engineering
mannesmann
Rexroth

CNC
NC Compiler
V17

Application Manual

DOK-MT*CNC-NC*COMP*V17-ANW1-EN-P

276772

Indramat

Title NC Compiler

Type of document Functional description

Documentation type DOK-MT*CNC-NC*COMP*V17-ANW1-EN-P

Internal filing
- Folder 5
- Drawing number 109-0768-4183-EN/06.97

Purpose of the document This document is used for explaining the following compiler functions:
- Chamfers and roundings
- Graphical NC editor (for contour and machining programming)
- Macro technique
- Modal function

Furthermore, it provides the information that is required for handling the compiler; and describes the integration and generation of a user compiler.

Configuration control

Documentation identification of previous releases	Release date	Comment
109-0768-4183-00	06/97	New issue

Publisher INDRAMAT GmbH • Bgm.-Dr.-Nebel-Str. 2 • D-97816 Lohr a. Main

Phone +49 (0)9352/40-0 • Tx 689421 • Fax +49 (0)9352/40-4885

Dept. ENC (GL)

Contents

1 Basics

NC-Compiler

From software version 5.17 onwards, the NC compiler has been integrated into the user interface. It permits NC programs to be pre-compiled. Using these features, the following functions have been implemented:

- Chamfers and roundings,
- enhanced look-ahead function,
- graphical NC editor (for contour and machining programming),
- macro technique, and
- modal function.

Source program

- The programmer creates a program that is referred to as the 'source program'.
- The source block number is used for numbering the NC blocks.
- The source block number consists of a preceding letter 'N' and a four-digit block number.
- Example: *Insert chamfer*

```
      :
N0053  G1 Z... X... CF...          ;insert chamfer
N0054  G1 Z... X...
      :
```

Compiled program (object file)

- The program that is produced by the compiler is referred to as 'compiled program' or as 'object file'.
- The user can neither view nor modify such a compiled program.
- The NC blocks are numbered using the source block number and the enhanced block number.
- The enhanced block number consists of a four-digit number, that is separated from the source block number by a colon.
- Example: *Insert chamfer*

  The compiler modifies block N0053, inserts an intermediate block (N0053:0001), and modifies block N0054.

```
      :
N0053:0000  G1 Z... X...           ;insert chamfer
N0053:0001  G1 Z... X...
N0054:0000  G1 Z... X...
      :
```

Runtime program

- A 'runtime program' is a compiled program that is loaded into the controller. During tool path compensation, the CPU inserts additional (marked by an arrowhead) intermediate blocks in that program.
- The runtime program only exists during program execution.
- The NC employs the program as the basis of the path movements that are to be performed.
- The user may view the runtime program, but cannot edit it.
- Example: *Insert chamfer*

```
      :
N0053:0000  G1 Z... X...
→           G2 Z... X... I... K...
```

```
N0053:0001  G1 Z... X...
→           G2 Z... X... I... K...
N0054:0000  G1 Z... X...
   :
```

2 Chamfers and Roundings

Chamfers and roundings

The commands

- CF (insert chamfer) and
- RD (insert rounding)

enable chamfers and roundings to be inserted.

Syntax

CF.. or CF=... ;insert chamfer (chamfer)

RD.. or RD=... ;insert rounding (round)

Explanation

- A further linear contour (chamfer) or an arc (rounding) can be inserted between linear or circular contours.

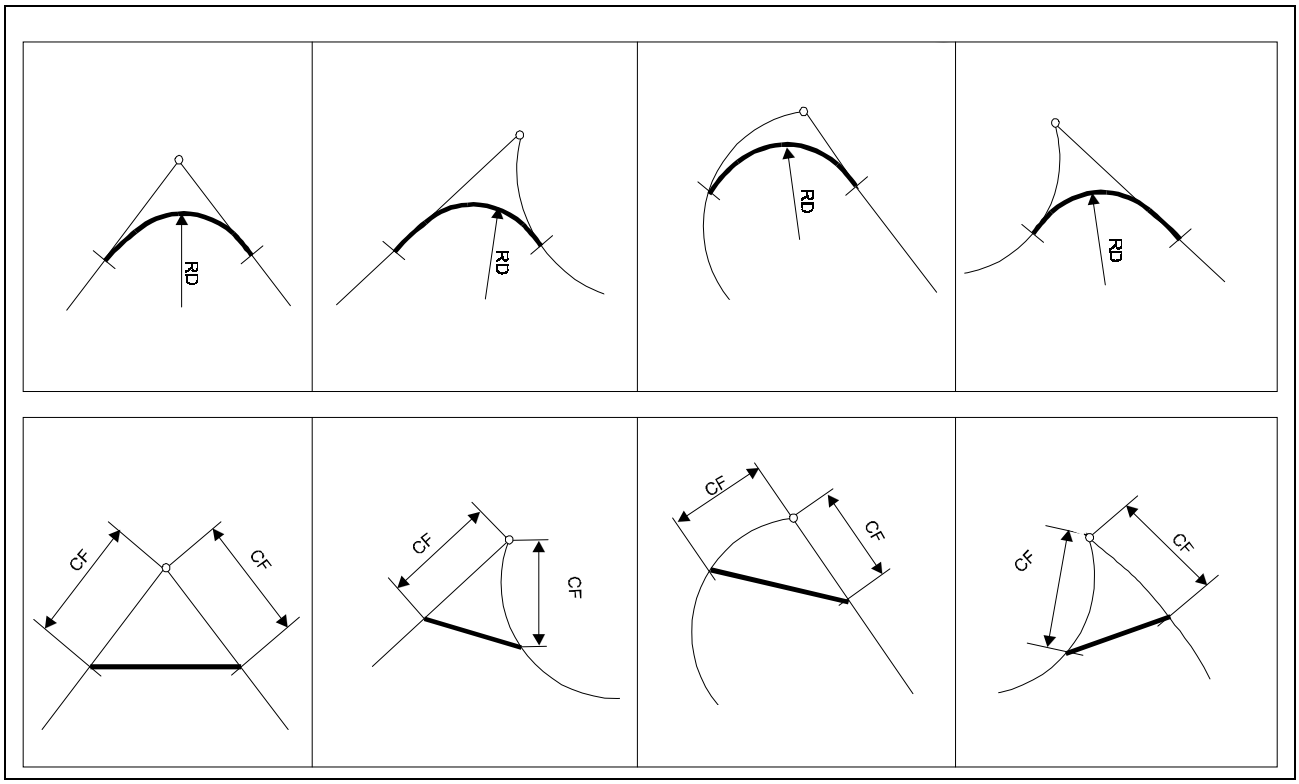


Fig. 2-1: Inserting chamfers and roundings between linear and circular contours

- Specifying the RD command tangentially inserts an arc of the radius RD between the preceding and the subsequent motion command.
- The CF command has the following effect: Starting from the intersection point of the motion commands involved, the chamfer width CF is removed from both motion blocks; and the resulting co-ordinate values are interconnected by a linear path (G1).
- The value that follows CF specifies the chamfer width; the value after RD specifies the rounding radius.
- The instructions CF and RD may be inserted between two motion blocks, at the end of the first block. The required chamfer or rounding will then be inserted after the block in which it has been programmed. Alternatively, the CF or RD command may be inserted in a separate block between two motion blocks.
- Chamfers and roundings are always produced on the active plane.

