This section describes various aspects which should be taken into consideration when the Natural Optimizer Compiler is installed at your site. The information provided in this documentation helps you to make full use of the benefits offered by the Natural Optimizer Compiler.

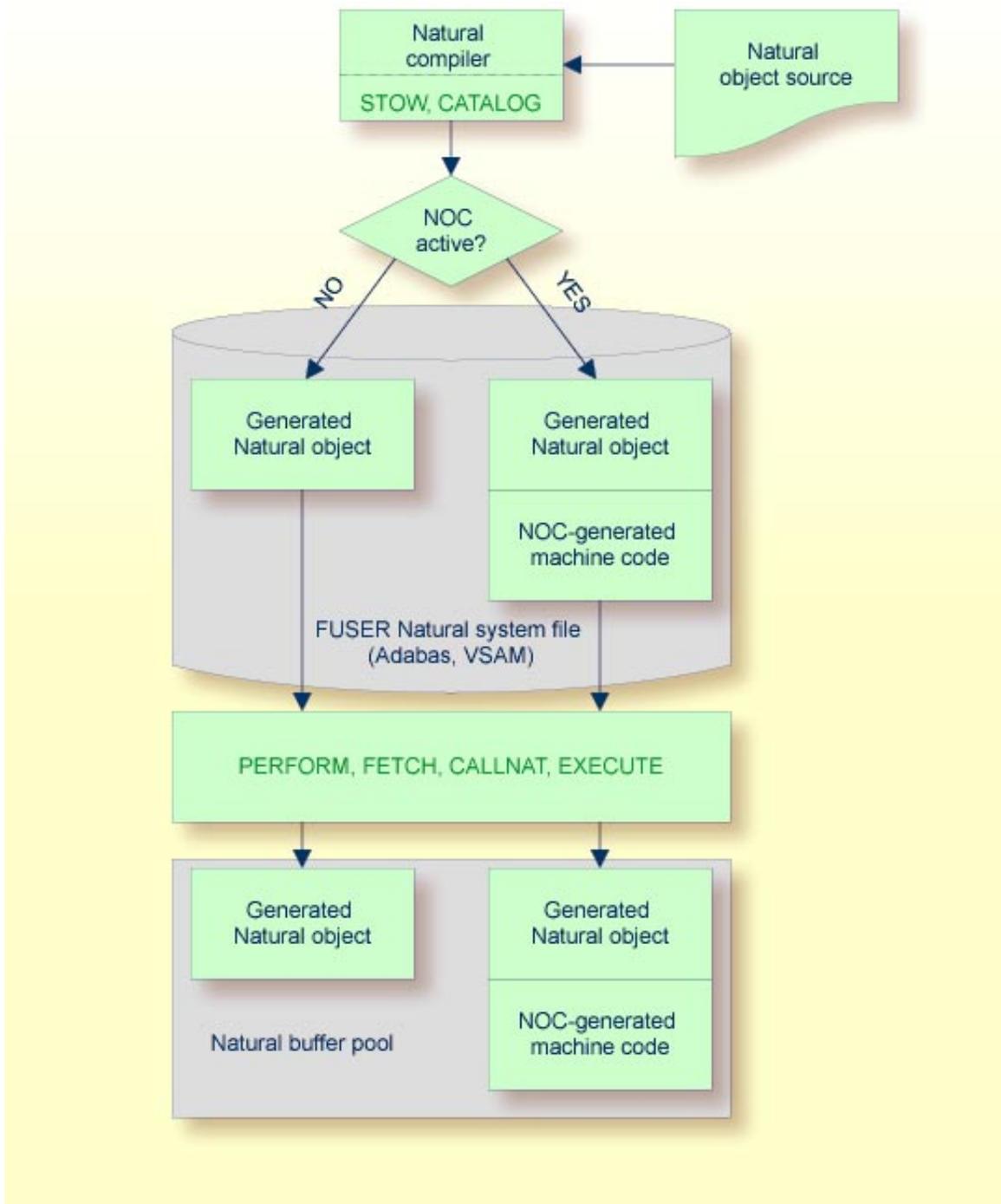
This section covers the following topics:

- Natural Nucleus Standard Optimization
 - Natural Optimizer Compiler
-

Natural Nucleus Optimization

The Natural nucleus optimizes simple arithmetic, assignment, and comparison statements by translating parts of them into machine code. All programs are optimized automatically in this way.

The following graphic illustrates how the Natural Optimizer Compiler generates machine code when a Natural programming object is compiled or executed:



Natural Optimizer Compiler

The Natural Optimizer Compiler goes one step further than standard optimization. It compiles not only simple statements to machine code, but also complex statements and statement sequences.

The compiled code is further optimized as far as array range operations, field concatenation, and optimum base register assignment are concerned.

All statements (including arithmetic operations) optimized with NOC provide the same results as the same statements generated by standard Natural.

To activate the Natural Optimizer Compiler (see the relevant section), use the macro NTOPT in the Natural parameter module, the dynamic profile parameter OPT, the system command NOCOPT, or the OPTIONS statement.

All programs that are cataloged (STOW or CATALOG system command) with the Natural Optimizer Compiler activated are compiled to machine code. This will also result in the object code size of the programs being larger than usual, depending on how much of the program can be optimized.

A program executed with the RUN system command is compiled to machine code if the Natural Optimizer Compiler is activated with the system command NOCOPT, the macro NTOPT or the OPTIONS statement for all or part of the program.

To see if a program is suitable for compilation with the Natural Optimizer Compiler, use the NOCSTAT command as described in the relevant section.

Note:

The dynamic recatalog feature (RECAT=ON) cannot be used with programs compiled to machine code.

To execute programs that have been compiled with the Natural Optimizer Compiler, it is not necessary that the Natural Optimizer Compiler is installed.

Installing the Natural Optimizer Compiler

This section describes how to install the Natural Optimizer Compiler (also referred to as NOC) in the various environments supported.

- General Information
 - Prerequisites
 - Installation Tape - OS/390
 - Installation Tape - VSE/ESA
 - Installation Tape - BS2000/OSD
 - Installation Tape - VM/CMS
 - Installation Procedure
 - Installation Verification
-

General Information

Below is information on:

- Installation Jobs
- Using System Maintenance Aid

Installation Jobs

The installation of Software AG products is performed by installation jobs. These jobs are either created manually or generated by Software AG's System Maintenance Aid (SMA).

For each step of the installation procedure described below, the job number of a job performing the corresponding task is indicated. This job number refers to an installation job generated by SMA.

Using System Maintenance Aid

For information on using SMA for the installation process, refer to the System Maintenance Aid documentation.

Prerequisites

Products and versions are specified in the sections Natural and Other Software AG Products and Operating/Teleprocessing Systems Required in the current Natural Release Notes for Mainframes.

Installation Tape - OS/390

The installation tape contains the dataset listed in the table below.

Dataset Name	Contents
NOC nnn .LOAD	This dataset contains the Natural Optimizer Compiler load modules.

The notation nnn in dataset names represents the version number of the product.

For a detailed description of the installation tape refer to the Report of Tape Creation which accompanies the tape.

Space Requirements

The space the dataset requires on disk is shown in the Report of Tape Creation.

Copying the Tape Contents to Disk

If you are using System Maintenance Aid (SMA), refer to the SMA documentation (included on the current edition of the Natural documentation CD).

If you are **not** using SMA, follow the instructions below.

This section explains how to:

- Copy data set COPY.JOB from tape to disk.
- Modify this data set to conform with your local naming conventions.

The JCL in this data set is then used to copy all data sets from tape to disk.

If the datasets for more than one product are delivered on the tape, the dataset COPY.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk.

After that, you will have to perform the individual install procedure for each component.

Step 1 - Copy data set COPY.JOB from tape to disk

The data set COPY.JOB (label 2) contains the JCL to unload all other existing data sets from tape to disk. To unload COPY.JOB, use the following sample JCL:

```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
//* -----
//COPY EXEC PGM=IEBGENER
//SYSUT1 DD DSN=COPY.JOB,
// DISP=(OLD,PASS),
// UNIT=(CASS,,DEFER),
// VOL=(,RETAIN,SER=<Tnnnnn>),
// LABEL=(2,SL)
//SYSUT2 DD DSN=<hi lev>.COPY.JOB,
// DISP=(NEW,CATLG,DELETE),
// UNIT=3390,VOL=SER=<vvvvvv>,
// SPACE=(TRK,(1,1),RLSE),
// DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//
```

Where:

<hilev> is a valid high level qualifier

<Tnnnnn> is the tape number

<vvvvvv> is the desired volser

Step 2 - Modify COPYTAPE.JOB

Modify the COPYTAPE.JOB to conform with your local naming conventions and set the disk space parameters before submitting this job:

- Set HILEV to a valid high level qualifier.
- Set LOCATION to a storage location.
- Set EXPDT to a valid expiration date.