Example

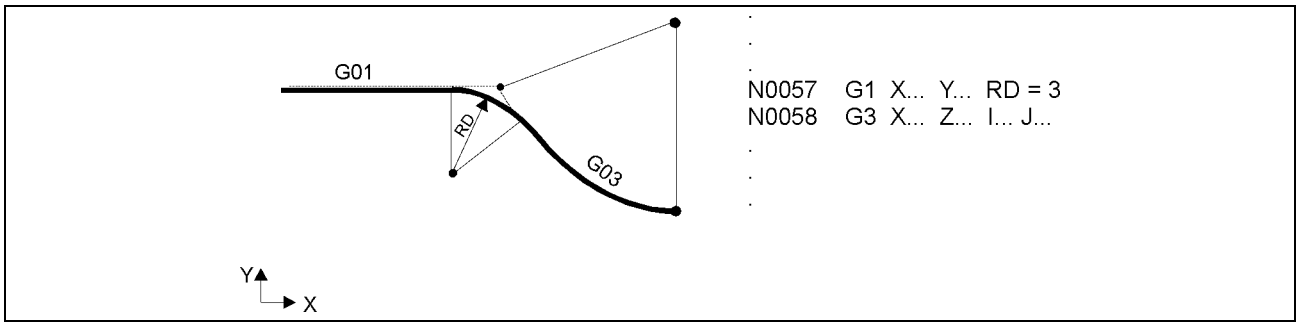


Fig. 2-2: Inserting a rounding

Contiguous motion blocks

- Chamfers and roundings should only be inserted between contiguous motion blocks. There may be a maximum of twenty blocks without motion between two motion blocks that shall be interconnected by a chamfer or rounding.
- The preceding and the subsequent motion block must contain either a linear or a circular movement.
- The command for inserting a chamfer or rounding must be written either in the first motion block, or after it, but always before the second motion block. If the compiler encounters the insertion command for a chamfer or rounding in the second motion block, it inserts the chamfer or rounding between the second and the subsequent movement.
- If the instruction for inserting a chamfer or rounding is written in a separate NC block, the immediately preceding NC block must contain the related linear or circular movement.
- Movements that are outside the active working plane cannot be interconnected by chamfers or roundings.

Illegal commands

Chamfers or roundings cannot be inserted between two motion blocks if one of the following functions is selected or de-selected:

- Radius/diameter programming (G15, G16),
- Changing planes (G17, G18, G19 and G20),
- Transformation functions (G30, G31, G32),
- Zero offsets and rotations (G50 through G59),
- Dimension inch/mm (G70, G71),
- Mirror function (G72, G73),
- Homing axes (G74),
- Travel to dead stop / canceling any axis pre-loading (G75, G76),
- Repositioning and restarting (G77),
- Scaling function (G78, G79),
- Absolute/incremental dimension (G90, G91),
- Jump instructions and program branches (BEQ, BER, BES, BEV, BMI, BNE, BPL, BRA, BRF, BSR, BST, BTE, JVE JMP, JSR)
- Jump labels.

No variables

For the NC blocks, between which a chamfer or rounding shall be inserted, the end points that lie in the current working plane must not be specified by variables.

Note: Inserting a specified chamfer or rounding between the preceding and the subsequent motion block must geometrically be possible. If this is not possible, the compiler automatically reduced the chamfer or rounding concerned to a corresponding value (if necessary even to '0', without error message).

3 Graphical NC Editor

Function

The graphical NC editor represents an efficient and highly precise tool that supports parts programming. It enables the user to easily define geometric elements (e.g. parts contours) graphically, and to specify their machining.

At the end of the dialog, the user may chose whether the data that is required for machining shall be saved in the form of NC blocks or in the form of a function call, together with the related parameters, in the NC program.

Access

The graphical NC editor is called within the user interface:

- In the line editor of the textual user interface via the soft key <GNE>.
- In the full screen editor of the textual user interface via the soft key <Control W>.
- In the editor of the graphical user interface via the soft key <GNE>.

Syntax

The graphical NC editor produces the following instructions:

- WINDOW_01 (..., ..., . . .) ;Define window size for turning
- WINDOW_02 (..., ..., . . .) ;Define window size for milling
- CONT (..., ..., . . .) ;Definition of the initial part contour or of the final part contour

  :
  :

  END_CONT
- FORM_20 (..., ..., . . .) ;recess - turning
- FORM_50 (..., ..., . . .) ;straight elongated hole - milling
- FORM_51 (..., ..., . . .) ;round elongated hole - milling
- FORM_52 (..., ..., . . .) ;circle - milling
- FORM_53 (..., ..., . . .) ;polygon - milling
- FORM_54 (..., ..., . . .) ;straight text - milling
- FORM_55 (..., ..., . . .) ;round text - milling
- FORM_56 (..., ..., . . .) ;rectangle - milling
- FORM_57 (..., ..., . . .) ;rectangle centered - milling
- CYCLE_10 (..., ..., . . .) ;contour cut - turning
- CYCLE_11 (..., ..., . . .) ;roughing - turning
- CYCLE_12 (..., ..., . . .) ;residual cut - turning
- CYCLE_40 (..., ..., . . .) ;contour cut - milling

Notes:

- Further functions and the related instructions of the graphical NC editor are in preparation.
- The instructions of the graphical NC editor shall only be created and modified in a dialog. To perform a modification, set the cursor on the key word concerned and activate the graphical editor.
- Subsequent changes of the machining strategy are not possible if machining programming is entered in the NC program in the form of NC blocks. Starting from the contour concerned, machining must then newly be programmed. The graphical editor must have been exited after the contour concerned has been created, thus saving the contour definition in the NC program. If the contour(s) has (have) not been saved by exiting the graphical editor, geometry and machining must both be programmed again.

CAUTION

⇒ The data items *tool edge orientation*, *tool radius*, *corner angle*, and *setting angle* that had been assumed when the machining program was created must exist when machining is performed.

4 Macro Technique

Macro A macro is the combination of individual instructions, that usually must be programmed repeatedly, into a comprehensive instruction with its own name.

Syntax DEFINE ... AS ...

Explanation A macro permits instructions to be combined, that must always be written in the same sequence (for safety reasons, for example). It enables DIN G codes (such as the drilling cycles G80 through G89) or DIN auxiliary functions (such as M6) to be simulated. Furthermore, it enables functional sequences that cannot be accessed from the PLC (such as spindle control during program mode) to be controlled by a single command from the NC.