Step 3 - Submit COPY.JOB

Submit COPY.JOB to unload all other data sets from the tape to your disk.

Installation Tape - VSE/ESA

The installation tape contains the following dataset:

Dataset Name	Contents
NOC nnn .LIBR	LIBR backup file.

The notation nnn in dataset names represents the version number of the product.

Copying the Tape Contents to Disk

If you are using System Maintenance Aid (SMA), refer to the SMA documentation (included on the current edition of the Natural documentation CD).

If you are **not** using SMA, follow the instructions below.

This section explains how to:

- Copy data set COPYTAPE.JOB from tape to library.
- Modify this member to conform with your local naming conventions.

The JCL in this member is then used to copy all data sets from tape to disk.

If the datasets for more than one product are delivered on the tape, the member COPYTAPE.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk, except the datasets that you can directly install from tape, for example, Natural INPL objects.

After that, you will have to perform the individual install procedure for each component.

Step 1 - Copy data set COPYTAPE.JOB from tape to disk

The data set COPYTAPE.JOB (file 5) contains the JCL to unload all other existing data sets from tape to disk. To unload COPYTAPE.JOB, use the following sample JCL:

```

* $$ JOB JNM=LIBRCAT,CLASS=0,                                     +
* $$ DISP=D,LDEST=(*,UID),SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB LIBRCAT
* *****
*       CATALOG COPYTAPE.JOB TO LIBRARY
* *****
// ASSGN SYS004, $NNN$                                            <----- tape address
// MTC REW,SYS004
// MTC FSF,SYS004,4
ASSGN SYSIPT,SYS004
// TLBL IJSYSIN,'COPYTAPE.JOB'
// EXEC LIBR,PARM='MSHP; ACC S= $lib.sublib$ '                   <----- for catalog
/*
// MTC REW,SYS004
ASSGN SYSIPT,FEC
/*
/&
* $$ EOJ

```

Where:

NNN is the tape address

lib.sublib is the library and sublibrary of the catalog

Step 2 - Modify COPYTAPE.JOB

Modify COPYTAPE.JOB to conform with your local naming conventions and set the disk space parameters before submitting this job:

Step 3 - Submit COPYTAPE.JOB

Submit COPYTAPE.JOB to unload all other data sets from the tape to your disk.

Installation Tape - BS2000/OSD

The installation tape contains the following dataset:

Dataset Name	Contents
NOC nnn .PAMS	Optimizer Compiler module library.

The notation nnn in dataset names represents the version number of the product. For a detailed description of the installation tape refer to the Report of Tape Creation which accompanies the tape.

Space Requirements

The space the dataset requires on disk is shown in the Report of Tape Creation.

Copying the Tape Contents to Disk

If you are not using SMA, use the procedure described below. In this procedure, the values specified below must be supplied.

To copy the datasets from tape to disk, perform the following steps:

1. Copy the Library SRV nnn .LIB from Tape to Disk

This step is not necessary if you have already copied the library SRV nnn .LIB from another Software AG tape. For more information, refer to the element #READ-ME in this library.

The library SRV nnn .LIB is stored on the tape as the sequential file SRV nnn .LIBS containing LMS commands. The current version nnn can be obtained from the **Report of Tape Creation**. To convert this sequential file into an LMS-library, execute the following commands:

```

/IMPORT-FILE  SUPPORT=*TAPE(FILE-NAME=SRV $nnn$ .LIBS, -
/  VOLUME=<volser>, DEV-TYPE=<tape-device>)
/ADD-FILE-LINK LINK-NAME=EDTSAM, FILE-NAME=SRV $nnn$ .LIBS, -
/  SUPPORT=*TAPE(FILE-SEQ=3), ACC-METH=*BY-CAT, -
/  BUF-LEN=*BY-CAT, REC-FORM=*BY-CAT, REC-SIZE=*BY-CAT
/START-EDT
@READ  ' / '
@SYSTEM 'REMOVE-FILE-LINK  EDTSAM'
@SYSTEM 'EXPORT-FILE  FILE-NAME=SRV $nnn$ .LIBS'
@WRITE  'SRV $nnn$ .LIBS'
@HALT
/ASS-SYSDTA  SRV $nnn$ .LIBS
/MOD-JOB-SW  ON=1
/START-PROG  $LMS
/MOD-JOB-SW  OFF=1
/ASS-SYSDTA  *PRIMARY

```

Where:

<tape-device> is the device-type of the tape, e.g. TAPE-C4

<volser> is the VOLSER of the tape (see **Report of Tape Creation**)

2. Copy the Procedure COPY.PROC from Tape to Disk

To copy the procedure COPY.PROC to disk, call the procedure P.COPYTAPE in the library SRV*nnn*.LIB:

```
/CALL-PROCEDURE (SRVnnn.LIB,P.COPYTAPE), -  
/ (VSNT=<volser>, DEVT=<tape-device>)
```

If you use a TAPE-C4 device, you may omit the parameter DEVT.

3. Copy all Product Files from Tape to Disk

To copy all Software AG product files from tape to disk, enter the procedure COPY.PROC:

```
/ENTER-PROCEDURE COPY.PROC, DEVT=<tape-device>
```

If you use a TAPE-C4 device, you may omit the parameter DEVT. The result of this procedure is written to the file 'L.REPORT.SRV'.

Installation Tape - VM/CMS

The installation tape contains the dataset listed in the table below.

Dataset Name	Contents
NOC nnn .TAPE	This dataset contains the Natural Optimizer Compiler load module.

The notation nnn in dataset names represents the version number of the product.

For a detailed description of the installation tape refer to the Report of Tape Creation which accompanies the tape.

Space Requirements

The space the dataset requires on disk is shown in the Report of Tape Creation.

Copying the Tape Contents to Disk

To copy the tape contents to disk

1. Position the tape for the TAPE LOAD command by calculating the number of tape marks as follows:
If the sequence number of NOC nnn .TAPE, as shown by the Report of Tape Creation, is n , you must position over $3n - 2$ tape marks (that is, FSF 1 for the first dataset, FSF 4 for the second, etc.)
2. Access the disk that is to contain the Natural installation files as disk **A**.
3. Ask the system operator to attach a tape drive to your virtual machine at the address X'181' and mount the Natural Optimizer Compiler installation tape.
4. When the tape has been attached, enter the following CMS command:
TAPE REW
Position the tape by entering the CMS command:
TAPE FSF n
where n is the number of tape marks and is calculated as described above ($3n - 2$).
5. Load the Natural Optimizer Compiler/CMS installation material by entering the CMS command:
TAPE LOAD * * A
Keep the tape drive attached to your virtual machine, because the tape is needed later in the installation procedure.

Installation Procedure

Step 1 - Modify the Natural Parameter Module - Jobs I060, I080

Activate the Natural Optimizer Compiler by adding the following macro to your Natural parameter module (NATPARM):

```
NTOPT ON
```

Assemble and link the parameter module.

Step 2 - Relink all Natural Nuclei - Jobs I060, I080

Adapt the link steps for Natural.

- **OS/390**

Add the following INCLUDE instruction to all links of the Natural nuclei (if you are using a shared nucleus, then include this statement in the link of the shared part):

```
INCLUDE NOCLIB(NOCNUC)
```

Add the corresponding DD statement:

```
//NOCLIB DD DSN=NOCnnn.LOAD,DISP=SHR
```

- **VSE/ESA**

Add the following INCLUDE instruction and the corresponding sublibrary for the Natural Optimizer Compiler in the search chain for the linkage editor:

```
INCLUDE NOCNUC
```

- **BS2000/OSD**

Add the following INCLUDE instruction to the element LNATSHAR in NATnnn.JOBS:

```
INCLUDE NOCNUC,NOCnnn.MOD
```

Relink your Natural nucleus as described in Step 5: Link the Natural Nucleus in Installing Natural under BS2000/OSD in the Natural Installation Guide for Mainframes.

- **VM/CMS**

The list of text files to be included in the Natural module or DCSS is contained in REXX program NAT\$LOAD EXEC (variable LOADLIST). To customize your Natural system, modify this EXEC with XEDIT by changing the LOADLIST as required.

Add the following INCLUDE instruction to the program NAT\$LOAD EXEC

```
LOADLIST = LOADLIST 'NOCNUC'
```

Relink your Natural nucleus with the procedure NATBLDM.

Installation Verification

1. Recatalog an existing program or write a new program and then catalog it.
2. Check the directory information for the program you have just cataloged, by using the LIST system command:
`LIST DIR object-name`

The directory information for the specified object will be displayed, showing the size of the machine code at the bottom of the screen.

Using the Optimizer Compiler - Overview

- What is Compiled and What is Not
- NOCSTAT Command
Which programs are suitable for compilation
- Displaying the Size of the Machine Code
- Optimizer Usage Examples

What is Compiled and What is Not

The Natural Optimizer Compiler is particularly effective for programs that contain a considerable amount of data manipulation, such as computation, transfer, and logical condition processing.

The Natural Optimizer Compiler compiles the following statements to machine code:

- assignment statements (ASSIGN and MOVE)
- RESET
- arithmetic statements (COMPUTE, ADD, SUBTRACT, MULTIPLY, DIVIDE)
- conditional statements (IF, DECIDE)
- control statements (FOR, REPEAT)
- ESCAPE
- COMPRESS
- EXAMINE
 - with the following clauses only:
GIVING NUMBER, GIVING POSITION or GIVING LENGTH (see also the Natural Statements documentation).
 - GIVING INDEX is not optimized.

Example:

```
EXAMINE #TEXT FOR #A GIVING NUMBER #NMB1
EXAMINE #TEXT FOR #A GIVING POSITION #POSEX5
EXAMINE #TEXT FOR #A GIVING LENGTH #LGHEX6
```

The Natural Optimizer Compiler **does not** compile the following statements:

- I/O statements (DISPLAY, WRITE, READ/WRITE WORK FILE).
- complex special statements like SEPARATE.
- statements that pass control to another programming object like FETCH, PERFORM, CALLNAT, CALL.
- statements that perform database access (READ, FIND, HISTOGRAM, GET, UPDATE, DELETE, END TRANSACTION, BACKOUT TRANSACTION)

Note:

The options the Natural Optimizer Compiler provides cannot be used for specifying statements to be optimized as described in the Optimizer Options.

NOCSTAT Command

For programs optimized with the Natural Optimizer Compiler, certain statements can be directly converted into machine code when cataloged. As a result, when executing the optimized objects with Natural at runtime, the performance can be improved considerably.

The NOCSTAT command analyses cataloged programming objects and provides statistical information to help decide whether program statements benefit from optimization with the NOC and, if so, to what extent they can be optimized.

If a program is cataloged (STOW, CATAL), the Natural compiler generates an internal (pseudo) object code based on the statements in the source program. In most cases, one source statement is transformed into one pseudo-code instruction. However, for complex statements, such as FOR and REPEAT, several pseudo-code instructions are generated. The NOCSTAT analyses are based on the generated pseudo-code instructions. Therefore, the number of statements indicated in the statistical reports may exceed the number of statements in the source program.

This section covers the following topics:

- Invoking NOCSTAT
 - Generating Reports
 - Report Formats
 - Batch Execution
-

Invoking NOCSTAT

▶ To invoke the NOCSTAT command

- Enter the direct command NOCSTAT.
The main NOCSTAT screen is displayed:

```

14:02:01          ***** NATURAL NOCSTAT COMMAND *****          2000-09-04

Name ..... _____
Library ..... SAGTEST_

NOCable Objects only .. _

Output Report ..... X Statement Category
                   _ Statement Type
                   _ Code Profile

Output Destination .... X Screen
                   _ CSV to Work File 1
                   _ XML to Work File 1
                   with XSL _____

Progress Control ..... X

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help           Exit                                     Canc
    
```

To obtain field-specific help information, either enter a question mark in the relevant field and press ENTER, or place the cursor in the field and press PF1. Press PF3 to exit NOCSTAT.

Generating Reports

You can generate statistical reports for a single program or a set of programs. If you analyze more than one program at a time, the reports are produced in series. When you have finished looking at one report, press ENTER to view the next report.

The main NOCSTAT menu provides the following options:

Field	Explanation												
Name	Enter a name or a range of names to specify the program(s) you want to examine: <table border="1" data-bbox="373 667 1027 931"> <tr> <td></td> <td><i>value</i> is any combination of one or more characters.</td> </tr> <tr> <td><i>value</i></td> <td>Single program.</td> </tr> <tr> <td>*</td> <td>All programs.</td> </tr> <tr> <td><i>value</i>*</td> <td>All programs whose names begin with <i>value</i>.</td> </tr> <tr> <td><i>value</i>></td> <td>All programs whose names are greater/equal <i>value</i>.</td> </tr> <tr> <td><i>value</i><</td> <td>All programs whose names are less/equal <i>value</i>.</td> </tr> </table>		<i>value</i> is any combination of one or more characters.	<i>value</i>	Single program.	*	All programs.	<i>value</i> *	All programs whose names begin with <i>value</i> .	<i>value</i> >	All programs whose names are greater/equal <i>value</i> .	<i>value</i> <	All programs whose names are less/equal <i>value</i> .
	<i>value</i> is any combination of one or more characters.												
<i>value</i>	Single program.												
*	All programs.												
<i>value</i> *	All programs whose names begin with <i>value</i> .												
<i>value</i> >	All programs whose names are greater/equal <i>value</i> .												
<i>value</i> <	All programs whose names are less/equal <i>value</i> .												
Library	Enter the name of a library or specify a range; the same applies as described for the Name field above. The current library is the default.												
NOCable Objects only	Mark this option to exclude programs already compiled with the NOC. Otherwise, the NOCSTAT command selects all Natural programs specified in the Name and Library fields by default, including NOC-compiled programs.												
Output Report	Mark any of the options to select statements by category, type or code profile. See Statement Category, Statement Type and Code Profile below.												

Field	Explanation
Output Destination	<p>Mark any of the following options to determine the output format and destination:</p> <p>Screen Online display.</p> <p>CSV to Work File 1 Generates spreadsheets with comma-separated values. Use the file extension ".csv" to write the work file directly to your PC for further processing.</p> <p> You can only route reports to a PC if Entire Connection is installed.</p> <p>XML to Work File 1 Generates XML documents. Use the file extension ".xml" to write the work file directly to your PC for further processing.</p> <p> If a value is entered in the field "with XSL", a processing instruction is added at the top of the XML output document:</p> <pre data-bbox="512 824 1193 853"><?xml-stylesheet type="text/xsl" href="value" ?></pre> <p> The <i>value</i> entered should be the absolute or relative URL of the style sheet, for example, "nocstat.xsl" or "http://natural.software-ag.de/nocstat.xsl".</p> <p> The processing instruction causes the document to be transformed according to the given style sheet when it is viewed by an XSLT-capable browser or transformed by a batch XSLT run. A typical use of this feature is to convert the output XML to an HTML page.</p> <p> There are two XSLT style sheets delivered with Natural as text members NOCSTLS1 and NOCSTLS2 in the Natural library SYSEXUEX on the FNAT system file.</p> <p> NOCSTLS1 provides formatting instructions for report type Statement Category, NOCSTLS2 for report type Statement Type as described below.</p> <p> Download the style sheets with file extension ".xsl" to the same directory in which the XML work files are stored.</p> <p> You can only route reports to a PC if Entire Connection is installed.</p> <p>See also Progress Control and Batch Execution below.</p>
Progress Control	<p>Only applies to Work File 1 output destinations.</p> <p>If this option is marked, a brief message appears online for each program listed in the report generated.</p>

Report Formats

You can choose between three output formats described below to display the statistics NOCSTAT provides for the statements analyzed. Different report layouts are produced for programs already optimized with the NOC and for programs to be considered for optimization. The example reports below show the difference. Press PF3 to interrupt report processing and return to the NOCSTAT menu.

Below is information on:

- Statement Category
- Statement Type
- Code Profile

Statement Category

The statistical report generated with the option "Statement Category" lists various categories of statements with the corresponding number of occurrences and the total number of statements already optimized or suitable for optimization, depending on whether or not the program was optimized with the NOC.