Global / local macros Besides the local macros, which the user may define within an NC program and employ subsequently, the machine manufacturer can store global macro definitions in the *NC Options* menu (in the *NC Programming* menu item). In contrast to the local macro definitions, they are valid in all NC programs and in MDI operation of the graphical user interface.

Example 1. Changing tools

```
      :
N0035  DEFINE  M860 AS  M86 M3 S10  ;Declutching while the spindle
                                     rotates slowly
N0036  DEFINE  M6 AS  BSR .WZW      ;Simulating the M6 DIN tool
                                     changing function
N0037  DEFINE  QUICK  AS  G01 F15000
                                    ;swift motion at 35 m/min
N0038  DEFINE  ANPOS_X  AS  MTD (112, 1, 8, 1)
                                    ;X load position for tool change
N0039  DEFINE  ANPOS_Y  AS  MTD (112, 1, 8, 2)
                                    ;Y load position for tool change
N0040  DEFINE  ANPOS_Z  AS  MTD (112, 1, 8, 3)
                                    ;Z load position for tool change
;
N0041  QUICK  X = ANPOS_X  Y = ANPOS_Y  M860
                                    ;swift loading in X, Y and
                                     declutching
N0042  Z = ANPOS_Z  M6              ;swift loading in Z and
                                     changing tools
          :
          :
```

2. Tool correction compensation

```
          :
N0086  DEFINE  L3_KORR  AS  TLD ( , 1, @101, , 1, 13, )
                                    ;tool wear
N0087  DEFINE  D_SOLL  AS  MTD (114, 2, 0, 1)
                                    ;actual tool diameter
N0088  DEFINE  D_IST  AS  MTD (114, 2, 0, 2)
                                    ;tool command diameter
;
N0089  L3_KORR = (D_SOLL - D_IST) /2
                                    ;computation of the tool wear
          :
```

Notes:

- A macro name may have up to 20 characters.
- The instruction related to a global macro may contain up to 156 characters (consisting of 2 lines with up to 78 characters each).
- With a local macro, the compiler interprets all NC instructions that follow the AS key word as the instruction sequence that must be inserted instead of the macro name. Consequently, no further NC commands may be programmed after a DEFINE instruction within an NC block.
- Nesting macros is not permitted. This means that there may be no further macro calls within an instruction sequence that is to be inserted.

  Example: DEFINE M86 AS M86 M3 S10
- In contrast to the textual user interface and to the SOT, within the graphical user interface, the user may program global macros in MDI mode.
- Key words may not be super-defined by macros.

CAUTION

⇒ Macro technique permits the programming language to be heavily simplified. Thus, it must be used with extreme care.

5 Modal Function

Modal function

The MODF_ON(STRI) modal function permits repeatedly used expressions to be written once only.

Syntax

```
MODF_ON(STRI)     ;activates modal function (modal function on)
MODF_OFF          ;de-activates modal function modal function off)
```

Explanation

- The string *STRI* that is, in parentheses, transferred with the modal function may contain up to 80 characters.
- It is inserted in all subsequent blocks with axis movements.
- The modal function is de-selected using the MODF_OFF key word.

Notes:

- The instruction concerned is executed immediately in the NC blocks in which the user writes a modal instruction using MODF_ON.
- The MODF_OFF instruction de-activates the modal instruction in the block in which it is programmed.
- It must be noted that the modal function (such as MODF_ON(RD 2)) does not have an effect on blocks without axis movements (i.e. without feed axes). This is also true for contours that were created in the graphical editor and were saved as function block in the NC program.

Examples

1) Drilling holes

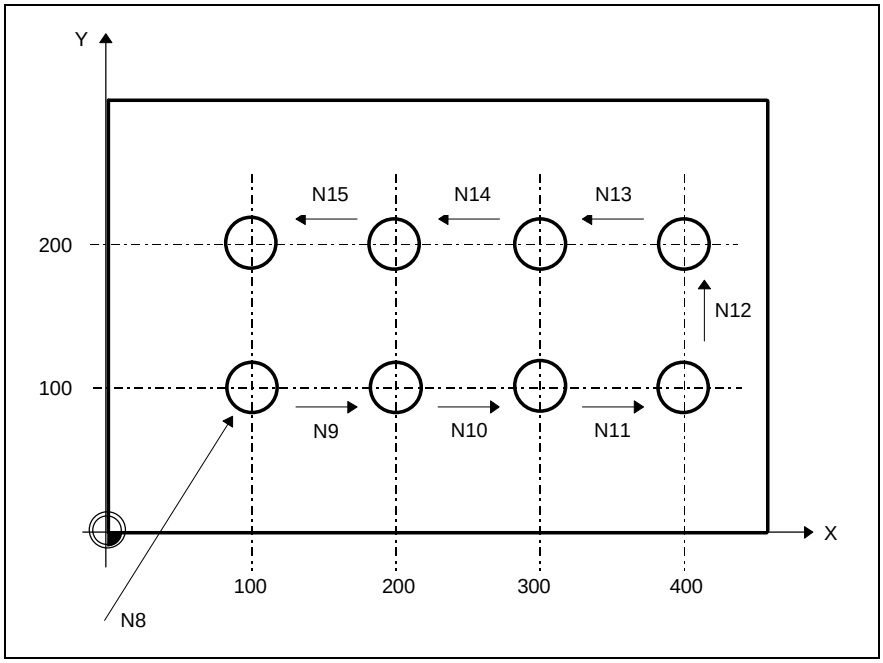


Fig. 5-1: Example: Drilling holes

```
;
N0000 T6 M6
N0001 G54 G0 X-10 Y-10 Z50 S3500 M3
;
;Pre-assign standard drilling cycle '∗G83'
;
;**************** G83 - deep hole drilling chip removing **********************
;
N0002 @171=-20.0                    ;depth (abs)
N0003 @172=6.0                      ;chip depth (ink)
N0004 @173=2.0                      ;safety distance (abs)
N0005 @174=0.5                      ;cutter distance (ink)
N0006 @175=0.0                      ;dwell
N0007 @176=250.0                    ;feed rate
;*******************************************************************************
;
N0008 X100 Y100 Z10 MODF_ON (BSR .*G83)
N0009 X200
N0010 X300
N0011 X400
N0012 Y200
N0013 X300
N0014 X200
N0015 X100
N0016 MODF_OFF
N0017 T0 M6
N0018 G0 G53 X570 Y490
N0019 M30
```

2) Modal rounding and chamfering

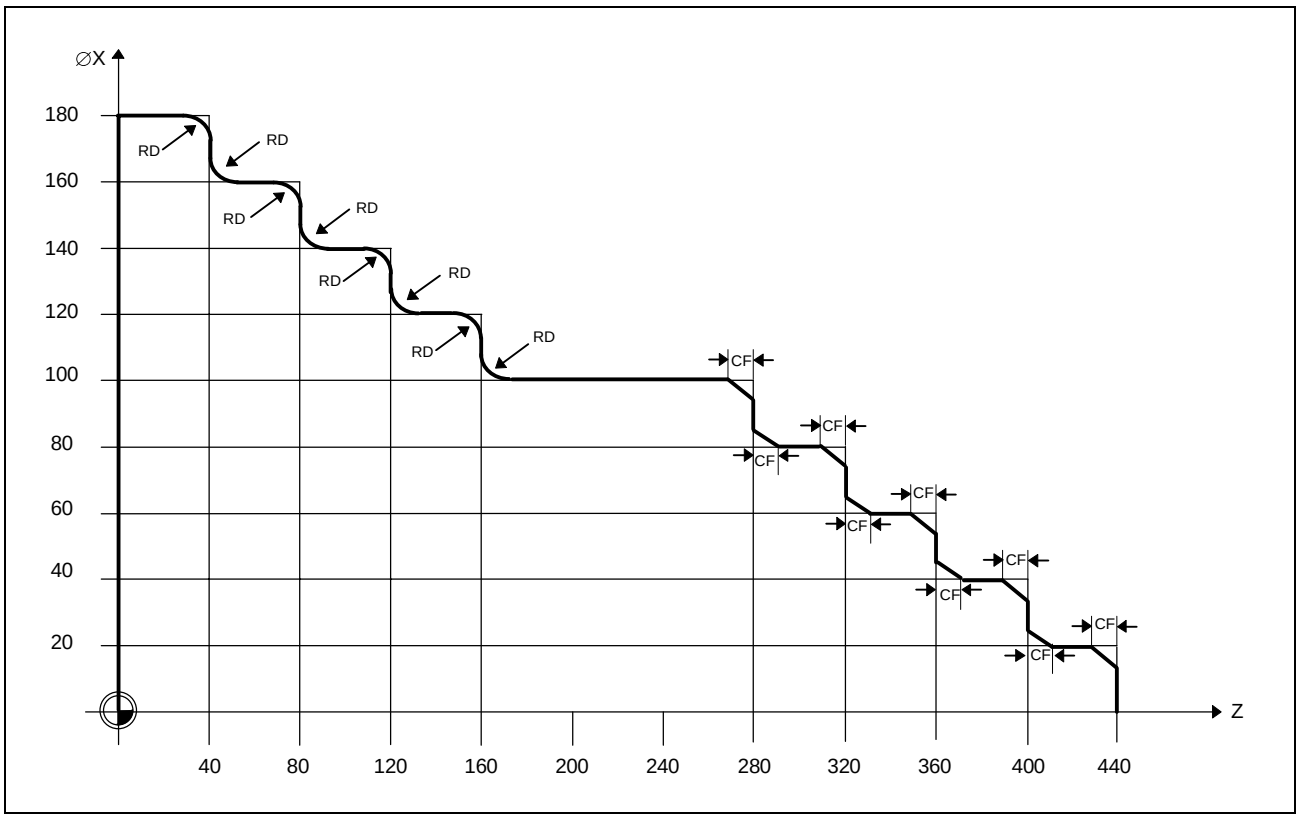


Fig. 5-2: Example: Modal rounding and chamfering

```
N0000 (parts name : stairs)
N0001 T3 BSR .M6 (PRE-TURNING TOOL)
N0002 G18 G54 G16 G90 G71
N0003 M69
N0004 G92 S2000
N0005 [turning contour C1 without cut segmentation]
N0006 G0 G18 G54 G16 G95 G97 G9 G7 Z444 S2000 M3 M9
N0007 X0
N0008 G1 G42 Z440 F.3
N0009 X20 MODF_ON (CF2.0)
N0010 Z400
N0011 X40
N0012 Z360
N0013 X60
N0014 Z320
N0015 X80
N0016 Z280
N0017 X100
N0018 Z160 MODF_ON (RD2.5)
N0019 X120
N0020 Z120
N0021 X140
N0022 Z80
N0023 X160
N0024 Z40
N0025 X180
N0026 Z0 MODF_OFF
N0027 G0 G40 X182 Z1
N0028 X184
N0029 Z450
N0030 M5
N0031 M70
N0032 M62
N0033 G53 G90 G47 M5
N0034 M30 [ ]
```

6 Enhanced Look-Ahead Function

Enhanced look-ahead function

The enhanced look-ahead function optimizes the velocity curve of the programmed path movement during compilation and/or the program download. If required and without modifying the programmed contour, the look-ahead function inserts intermediate blocks in order to achieve a steadier path velocity curve.

Using the enhanced look-ahead function

Using the enhanced look-ahead function is always expedient if an NC program is to be executed that consists of very short NC blocks, and if the internal block look-ahead function proves insufficient.

With non-tangential block transitions, the NC always reduces the velocity to zero at transitions that are crossed with G6 or G8. In order to be able to stop in the last block, this process frequently requires continuous deceleration across several blocks. With very short NC blocks, the internal *MT-CNC look-ahead function*, however, usually does not recognize the end of the polygon blocks, or too short an NC block, or a non-tangential block transition in time. Consequently, the NC does not induce the deceleration process in time, aborts NC program execution during the deceleration process, and issues the error message 'deceleration distance too short'.

Using the enhanced look-ahead function enables the compiler to adjust the velocity profile of certain program sequences within the NC program to the maximum velocities and the acceleration capability of the individual axes. During acceleration and deceleration processes, the compiler therefore splits the NC blocks into sub-blocks of different F values wherever that is necessary.

Syntax

```
LA_ON     ;activates the enhanced look-ahead function
           (Look-Ahead function, on)
LA_OFF    ;de-activates the enhanced look-ahead function
           (Look-Ahead function, off)
```

Global variables

Global variables have been introduced that are used as transfer parameters for the enhanced look-ahead function. Usually, the user can employ these variables without modification. Some variables may be pre-assigned in the *NC Options* menu.

METB

1. Minimum execution time of an NC block

Name: METB (minimum execution time for one block)

Range: 3 ... 30 [ms]

Default: in *NC Options*

Explanation: The global variable *minimum execution time of an NC block* (METB) specifies the shortest execution time of an NC block within the polygon sequence that is to be optimized. It must be greater than the related block cycle time.

VFBT

2. Velocity factor for block transition

Name: VFBT (velocity factor for block transition)

Range: 1 ... 10 [-]

Default: in *NC Options*

Explanation: The global variable *velocity factor for block transition* (VFBT) permits the velocity changes at non-tangential block transitions to be influenced. It is:

$$velocity\,change = 2 * axis\,acceleration * VFBT\,[mm/s^2]$$

BBTRC *3. Block buffer for tool radius compensation*

Name: BBTRC (block buffer for tool radius compensation)

Range: 1 ... 10 [-]

Default: in *NC Options*

Explanation: The global variable *block buffer for tool radius compensation* (BBTRC) specifies how many NC blocks the enhanced look-ahead function shall take into account in advance when it computes and checks the tool radius compensation.