Example of NOC-Optimized Program:

```

14:07:17          ***** NATURAL NOCSTAT COMMAND *****          2000-09-04
                Library SAGTEST  Name NOCTEST1 Type Program

MCG Options: (ON,OVFLW,INDX,MIX,IO)

Database Loop:          0
Database Simple:        0
SORT / WORK I/O:       29
  FOR / REPEAT:         0
Screen / Printer:       59
String Manipulation:    6
  Arith / Logical:      0
  Program Calls:        3
Control Transfer:       49
  Block Start:          25
  Set Environment:       2
System Functions:       0
  Miscellaneous:        0

Total Statements:       949
  NOC optimized:        762 ( Ratio:  80 % )
  Longest NOC Run:      180 Statements

```

Example of Program without NOC Optimization:

14:13:01		***** NATURAL NOCSTAT COMMAND *****		2000-09-04
		Library SAGTEST Name NOCTEST2 Type Program		
	No NOC	NOCable		
	-----	-----		
Database Loop:	0	0		
Database Simple:	0	0		
SORT / WORK I/O:	0	0		
FOR / REPEAT:	0	5		
Screen / Printer:	57	0		
String Manipulation:	4	8		
Arith / Logical:	0	491		
Program Calls:	3	0		
Control Transfer:	19	69		
Block Start:	15	0		
Set Environment:	0	0		
System Functions:	0	0		
Miscellaneous:	0	0		
Total Statements:		672		
NOC optimizable:		573	(Ratio: 85 %)	
Longest NOC Run:		192	Statements	

Report Columns and Fields:

Column	Explanation
No NOC	Statements not suitable for optimization.
NOCable	Statements suitable for optimization.
Field	
Database Loop	The number of database statements that generate a processing loop, such as FIND and READ.
Database Simple	Database statements that do not generate a processing loop, such as STORE, UPDATE, DELETE and GET.
SORT / WORK I/O	SORT and WORK file statements.
FOR / REPEAT	Statements generating loops.
Screen / Printer	Screen and printer I/O, such as WRITE, DISPLAY and INPUT.
String Manipulation	String statements, such as EXAMINE and COMPRESS.
Arith / Logical	Arithmetic and logical statements, such as MOVE, COMPUTE and IF.
Program Calls	Transfer of control to a subroutine or subprogram, such as PERFORM, CALLNAT and FETCH.
Control Transfer	Jumps within the program, such as ESCAPE BOTTOM, FOR and REPEAT loops.
Block Start	Non-executed statements that demarcate code blocks, such as DEFINE SUBROUTINE and AT END. These statements are never optimized because they are never executed.
Set Environment	Statements that set the environment, such as SET CONTROL, SET GLOBALS and SET KEY.
System Functions	Statements, such as TOTAL, SUM, COUNT, MAX, MIN and *COUNT.
Miscellaneous	Pseudo-code statements not relevant for optimization and, therefore, ignored by the NOC.
Totals	
Total Statements	The total number of statements found in the program. This number may not correspond to the actual source statements as described in the introduction to NOCSTAT Command above.
NOC optimized	For an optimized program, these are the actual pseudo-code statements (as described in the introduction to NOCSTAT Command above) that have been NOC-optimized to machine code.
NOC optimizable	For non-optimized programs, this is the possible number of statements that could be optimized. The figure may be slightly higher than the actual number, since certain factors are not considered in the NOCSTAT program. For example, a SUBSTRING statement that has more than four arrays will be indicated as "optimizable" though it will not be optimized.
Ratio	Relation between Total Statements and NOC-optimized statements or Total Statements and NOC-optimizable statements in percent.
Longest NOC Run	NOC-optimized program: The number of contiguous optimized statements - the fewer fragment sequences, the better the performance.
	Non-optimized program: The number of contiguous statements to be expected if the program were optimized.

Statement Type

The statistical report generated with the option "Statement Type" lists single statements with the corresponding number of occurrences and the NOC coding generated for optimized objects.

Example of NOC-Optimized Program:

```

09:21:45          ***** NATURAL NOCSTAT COMMAND *****          2000-09-06
                   Library SAGTEST  Name NOCTEST1 Type Program

Statement          Number
-----
DB AT CONDITION           6
READ/WRITE WORK FILE    29
EXAMINE                   6
WRITE                     51
INPUT                     3
NEWPAGE                   2
REINPUT                   3
FIND                      1
READ                      2
NOC CODE                  760
BLOCK START              18
ON ERROR                  1
END                       1
STOP                      2
RETURN                   3
RETURN INLINE            15
ESCAPE ROUTINE           3
ESCAPE ROUTINE IMMEDIATE 1
MORE

```

Example of Program without NOC Optimization:

```

09:23:15          ***** NATURAL NOCSTAT COMMAND *****          2000-09-06
                   Library SAGTEST  Name NOCTEST2 Type Program

Statement          No NOC   NOCable
-----
DB AT CONDITION           6         0
MOVE/COMPUTE/ASSIGN      0       371
EXAMINE                   4         0
COMPRESS                  0         7
WRITE                     47        0
INPUT                     2         0
NEWPAGE                   2         0
REINPUT                   6         0
FIND                      1         0
READ                      1         0
HISTOGRAM                 1         0
ELSE/CLOSE LOOP          0       55
LOOPEND FOR/REPEAT       0         5
BLOCK START              8         0
ON ERROR                  1         0
END                       1         0
STOP                      2         0
RETURN                   2         0
MORE

```

Code Profile

The statistical report generated with the option "Code Profile" displays contiguous sequences of statements grouped by categories in a source program suitable for optimization, or lists the NOC coding generated for an optimized program. Occurrences are highlighted.

Example of NOC-Optimized Program:

```

09:59:04          ***** NATURAL NOCSTAT COMMAND *****                2000-09-06
                   Library SAGTEST  Name NOCTEST1 Type Program

Line  Statement
-----
0000  ON ERROR
0000  MCG OPTIONS
0045  MCG OPTIONS
0050  NOC CODE
0050  NOC CODE
0050  NOC CODE
0050  NOC CODE
1110  SET KEY
1140  NOC CODE
1140  NOC CODE
1145  NOC CODE
1145  NOC CODE
1150  NOC CODE
1150  NOC CODE
1155  NOC CODE
1155  NOC CODE
1160  NOC CODE
1160  NOC CODE
MORE
    
```

Example of Program without NOC Optimization:

```

10:01:36          ***** NATURAL NOCSTAT COMMAND *****                2000-09-06
                   Library SAGTEST  Name NOCTEST2 Type Program

Line  Statement
-----
0000  ON ERROR
0000  MCG OPTIONS
0100  MOVE/COMPUTE/ASSIGN          <-- NOCable
0100  MOVE/COMPUTE/ASSIGN          <-- NOCable
0100  MOVE/COMPUTE/ASSIGN          <-- NOCable
1920  MOVE/COMPUTE/ASSIGN          <-- NOCable
1920  FOR                          <-- NOCable
1920  MOVE/COMPUTE/ASSIGN          <-- NOCable
1920  FOR/REPEAT IF                <-- NOCable
1930  COMPRESS                     <-- NOCable
1940  LOOPEND FOR/REPEAT           <-- NOCable
1960  MOVE/COMPUTE/ASSIGN          <-- NOCable
1960  MOVE/COMPUTE/ASSIGN          <-- NOCable
1970  MOVE/COMPUTE/ASSIGN          <-- NOCable
1970  MOVE/COMPUTE/ASSIGN          <-- NOCable
1980  MOVE/COMPUTE/ASSIGN          <-- NOCable
1980  MOVE/COMPUTE/ASSIGN          <-- NOCable
1990  MOVE/COMPUTE/ASSIGN          <-- NOCable
MORE
    
```

Batch Execution

Below are job examples for processing NOCSTAT reports in batch mode. After job execution, the work files generated can be transferred from host to PC for further processing with standard transfer tools.