TL_RADIUS *4. Specify tool radius*

Name: TL_RADIUS[T no., E no.]

Range: T no.: 1 ... 9999999

E no.: 1 ... 9

TL_RADIUS [T no., E no.]: 0 ... 999.99999

Default: If, within the NC program, the compiler cannot find any specifications with respect to the tool radius with T_RADIUS[T no., E no.], the compiler accesses the maximum and minimum tool radius of the related tool in the setup list, and computes the radius as follows:

$$Radius = ½ * (R\_min + R\_max)$$

Explanation: Using the TL_RADIUS[T no., E no.] command, the tool radii that are required for the enhanced look-ahead function may centrally be defined at the beginning of the program. The compiler employs the current T or E no. if a T no. or an E no. has not been specified.

Example:

```
:
N0005  TL_RADIUS[1234567,1]=24.995
N0006  TL_RADIUS[923,3]=20.31
N0007  TL_RADIUS[9,9]=29.89
:
```

CAUTION

⇒ If the tool radius path correction of the enhanced look-ahead function is employed (TRC <> 0), the tool radius that, using the pre-defined TL_RADIUS[T no., E no.], has been specified in the NC program during compilation must exist during machining.

⇒ The tool radius compensation is limited to milling tools of correction type 3.

TRC *5. Tool radius correction*

Name: TRC (tool radius correction for look-ahead function)

Range: 0 ... 2 [-]

Default: 0 (cannot be modified)

Explanation:

- TRC=0: The enhanced look-ahead function does not perform radius correction
- TRC=1: The enhanced look-ahead function performs radius correction to the left of the contour, using the tool radius specified under TL_RADIUS.
- TRC=2: The enhanced look-ahead function performs radius correction to the right of the contour, using the tool radius specified under TL_RADIUS.

Recommendation: If the enhanced look-ahead function shall be used, and if tool radius correction is necessary, the tool radius correction o the enhanced look-ahead function (TRC $\neq$ 0) shall be used and the internal tool radius correction shall be de-activated (G40).

ADTRC *6. Approach distance for tool radius compensation*

Name: ADTRC (approach distance for tool radius compensation)

Range: T no.: 0 ... 999 99999

Default: 0 (cannot be modified)

Explanation:

- ADTRC = 0 To set up the tool radius compensation, the enhanced look-ahead function does not take any approach or retract distance into account.
- ADTRC $\neq$ 0 If tool radius compensation is activated with TRC=1 or TRC=2, the enhanced look-ahead function inserts a straight line with tangential transition and of the length that must be specified here before the first polygon element (first motion block after LA_ON), and after the last polygon element (last motion block before LA_OFF).

Contiguous motion blocks Only NC blocks that contain G1,G2, G3 movements, event commands (SE,RE) velocity specifications (F), acceleration limits (ACC_EFF) and swift auxiliary function outputs (MQxxx, QQxxxx and Sxxxxx.xx, if S has been selected as a swift auxiliary function) may occur within the program sequence that shall be optimized.

No variables The end points of NC blocks whose velocity profile shall be processed by the enhanced look-ahead function may not be specified by variables.

Tool management Changing tools, including the related T call and the edge selection, must be performed prior to activating the enhanced look-ahead function or after it has been de-activated.

Per cent acceleration correction In certain program sequences and, if applicable, depending on the tool or workpiece weight, the resulting path acceleration must be reduced.

```
ACC_EFF          ;changing the efficient resulting path acceleration
```

permits the actual resulting path acceleration to be modified. This acceleration factor ranges from 1% through 200%.

Note: In contrast to the ACC command, the ACC_EFF command does not delimit the maximum path acceleration that is specified in the process parameters. It modifies the actual path acceleration according to the specification.

Axis-related velocities

Besides programming the path velocity via the F value, axis velocities may also be programmed during the look-ahead function.

To specify an axis velocity, the 'F' must immediately (without blank) be followed by the axis name.

Syntax: F<axis name>=<axis velocity in mm/min>

Example: :

```
N0045  G01  X 2034  Z1 421  FZ1=4500      ;axis-related velocity
                                            for Z1
:
```

Note: If the user programs several velocities within an NC block, that NC block and the subsequent NC blocks are executed with the last velocity to have been specified until the next velocity instruction is encountered.

Access to current data in the controller

The command *access to current data* ACD_COMP[...] permits the access to current controller data (currently only NC variables) during compilation.

Example: Reading the tool radius during compilation

After each dressing of a grinding wheel, a dressing program updates half the diameter of the grinding wheel in the NC variable '@1:120'. During compilation, that value must be taken into account as the tool radius.

```
TL_RADIUS[1,1] = ACD_COMP[@1:120]-0,2;   Read tool radius from
                                         NC variable @1:120
                                         and subtract 0.2 mm.
```

Example: Needle grinding

A given polygon curve must be traversed in reciprocated movement at highest velocity possible. This requires the velocity curve of the programmed path motion to be optimized, using the enhanced look-ahead function.

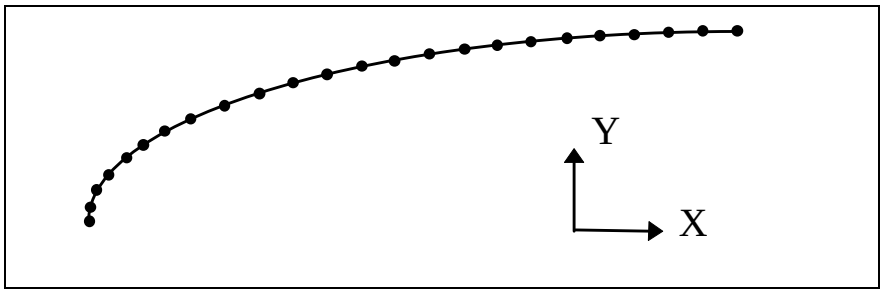


Fig. 6-1: Velocity curve of a polygon that is to be optimized for grinding needles

```
;Grinding needles on the XY plane
;Grinding wheel radius:          2.50000
;File name:                      TP1
;
N0000  (parts name : TP1)
N0001  T2 BSR .M6  [SCHLEIFSCHEIBE D5]     ;activate tool
N0002  TL_RADIUS [ ] = ACD_COMP[@200]      ;read current tool radius
                                            for compiler
N0003  G0 G17 G40 G54 G71 G48 G8 G6 G98 X-0.19306
       Y3.49431 S1 3000 M3                 ;establish initial  state
N0004  @101=200                            ;loop counter for number
                                            of oscillating strokes =
                                            200
N0005  .PEN @100=@101-0 BEQ .ENDPEN        ;terminate oscillation ?
N0006  F4000                               ;set path velocity
;
N0007  TRC=1                               ;tool radius correction
                                            left of the contour
N0008  ADTRC=1                             ;approach dist. to set up
                                            tool radius compens.
N0009  ACC_EFF=90                          ;modify effective path-
                                            acceleration
N0010  LA_ON                               ;enhanced look-ahead
                                            function ON
;
N0011  G1 X0.8 Y1.2                        ;polygon curve
       :
       :
N0030  LA_OFF                              ;enhanced look-ahead
                                            function OFF
       :
;
N0051  @101=@101-1 BRA .PEN                ;decrement loop
                                            counter
N0052 .ENDPEN BSR .ABRICH          ; call dressing cycle
N0053  RTS
;
N0054  PROGRAMMENDE
```

Notes:

- In reverse programs, the LA_OFF command must be programmed when the LA_ON command is used.
- The compiler does not take into account any velocity changes of the axes that are caused by a rotation of the contour.

7 INDRAMAT Compiler

Tasks of the INDRAMAT compiler

The INDRAMAT compiler

- searches the NC source program for the key words listed above according to the compiler settings;
- checks whether all necessary boundary conditions have been fulfilled;
- retains new
  - ⇒ macro and
  - ⇒ modal definitions and
- inserts the programmed
  - ⇒ macro instructions,
  - ⇒ modal instructions,
  - ⇒ chamfers and roundings,
  - ⇒ machining processes (chip removing movements or contour cuts for the contours created in the dialog);
  - ⇒ performs the required path velocity optimization, and
  - ⇒ checks the syntax according to the compiler setting (in preparation).

Compiler settings

The INDRAMAT compiler can be activated and de-activated in the *NC Options* menu. The user interface also invokes the INDRAMAT compiler during the downloading process. Thus, that compiler setting is particularly important for all previous applications, in particular under the timing aspect. The INDRAMAT compiler should not be activated if the above-mentioned compiler function is not required. The presetting does not activate the compiler.

Note: If one of the above-mentioned key words occurs within the NC program, and if the compiler has been de-activated, the controller generates an error message at the location concerned during the download process.

CAUTION

⇒ After a modification of the machine parameters has been made, the NC programs must be made invalid, re-compiled, and loaded into the controller before they are processed.

Status bits

The user interface displays the following status bits in the parts program directories:

Symbol	Designation	Comment
		The NC program need not be compiled.
N	not compiled	The NC program has not yet been compiled.
C	compiled	The NC program has successfully been compiled. **Note:** A syntax check has not been performed (it will only be performed when the program is downloaded into the controller).
E	error	An error has occurred during NC program compilation.

Calling the INDRAMAT compiler

- Whenever it is necessary, the user interface automatically calls the INDRAMAT compiler at the NC program download of each NC program that is to be transferred.
- Using the <*Compile*> soft key, The operator can start the compilation process for an NC program in the parts or machining directory (independently of machining and program transfer). The compiling process then solely refers to the selected NC program.
- The user interface shows the <*Compile*> soft key only if the NC option 'Compiler active' in the *NC-Options* menu (in the *NC Programming* menu item) has been set to 'Yes'.

8 User-Related Compilers (Option)

Basics

User-related compilers enable the machine manufacturers to integrate their own know-how into the controller. They enable technology-related functions, such as look-ahead interference checks, program generators, circles with radii of several kilometers, ellipses, or any arithmetical function to be implemented.

The user-related compiler can be ordered as an option under

SWA-MT*CNC-NUC-17VRS-MS-C1,44

Entering user-related compiler

In the *NC Options* menu, the user may insert up to two user-related compilers before and after the INDRAMAT compiler. To do this, the related EXE name must be entered at the required location:

- User-related compiler 1 ....
- User-related compiler 2 ....
- INDRAMAT compiler
- User-related compiler 3 ....
- User-related compiler 4 ....
- Syntax check

Note: If the machine manufacturer inserts a user-related compiler before the INDRAMAT compiler, the compiler will -due to the sequence of the calls - encounter the unmodified NC program. If, in contrast, the user-related compiler is inserted after the INDRAMAT compiler, the key words have already been interpreted. This means that, for example, the macros or the function calls for cut segmentation (e.g. CYCLE_MP) have already been resolved.

Requirements for compiler run

The user interface only calls the INDRAMAT compiler for an NC program if all the following requirements are satisfied:

- The <Compile> soft key has been selected or an NC program or cycle download has been initiated.
- The *Compiler active* compiler setting has been set to 'Yes' in the *NC Options* menu.
- At least one key word or macro has been programmed or modified in the program concerned.
- The program has newly been created or edited.
- The program has not yet been compiled (status bit 'N' in the parts program directory).

The user interface activates a user compiler that the machine manufacturer has been entered by specifying the program name (e.g. CHECK.EXE) and by activating the compiler (*Compiler active = Yes*) in the *NC Options* menu whenever the <Compile> soft key is activated and upon each download of an NC program or cycle. The user interface calls the user-related compiler, even if there are no INDRAMAT key words within the NC program concerned.

Directory

The compiler must be saved as an EXE file in the '...\MT-CNC' directory.

Controlling the compiler calls

The compiler calls are controlled using the COMPILE.LST and COMPILE.RET files in the directory

'...\MT-CNC\ANLAGExx\MT_TMP' (xx ε {0 through 15}).

Control file structure COMPILE.LST

Prior to invoking a compiler, the user interface creates the COMPILE.LST control file. In that file, it saves, for the compiler concerned, the DOS file names (with extension 'DAT') of the NC programs that are to be compiled. Furthermore, it saves information for the INDRAMAT compiler.

Example: Contents of a COMPILE.LST file

```
C:\MT-CNC\ANLAGE00\NCPRG\N0010001.DAT 00000000000000000000000000110010
C:\MT-CNC\ANLAGE00\NCPRG\N0010002.DAT 00000000000000000000000000111010
C:\MT-CNC\ANLAGE00\NCPRG\N0010003.DAT 00000000000000000000000000011011
C:\MT-CNC\ANLAGE00\NCPRG\N0010005.DAT 00000000000000000000000000110100
```

Boundary conditions in working with the COMPILE.LST control file

- The user-related compiler must access the associated DAT file if it cannot find an object file with the extension 'NCO' for an NC program that has been specified in the COMPILE.LST control file. This particularly applies to user-related compilers that are called before the INDRAMAT compiler. User-related compilers that the user interface invokes after the INDRAMAT compiler must also reckon that there is no related NCO file. This is the case, for example, if the NC program concerned does not use key words that require the INDRAMAT compiler to be called.
- The user interface starts the entered user-related compilers even if the INDRAMAT compiler need not be called. They enter the files that have not yet been compiled into the associated COMPILE.LST file.
- The only case when the user interface does not start the user-related compilers is after all files have successfully been compiled.