Example Job OS/390:

```
//NOCBATCH JOB (NOC,,30),CLASS=K,MSGCLASS=X 00000100
//NATEX EXEC PGM=NATBAT31,REGION=6200K,PARM=('IM=D') 00000200
//STEPLIB DD DISP=SHR,DSN=TESTNAT.LOAD 00000300
//CMPRINT DD SYSOUT=X 00000400
//CMWKF01 DD DSN='NOC.NOCSTAT.OUT',DISP=(NEW,CATLG), 00000500
           SPACE=(CYL,(1,1)),UNIT=SYSDA,VOL=SER=SAG001 00000600
//SYSOUT DD SYSOUT=X 00000700
//CMSYNIN DD * 00000800
NOCSTAT 00000900
*,library,X,,X 00001000
. 00001100
FIN 00001200
/* 00001300
```

Example Job VSE/ESA:

```
* $$ JOB JNM=NOCTST,CLASS=5,DISP=D
* $$ LST CLASS=Q,DISP=D
// JOB NOCTST
// ASSGN SYS001,DISK,VOL=xxxxxxx,SHR
// DLBL CMWKF01,'NOCSTAT.FILE.ONE',0
// EXTENT SYS001,xxxxxxx,1,0,1,150
// EXEC NAT234BA,SIZE=NAT314BA,PARM='SYSRDR'
IM=D,OBJIN=R
/*
ADARUN DBID=185
/*
NOCSTAT
*;library;X; ; ; ;X;
.
FIN
/*
/&
```

Example Job BS2000/OSD:

```

/.BAT234 LOGON NAT,1
/      SYSFILE SYSOUT=NAT314.OUT
/      SYSFILE SYSLST=NAT314.LST
/SKIP  .NOP000
=====
      NAME      : E.NAT314          S T A R T  B A T C H  N A T U R A L
=====
/.NOP000 REMARK
/      OPTION  DUMP=YES,MSG=FL
/      FILE    NOCSTAT.OUT,LINK=W01
/      FILE    ADAUSER ,LINK=DDCARD
/      FILE    $SAG.ADA623.MOD      ,LINK=BLSLIB00
/      SYSFILE TASKLIB=MOD234
/      SYSFILE SYSDTA=(SYSCMD)
/      FILE    NAT314.CMPRMIN,LINK=CMRMIN
/      DCLJV   NATJV1,LINK=*NATB2JV
/      FILE    $NAT.ADALNK.PARMS,LINK=DDLNKPAR
/      REMARK  %%%%%%%%%% BATCH-PHASE %%%%%%%%%%
/      EXEC    NAT314
NOCSTAT
*,ADE,X, , , ,X, , ,X
.
FIN

```

Displaying the Size of the Machine Code

With the Natural system command LIST, you can see whether a program has been compiled to machine code and also the size of the machine code.

To list compiled programs

- Enter in the command line: LIST DIR *object-name*

The directory information for the specified object will be displayed, showing at the bottom of the screen the size of the machine code, the OPT parameters used for the compilation and the NOC version under which the program was cataloged.

Details of the LIST command are provided in the Natural System Command Reference documentation.

Optimizer Usage Examples

The examples below illustrate when to use the Natural Optimizer Compiler to the best advantage and to give an indication of its power:

- Example 1 - No Improvement through Optimization
 - Example 2 - Considerable Improvement through Optimization
 - Examples 3 and 4 - CPU Usage depending on the NOC Options set
-

Example 1 - No Improvement

Nothing would be gained by using the Natural Optimizer Compiler for the following program:

```
DEFINE DATA LOCAL
  1 EMPLOYEES VIEW OF EMPLOYEES
    2 JOB-TITLE
    2 BIRTH
    2 NAME
  END-DEFINE
  FIND EMPLOYEES WITH JOB-TITLE = 'PROGRAMMER' OR = 'ANALYST'
                                OR = 'PROGRAMMER/ANALYST'
                                OR = 'SYSTEM ANALYST'
  DISPLAY JOB-TITLE BIRTH NAME
  END-FIND
  END
```

Example 2 - Considerable Improvement

If the following program is compiled with the Natural Optimizer Compiler, you will see a performance improvement of approximately 30 % (that is a 30 % reduction in CPU load). The program performs a statistical analysis of the age of IT-employees. Optimized statements are indicated in boldface.

In this example, NOC increases the object size by 20.5 %, due to 952 bytes of additional machine code:

	Size in Buffer Pool	Size of Machine Code generated by NOC
OPT=NODBG	5768	952
OPT=OFF	4784	0

```

DEFINE DATA
LOCAL
1 EMPLOY VIEW OF EMPLOYEES
  2 JOB-TITLE      (A25)
  2 BIRTH          (D)
  1 I              (I1)  INIT <1>
1 CDATE           (D)
1 NUMB            (N4)
1 SUMM            (P7.2)
1 SQUARE          (F8)
1 DEVI            (F8)
1 DEVIATION       (N3.4)
1 MEAN            (P2.3)
1 AGEDIS          (F8/1:70)
1 AGEMAX          (F8)
1 AGEH            (P3)
1 AGE             (P3)
1 AGEDAYS         (P15)
1 LINE            (A71/1:20)
1 REDEFINE LINE
  2 POINTS         (A1/1:20,0:70)
END-DEFINE
*
MOVE *DATX TO CDATE
*
FIND EMPLOY WITH JOB-TITLE = 'PROGRAMMER' OR = 'ANALYST'
OR = 'PROGRAMMER/ANALYST' OR = 'SYSTEM ANALYST'
  AGEDAYS:= CDATE - BIRTH
  AGE:=AGEDAYS / 365
  ADD 1 TO AGEDIS(AGE)                /* DISTRIBUTION
  ADD 1 TO NUMB
  ADD AGE TO SUMM
  COMPUTE SQUARE = SQUARE + AGE * AGE
END-FIND
*
*****
* COMPUTE ESTIMATES
*****
*
COMPUTE DEVI = NUMB * SQUARE / (SUMM * SUMM) - 1
COMPUTE DEVIATION = SQRT(DEVI)
COMPUTE MEAN = SUMM / NUMB
*
*****
* GRAPHIC DISPLAY
*****

```

```

*
FOR I 1 70
  IF AGEDIS(I) > AGEMAX MOVE AGEDIS(I) TO AGEMAX
  END-IF
END-FOR
FOR I 1 70
  COMPUTE AGEDIS(I) = AGEDIS(I) * 20 / AGEMAX
END-FOR
FOR I 1 70
  COMPUTE AGEH = 21 - AGEDIS(I)
  IF AGEH < 21 MOVE '*' TO POINTS(AGEH:20,I)
  END-IF
END-FOR
*
*****
* COMPLETE GRAPHIC DISPLAY
*****
*
MOVE '!' TO POINTS(*,0)
WRITE TITLE LEFT
  AGEMAX(EM=999) 20X 'DISTRIBUTION OF IT-EMPLOYEES BY AGE'
WRITE NOTITLE NOHDR
LINE(*) /
'0-----10-----20-----30-----40-----50-----60-----'
/ 'MEAN=' MEAN 10X 'DEVIATION=' DEVIATION
END

```

Examples 3 and 4 - CPU Usage

The following program illustrates the difference in CPU usage, depending on the options you select when compiling the program. The table below lists the CPU usage in seconds and percent. The figures provided in the table were determined during a test run in an IBM OS/390 environment. They can only serve as general orientation, since absolute values vary depending on the hardware applied.

```

DEFINE DATA LOCAL
1 #I1      (I4) INIT <1>
1 #I2      (I4) INIT <2>
1 #J1      (I4) INIT <3>
1 #J2      (I4) INIT <4>
1 #F       (I4)
1 #ARR1    (N7/10,5)
1 #ARR2    (N5/10,5)
END-DEFINE
*
FOR #F = 1 TO 1000000
  MOVE #ARR1(#I1,#I2) TO #ARR2(#J1,#J2)
END-FOR
*
END

```

Option	CPU seconds	CPU percentage
OFF	8.78	100
ON	0.63	7.18
INDX	0.85	9.68
OVFLW	1.71	19.48
INDX,OVFLW	2.00	22.78
INDX,OVFLW,NODBG	1.61	18.34
INDX,OVFLW,NODBG,NOSGNTR	1.61	18.34
NODBG	0.44	5.01
NOSGNTR	0.63	7.18
NODBG,NOSGNTR	0.44	5.01

```

DEFINE DATA LOCAL
1 #I1      (P7) INIT <1>
1 #I2      (P7) INIT <2>
1 #J1      (N7) INIT <3>
1 #J2      (N7) INIT <4>
1 #K1      (I4) INIT <5>
1 #K2      (I4) INIT <6>
1 #F       (I4)
1 #FIELD1  (P5)
1 #FIELD2  (N5)
1 #FIELD3  (I2)
END-DEFINE
*
FOR #F = 1 TO 500000
*
#FIELD1:= #I1 - #I2 + (13 * 10 / 5)
#FIELD2:= #J1 - #J2 + (13 * 10 / 5)
#FIELD3:= #K1 - #K2 + (13 * 10 / 5)
*
END-FOR
*
END
    
```

Option	CPU seconds	CPU percentage
OFF	18.61	100.00
ON	4.95	26.60
INDX	4.95	26.60
OVFLW	5.38	28.91
INDX,OVFLW	5.38	28.91
INDX,OVFLW,NODBG	5.26	28.26
INDX,OVFLW,NODBG,NOSGNTR	5.09	27.35
NODBG	4.79	25.74
NOSGNTR	4.81	25.85
NODBG,NOSGNTR	4.63	24.88
NODBG,NOSGNTR,ZD=OFF	4.51	24.23
NODBG,NOSGNTR,ZD=OFF,SIGNCHCK=OFF	4.41	23.70

Activating the Optimizer Compiler

To activate the Natural Optimizer Compiler, use one of the following:

- Macro NTOPT
- Dynamic Profile Parameter OPT
- System Command NOCOPT
- Natural Statement OPTIONS

where the first alternative is the most static one and the last alternative the most dynamic one.