Re-compiling

The user interface will not accept any NC program for which the status bit 'C' is shown in the NC program directory into the COMPILE.LST file before the compiler is invoked. Thus, re-compiling a previously compiled file is only possible after a modification.

Structure of the COMPILE.RET control file

From each compiler within the COMPILE.RET control file, the user interface expects the DOS file names of the compiled NC programs (with extension 'DAT') and the related status string. The status string can assume the states 'ERROR' or 'COMPILED'. The user interface shows the status strings in the program directories as status bits 'N' (not compiled) 'C' (compiled), and 'E' (error).

Example: Contents of a COMPILE.RET files

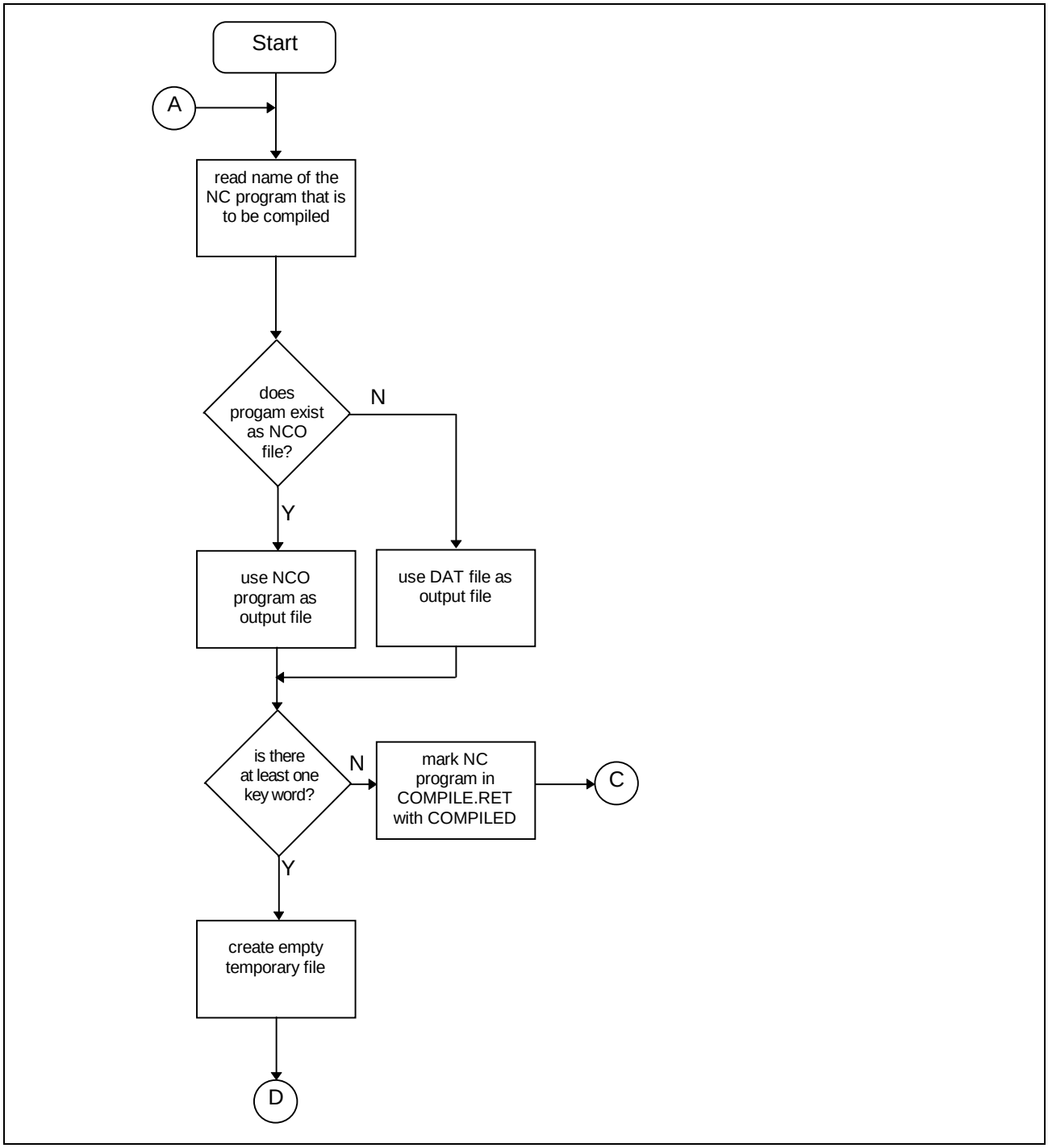
```
C:\MT-CNC\ANLAGE00\NCPRG\N0010001.DAT COMPILED
C:\MT-CNC\ANLAGE00\NCPRG\N0010002.DAT COMPILED
C:\MT-CNC\ANLAGE00\NCPRG\N0010003.DAT ERROR
C:\MT-CNC\ANLAGE00\NCPRG\N0010005.DAT COMPILED
```