All alternatives use the Optimizer options as described in the section Optimizer Options. Using these options you can control how and when machine code is generated, what tracing options are to be used and what the target architecture will be. The Optimizer options are the only control mechanism for the Natural Optimizer Compiler.

Macro NTOPT

With the macro NTOPT in the Natural Parameter Module, you can activate the Natural Optimizer Compiler statically for a linked Natural nucleus. Every time this Natural nucleus is started, the same Optimizer options are used again.

Example 1:

```
NTOPT 'INDX,OVFLW,ZD=OFF'
```

Example 2:

```
NTOPT 'INDX,OVFLW,ZD=OFF,TRGPT',
      'TRSTMT,OPTLEV03'
```

Note the continuation character "-" in column 72.

See the section Optimizer Options for an explanation of the options setting used.

Dynamic Profile Parameter OPT

When starting a Natural session, you can dynamically activate the Optimizer Compiler by specifying the Natural profile parameter OPT. As a synonym for OPT, you can use MCG. The specification of the parameter module is overwritten. The options are only valid for the current session.

Example:

```
OPT=(INDX,OVFLW,ZD=OFF)
```

or

```
MCG=(INDX,OVFLW,ZD=OFF)
```

See the section Optimizer Options for an explanation of the option setting used.

System Command NOCOPT

When you have started a Natural session, you can invoke the Optimizer command screen with the Natural system command NOCOPT. The screen monitors the current setting of the Natural Optimizer Compiler options as they were specified during Natural startup. You can now modify the setting online.

The updated parameter setting is only valid for the current session.

Natural Statement OPTIONS

The MCG parameter of the Natural compiler statement OPTIONS provides the most flexible and powerful control over machine code generation, since different options can be set for individual statements in a program. So, within one Natural program, the Optimizer can be activated and deactivated several times to enclose ranges of statements with different options settings.

Example

```
OPTIONS MCG=(OVFLW,INDX,ZD=OFF)
```

or

```
OPTIONS MCG=OVFLW,INDX,ZD=OFF
```

The options string of the MCG parameter may start with a plus (+) or minus (-) sign, indicating that the values of options not mentioned should be left unaltered, and only the options present should be set (+) or reset (-), for example:

Example:

```
OPTIONS MCG=+PGEN          /* turns tracing on
```

```
(statements to be traced)
```

```
OPTIONS MCG=-PGEN         /* turns tracing off
```

If the string starts with anything other than "+" or "-", all options are reset before the string is parsed.

Note that the Natural statement OPTIONS also provides other Natural compiler parameters than MCG.

See the section Optimizer Options and for an explanation of the options setting used.

Optimizer Options

When the Natural Optimizer has been activated, you can specify checks by setting the options explained in this section.

The options cannot be used for specifying statements to be optimized.

This section covers the following topics:

- List of Options
- PGEN Option
- Influence of other Natural Parameters

List of Options

The following table lists and describes NOC options. Default values are underlined (this is the value that will be assumed if the option is not present).

A NOC option consists of a string surrounded by brackets or single quotation marks (except in the Natural OPTIONS statement), with options separated by commas. Some options have values, while the very existence of some options in the option string is sufficient to modify the environment.

Optional clauses are surrounded by square brackets [], choices are surrounded by curly braces { } and each choice is separated by vertical lines "|". Only one of these choices can be specified; ON is equivalent to Y (Yes), OFF to N (No). Options specified without the optional clause ON or OFF (if applicable), or their equivalent values, are interpreted as set to ON. For example, OVFLW is identical to OVFLW=ON. Except for the option OFF, any specified option switches on optimizing (as if ON was specified) and the default values apply. For example, INDEX is identical to ON,INDEX.

Option	Explanation
ABEND	Forces the Natural Optimizer Compiler to generate code which causes Natural to be abnormally terminated immediately when the ABEND option is encountered by the Natural Optimizer Compiler during compilation. The option must appear by itself or it will be ignored. Other parameters are not changed or reset by this option. This option can be useful for debugging purposes.
CACHE[={ON <u>OFF</u> Y N}]	Switches variable caching on or off. See also Variable Caching in the section Performance Considerations.
CPU= <u>/370</u>	Specifies the target architecture. The /370 option is valid for IBM and Siemens.
DIGTCHCK[={ON <u>OFF</u> Y N}]	Specifies whether the digits of packed and unpacked numeric fields (Formats P and N) are to be checked when moving to another variable of the same type and precision. For example, if DIGTCHCK is ON and an unpacked numeric variable (Format N) contains an invalid digit, such as X'FA', moving to another unpacked numeric variable with the same precision will generate a SOC7 (or NAT0954) error. If DIGTCHCK is OFF, no error is generated but the generated code is much faster.
ERRDUMP[={ON <u>OFF</u> Y N}]	Specifies whether NOC should abend if an error condition is detected during the compile phase. This is useful for debugging the Natural Optimizer Compiler itself.

Option	Explanation
INDEX[={ON OFF Y N}]	Specifies whether array indexes will be checked for out-of-bound values in the optimized code. See also the Warning below.
INDX[={ON OFF Y N}]	Specifies whether array indexes will be checked for out-of-bound values in the optimized code. Additionally, RANGE will be set on. Therefore, this option is equivalent to INDEX=ON,RANGE=ON. See also the Warning below.
IO[={ON OFF Y N}]	Reserved for future use.
LOOPS[={ON OFF Y N}]	Provided for compatibility reasons only. No effect.
MIX[={ON OFF Y N}]	Provided for compatibility reasons only. No effect.
NODBG[={ON OFF Y N}]	If NODBG=OFF/N (default), the Natural Debugger can be used to debug optimized code (then, additional code is generated to check whether TEST mode has been set on). If NODBG=ON/Y, it is not possible to use the Natural Debugger to debug optimized programs. The program will also run faster and consume less CPU time. NODBG is also in effect if you specify the options INDEX, RANGE or OVFLW. See also NODBG in the section Performance Considerations.
NOSGNTR[={ON OFF Y N}]	Applies to packed numbers only. If NOSGNTR=OFF (default), signs of positive packed numbers which are the result of an arithmetic operation or the target of an assignment are set according to the COMPOPT parameter PSIGNF. If NOSGNTR=ON, the signs resulting from execution of the generated machine instruction are left unchanged. See also the section Influence of other Natural Parameters.
ON	Switches on optimizing. If no additional option is specified, the default value defined for each option is in effect. As indicated in the Warning below, this may cause unintended results, in particular regarding the options INDEX, INDX, OVFLW, and RANGE.
OFF	Switches off optimizing.
OPTLEV={2 3}	Specifies optimization level - roughly equivalent to the number of passes through the program. OPTLEV=3 is useful when PGEN is specified, since some branch targets cannot be determined during the first pass and PGEN output is made during the last pass. Thus, some values may be shown improperly.
OVFLW[={ON OFF Y N}]	Specifies whether checks for overflow in arithmetic operations or assignments will be included in the optimized code. See also the Warning below.
PGEN[={ON OFF Y N}]	Specifies whether a disassembly of the optimized code should be output. This option also enables all other tracing options. See also PGEN Option below.
RANGE[={ON OFF Y N}]	Specifies whether range checks will be performed in operations with arrays. This ensures that array ranges will have an equal number of elements in corresponding dimensions of all operands. See also the Warning below.

Option	Explanation
SIGNCHCK[={ON OFF Y N}]	Specifies whether the result of a multiplication with a packed or unpacked numeric multiplier should be checked for a negative zero. If zero is multiplied by a negative number, the MP machine instruction generates a negative zero result. If SIGNCHCK is on, this negative zero is converted to a positive zero. The check for a negative zero is done for every multiplication with a packed or unpacked numeric multiplier.
TRENTRY	For internal use by Software AG only. Do not change the setting of this parameter.
ZD[={ON OFF Y N}]	Specifies whether divisors should be checked for zero. If this option is specified, then code is inserted, so that the program behaves according to the ZD profile parameter of Natural, that is, Natural error NAT1302 is issued or the result is zero. If this option is not specified, Natural error NAT0954 occurs if the divisor is zero. See also ZD - Zero-Division Check in the section Profile Parameters in the Natural Parameter Reference documentation.



Warning for INDEX, INDX, OVFLW, and RANGE:

Apply values OFF and N with care. Suppressing overflow checking or array index checking may allow incorrect programs to lead to unpredictable results, storage corruption, or abnormal terminating.

See also the Example of INDEX and OVFLW below which demonstrates the impact of INDEX and OVFLW.

Below is information on:

- Example of INDEX and OVFLW
- Optimum Code Generation

Example of INDEX and OVFLW

```

DEFINE DATA LOCAL
...
1 P1 (P1/9)
...
1 P3 (P3/9)
...
1 I (I4)
1 J (I4)
1 K (I4)
1 L (I4)
END-DEFINE
...
P1(I:J) := P3(K:L)
...
END

```

Explanation of Example

With INDX=ON or INDEX=ON set, code is generated to verify that **I**, **J**, **K** and **L** are within the ranges defined for **P1** and **P3** respectively.

With INDX=ON or RANGE=ON set, code is generated to verify that **I:J** and **K:L** denote ranges of the same length.

With OVFLW=ON set, code is generated to verify that the value of P3 fits into the corresponding P1 variable. For example: Value 100 would cause an overflow here.

Example Error Situation:

If one of the occurrences of P3 contains the value 100, with OVFLW=OFF set, the value assigned to the corresponding P1 occurrence will be zero. If the index variable **I** is zero or greater than 9, with INDX=OFF set, storage areas that do not belong to Array P1 will be corrupted. If these options (OVFLW and INDX) are set to ON, a Natural error occurs like it does in standard Natural runtime.

For the NOC option specified above, additional code is generated. However, this is well compensated for by the advantage of a check that, for example, protects against hard-to-debug errors. Undetected errors can, of course, lead to unpredictable results.

Optimum Code Generation

To assure that the least amount of code is generated and thus achieve optimum performance, use OPT='NODBG,NOSGNTR,SIGNCHCK=OFF,ZD=OFF'. However, only apply this setting to programming objects that have been thoroughly debugged: see also the Warning below.

PGEN Option

The PGEN option causes the Natural Optimizer Compiler to output the generated code and internal Natural structures. Thus, code and structures can be examined, for example, for bug fixing, performance review and support issues.

An understanding of IBM's /370 assembler is required to interpret the results produced by the PGEN option.

We recommend that you use this option with the assistance of your local Software AG representative.

Below is information on:

- Setting PGEN
- Sub-Options of the PGEN Option
- Output of the PGEN Option
- Working with the PGEN Output

Setting PGEN

To use the PGEN facility, set the PGEN option when activating on the Optimizer Compiler.

Since the buffer is kept in memory, it is possible that the user thread will not be big enough to hold the trace information. In this case, try setting PGEN on only for the portion of the program which is to be traced, for example:

OPTIONS MCG=(PGEN=ON , TRGPT=ON) or OPTIONS MCG=+PGEN , TRGPT	Turns tracing on, including tracing of the GPT entries
OPTIONS MCG=(PGEN=OFF) or OPTIONS MCG=-PGEN	Turns tracing off

Various options affect the content of the output. The basic PGEN option causes a formatted listing of Natural source lines and a disassembly of the corresponding code to be generated and kept in memory for extraction by the NOCSHOW utility as described below, under Output of the PGEN Option.

The TRSTMT, TRGPT, TRMPT and TRVDT options cause hex dumps of internal data structures associated with each line to be output.

The TRBASES and TRCACHE options cause information on base registers and cache variables to be printed out.

Sub-Options of the PGEN Option

The following table describes the options when PGEN=ON. For an explanation of the syntax used see the introduction to List of Options above.

Option	Explanation
LPP={5 .. 55 .. 255}	Lines-per-page for the trace output, only used when TREXT=ON.
NOSRCE[={ON OFF Y N}]	If NOSRCE=OFF, the Natural source statement is included in the output.
TRACELEV={0 .. 255}	Specifies the trace level. Each bit in this one byte value specifies a buffer type to trace; these bits can be set on by using the TRxxx options as well.
TRBASES[={ON OFF Y N}]	Specifies whether base register allocations are traced.
TRCACHE[={ON OFF Y N}]	Specifies whether CACHE entries are traced.
TREXT[={ON OFF Y N}]	If TREXT=ON, trace is directed to the user exit NOCPRINT as described below.
TRGPT[={ON OFF Y N}]	Specifies whether GPT entries are traced.
TRMPT[={ON OFF Y N}]	Specifies whether MPT entries are traced.
TRSTMT[={ON OFF Y N}]	Specifies whether STMT entries are traced.
TRVDT[={ON OFF Y N}]	Specifies whether VDT entries are traced.

See also the examples below.

Output of the PGEN Option

There are two places to where the Natural Optimizer Compiler can direct the output of PGEN:

- Internal Buffer
- User Exit NOCPRINT

Internal Buffer

The contents of this buffer is overwritten each time a CHECK, CAT, STOW or RUN command is executed. A system utility NOCSHOW is provided whereby the contents of this buffer can be viewed, searched or printed.

To invoke the NOCSHOW utility

- Enter the direct command NOCSHOW after a CHECK, STOW, CAT or RUN where the Natural Optimizer Compiler has been active.

The following PF keys are available on the screen:

Key	Function
PF2	Position to top of output
PF4	Position one line backward
PF5	Position one line forward
PF6	Print to Natural printer support No.1
PF7	Position one page backward
PF8	Position one page forward
PF10	Scan for text string
PF11	Repeat scan

User Exit NOCPRINT

If TREXT=ON is specified, the Natural Optimizer Compiler passes every output line to the user exit NOCPRINT instead of adding it to the trace buffer.

NOCPRINT is invoked following normal OS register conventions. Register 1 points to a full word containing the address of the 81 byte print line with ANSI carriage control characters in position 1. Register 13 points to an area of 18*4 bytes which may be used as a save area. Register 14 contains the return address and Register 15 contains the entry address of NOCPRINT.

The user exit NOCPRINT can be written in any language which supports the register conventions described above. It must be linked to the Natural nucleus together with the Natural Optimizer Compiler nucleus.

Working with the PGEN Output

This section provides hints and explanations on how to interpret the output created with the PGEN option.

- At the top of the PGEN output are some disassembled lines which do not appear to belong to any source line. These are the instructions which make up the prologue, which is executed whenever control passes from non-optimized to optimized code. Permanent base registers are loaded and control is passed to the correct point in the prologue.
See Example Section A below.
- Sometimes a lot of source lines are printed without any code. This is because the Natural compiler puts a single line number in the object of statements which may span more than one line.
See Example Section B below.
- If the NODBG=OFF (default) has been specified, a sequence of instructions is generated at the start of each Natural statement:

```
BALR R9,R11
DC X'.....'
```

This sequence sets the line number (in case of error) and checks whether the TEST mode is switched ON. Without this sequence, debugging of NOC-compiled statements by the Natural Debugger is not possible. See Example Section C below.

- Sometimes there is a line break between disassembled lines. This break indicates an internal statement separation. It happens because often a single Natural statement will generate multiple internal (pseudo-code) statements.

Example Section A:

```

000000 5880 D354      L      R8,RTADR+4
000004 5870 D370      L      R7,RTADR+32
000008 4810 6006      LH     R1,6(,R6)
00000C 1F60           SLR   R6,R0
00000E 47F1 A000      BC     15,0(R1,R10)

```

Example Section B:

```

0010 OPTIONS MCG=(PGEN=ON,TRGPT=ON)
0020 DEFINE DATA LOCAL
0030 1 I(I4)
0040 1 P(P7.2)
0050 1 T(P7.2)
0060 END-DEFINE
0070 *

0080 SETTIME
0090 *

000012 45E0 B040      BALR  R14,RETH
000016 0036          DC    X'0036'

0100 FOR I=1 TO 100000

```

Example Section C:

```

000018 059B          BALR  R9,R11
00001A 003E          DC    X'003E'
00001C D203 7000 833B  MVC    I,#VAR033B

000022 059B          BALR  R9,R11
000024 004C          DC    X'004C'
000026 47F0 A040      BC     15,64(,R10)

00002A 059B          BALR  R9,R11
00002C 005A          DC    X'005A'
00002E BFFF 8343      ICM   R15,15,#VAR0343
000032 BF0F 7000      ICM   R0,15,I
000036 1A0F          AR    R0,R15
000038 BE0F 7000      STCM  R0,15,I

00003C 059B          BALR  R9,R11
00003E 006C          DC    X'006C'
000040 BFFF 833F      ICM   R15,15,#VAR033F
000044 BF0F 7000      ICM   R0,15,I
000048 190F          CR    R0,R15
00004A 4720 A066      BC     2,102(,R10)

0110 ADD 1.00 TO P

00004E 059B          BALR  R9,R11
000050 0082          DC    X'0082'
000052 FA41 7004 8347  AP    P,#VAR0347
000058 DC00 7008 B488  TR    P+4(1),PSGNTR

0120 END-FOR
0130 *

00005E 059B          BALR  R9,R11
000060 0094          DC    X'0094'
000062 47F0 A02A      BC     15,42(,R10)

```

```

0140 T:=*TIMD(0080)

000066 059B          BALR  R9,R11
000068 009C          DC    X'009C'
00006A 45E0 B0D8     BAL   R14,SYSFUNC
00006E 0330 B881     DC    X'0330B881'

000072 F246 7009 8330  PACK  T,#VAR0330
000078 F040 7009 0002  SRP   T,2,0
00007E DC00 700D B488  TR    T+4(1),PSGNTR

0150 T:=T / 10
0160 *

000084 059B          BALR  R9,R11
000086 00AE          DC    X'00AE'
000088 F864 D100 7009  ZAP   OP1(7),T
00008E F811 D130 8349  ZAP   WORK2(2),#VAR0349
000094 45E0 B104     BAL   R14,ZDCHECK
000098 F240 7009 B355  PACK  T,ZEROZ
00009E 47F0 E01C     BC    15,28(,R14)
0000A2 FD61 D100 8349  DP    OP1(7),#VAR0349
0000A8 D204 7009 D100  MVC   T,OP1
0000AE DC00 700D B488  TR    T+4(1),PSGNTR

0170 DISPLAY 'ELAPSED TIME (S)' T

0000B4 45E0 B040     BAL   R14,RETH
0000B8 00C0          DC    X'00C0'

0180 END
    
```

Influence of other Natural Parameters

The global parameter ZD influences the behavior of the NOC compiler. See the description of the ZD option as described under List of Options above.

The COMPOPT parameter PSIGNF (see also the system command COMPOPT in the Natural System Command Reference documentation) influences the behavior by forcing the signs of positive packed decimal numbers to **F** if ON, and to **C** if OFF. The parameter is applied if NOSGNTR=OFF is specified.

See the chart below for packed data (Type P):

NOSGNTR=OFF	and	PSIGNF=ON	All signs are normalized to F (default).
NOSGNTR=OFF	and	PSIGNF=OFF	All signs are normalized to C.
NOSGNTR=ON			All signs are left as they were generated by the last operation.

For numeric data (Type N) the signs are always normalized to F, regardless of the settings of NOSGNTR and PSIGNF.

Performance Considerations

This section covers the following topics:

- Formats
 - Arrays
 - Alphanumeric Fields
 - DECIDE ON
 - Numeric Values
 - Variable Positioning
 - Variable Caching
 - NODBG
-

Formats

Best performance is achieved when you use the data formats packed numeric (P) and integer (I4) in arithmetic operations.

Avoid converting data between the formats packed numeric (P), unpacked numeric (N), integer (I), and floating point (F), as this causes processing overhead even with optimized code.

As there is no interpretation overhead with optimized code, the differences between the various data formats become much more prominent: with optimized code the performance improvement gained by using Format P instead of N, for example, is even higher than with normal code.

Example:

```
A = A + 1
```

In the above numeric calculation

- with non-optimized code, Format P executes approximately 13 % faster than Format N.
- with optimized code, however, Format P executes approximately 56 % faster than Format N.

The performance gain which would be achieved by applying the Natural Optimizer Compiler to this simple statement is

- with unpacked operands (N): 8 times faster
- with packed operands (P): 15 times faster

Arrays

Array range operations, such as

```
MOVE A(*) TO B(*)
```

are executed more efficiently than if the same function were programmed using a FOR statement processing loop. This is also true for optimized code.

When indexes are used, integer Format I4 should be used to achieve optimum performance.

Alphanumeric Fields

We recommend that you adjust the length of the alphanumeric constant to the length of the variable, when moving an alphanumeric constant to an alphanumeric variable (Format A), or when comparing an alphanumeric variable with an alphanumeric constant. This will significantly speed up operation, for example:

```
A(A5) := 'XYZAB' ...
IF A = 'ABC ' THEN ...
```

is faster than

```
IF A = 'ABC' THEN ...
```

DECIDE ON

When using DECIDE ON with a system variable, array or parameter *operand1*, it is more efficient to move the value to a scalar variable of the same type and length defined in the LOCAL storage section.

Numeric Values

When using numeric constants in assignments or arithmetic operations, try to force the constants to have the same type as the operation.

Rules of Thumb

- Any numeric constant with or without a decimal but without an exponent is compiled to a packed number having the minimum length and precision to represent the value, unless the constant is an array index or substring starting position or length, in which case it becomes a four-byte integer (I4). This rule applies irrespective of the variable types participating in the operation.
- Operations containing floating point will be executed in floating point. Add **E00** to numeric values to force them to be floating point, for example:

```
ADD 1E00 TO F(F8)
```

- Operations not containing floating point, but containing packed numeric, unpacked numeric, date or time variables will be executed in packed decimal. For ADD, SUBTRACT and IF, force numeric constants to have the same number of decimal places as the variable with the highest precision by adding a decimal place and trailing zeros, for example:

```
ADD 1.00 TO P(P7.2)
```

This technique is unnecessary for MULTIPLY and DIVIDE.

Variable Positioning

To ease the optimization process, try to keep all scalar references at the front of the data section and all array references at the end of the data section.

Variable Caching

The Natural Optimizer Compiler contains an algorithm to enhance the performance even further. In terms of performance, a statement will differ depending on the types of operands. The statement will execute more slowly if one or more of the operands is a parameter, array or scalar field of Type N (numeric) or combinations

of these operands. The NOC analyzes the program flow and determines which variables with one or more of these characteristics are read two or more times without being written to. It then moves the value of each variable to a temporary cache area where it can be accessed quickly under the following conditions:

- The variable is accessed often but seldom modified **and**
- The variable is an array of any type or a scalar field of Type N (numeric).

Most suitable for variable caching are programs with long sequences that repeatedly access the same variable, in particular if the variable is an array. Variable caching then avoids complex and recurring address computation.

Example of Variable Caching

The example program displayed below demonstrates the advantage of variable caching. Cataloged with NODBG (see below) and CACHE=ON, executing this program in a test environment took 47 % of the time required to execute the program with NODBG and CACHE=OFF. Cataloging the program with CACHE=ON, reduces the code generated by the NOC from 856 bytes to 376 bytes.

```

DEFINE DATA LOCAL
1 ARR(N2/10,10,10)
1 I(I4) INIT <5>
1 J(I4) INIT <6>
1 K(I4) INIT <7>
END-DEFINE
DECIDE ON EVERY ARR(I,J,K)
VALUE 10 IGNORE
VALUE 20 IGNORE
VALUE 30 IGNORE
VALUE 40 IGNORE
VALUE 50 IGNORE
VALUE 60 IGNORE
VALUE 70 IGNORE
VALUE 80 IGNORE
VALUE 90 IGNORE
NONE IGNORE
END-DECIDE

```



Warning:

If the content of a cached variable is modified with the command MODIFY VARIABLE of the Natural Debugger, only the content of the original variable is modified. The cached value (which may still be used in subsequent statements) remains unchanged. Therefore, variable caching should be used with great care if the Natural Debugger is used. See also the Natural Debugger documentation.

NODBG

Once a program has been thoroughly tested and put into production, you should catalog the program with the NODBG option as described in the section Optimizer Options. Without debug code, the optimized statements will execute from 10% to 30% faster.

The code to facilitate debugging is removed when this option is specified, even with INDX or OVFLW options turned on.

Listing Zaps

If you want to have an overview of the Zaps that have been applied to the Natural Optimizer Compiler at your site, use the DUMP system command.

 **To obtain a Zap overview**

- In the command line, enter
DUMP ZAPS NOC.
A list of the Zaps that have been applied is displayed.

If no Zaps have been applied to the Natural Optimizer Compiler, you will receive the appropriate message.