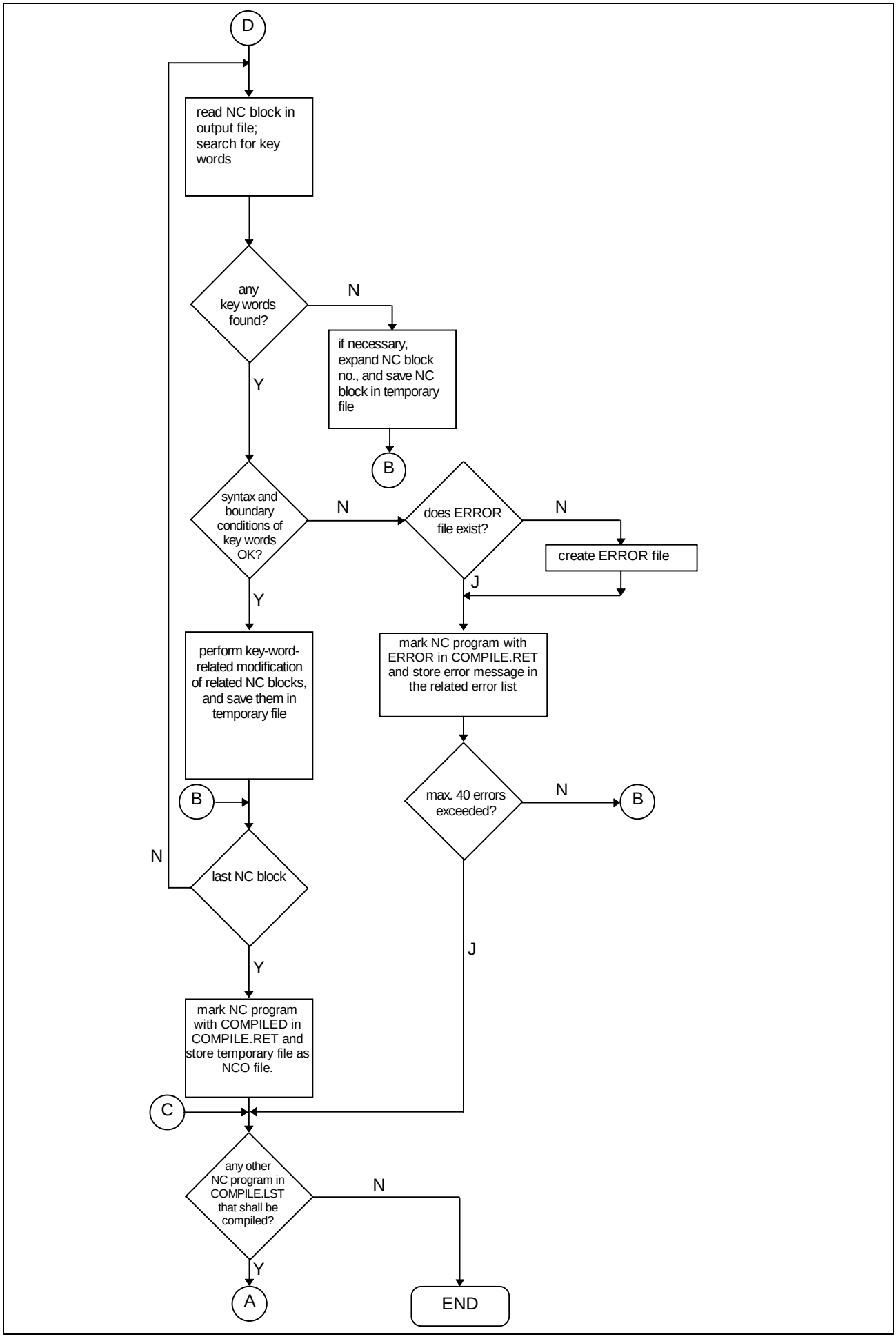
Boundary condition in working with the COMPILE.RET control file

- For the subsequent compiler, the user interface removes NC programs that a compiler has marked with ERROR in the COMPILE.RET file from the related LST file.
- If a compiler detects at least one error in an NC program (in the DAT or NCO file), and marks that program with ERROR in the COMPILE.RET file, the user interface expects an error file of the same file name as the NC program, and the extension '.ERR' to exist in the '...\MT-CNC\ANLAGExx\ NCPRG\' (xx ε {0 through 15}) directory.

- If a user-related compiler cannot find a user key word in an NC program, it need not create an NCO file (the same is true if an NCO file does not exist yet). The 'COMPILED' state, however, must be set in the COMPILE.RET file for the NC program concerned

Base structure of a user-related compiler




D
read NC block in output file; search for key words
any key words found?
N
if necessary, expand NC block no., and save NC block in temporary file
B
Y
syntax and boundary conditions of key words OK?
N
does ERROR file exist?
N
create ERROR file
J
Y
perform key-word-related modification of related NC blocks, and save them in temporary file
mark NC program with ERROR in COMPILE.RET and store error message in the related error list
B
max. 40 errors exceeded?
N
B
N
last NC block
J
Y
mark NC program with COMPILED in COMPILE.RET and store temporary file as NCO file.
C
any other NC program in COMPILE.LST that shall be compiled?
N
Y
A
END

9 Error Handling

Error messages

When the INDRAMAT compiler detects an error, it immediately stores the related error text in the associated error file, and continues the compilation process of the program concerned until either the program end or a maximum of 40 errors has been reached.

Likewise, the user-related compilers store the related error text of each error in the associated error file.

Name

Except for the extension '*.ERR', the error file is of the same name as the associated NC program.

Directory

The error file must be saved in the '...\ANLAGExx\NCPRG\' (xx ε {0 through 15}) directory.

Error file structure

The first line of the error file contains an identification string that is followed by the list of errors in the following form:

```
Nxxxx:$W:yyyy
zzzz
wwww
```

or

```
Auuuu:$W:yyyy
zzzz
wwww
```

Explanation:

- Nxxxx: Number of the NC source block in which the error has been detected.
- yyyy: Number of lines with additional text.
  The additional text must not be longer than 14 lines.
- zzzz: Error message
  An error message must not be longer than 52 characters.
- wwww: Additional text
  Each of the maximum of 14 lines of additional text may contain up to 65 characters.
- Auuuu: General error number that is not related to a specific NC source block.

Notes:

- The user compiler must transfer the identification string of the NC source program directly from the related NC program into the error file. The user interface automatically updates the identification string at the end of the compiler run.
- The error messages, together with their additional text, must sequentially (without gaps or delimiting strokes) be saved in the error file.

Example: Contents of an error file

iMTc/002021/04.16/0000000195/08.08.1996/09:36:38/N0010003.ERR/1

N0030:$W:0002

Target position for CF/RD instruction not found.

Either the next motion block is missing or there are more than 20 NC blocks between the first and the next motion block.

A0001 : $0003

Maximum number of NC blocks exceeded.

Proper compilation of the program requires more than 10,000 NC blocks.

Measure: Please split the NC program.

Creating and deleting an error file

After a modification has been made in the NC program, the user interface clears the error file concerned. Likewise, the user interface clears all error files after a global macro has been modified.

The INDRAMAT compiler only creates a new error file if it encounters an error in the NC program that shall be compiled.

Error window

Selecting the <*Error display*> soft key enables the user to display the errors within the textual user interface or the graphical user interface. Selecting <*Control-E*> displays the errors in the full-screen editor.

Within an error window, the user interface consecutively displays the error messages (brief messages) that have been stored in the error file. The additional messages may be displayed by highlighting the required error message and pressing the <*Additional text*> soft key.

In addition to scrolling the window, the cursor can be set on an error message in the error window. Pressing the ENTER key locates the associated error in the NC program.

Reaction after an error

If a compiler detects an error, it continues the compiling process until either the end of the NC program concerned or the maximum of 40 error messages has been reached, marks the NC program concerned with 'ERROR' in the COMPILE.RET file, and aborts the further processing of the file. Next, the compiler processes the subsequent files. Finally, the user interface clears the file that has been marked with 'ERROR' in the COMPILE.LST file for all subsequent compilers, and starts the next compiler. The same procedure will be repeated if the newly activated compiler detects an error in one of the remaining files.

10 Source Program

Name

The source program is named ‘Nuxxyzzz.DAT’.

Symbol	Designation	Range
u	Plant address	0, ... 9, A, B, C, D,E, F
xx	Package number	1 ... 99
y	Process number	0 ... 6
zzz	Directory number	001 ... 099

Directory

The source program can be found in the ‘...\ANLAGExx\NCPRG\’ (xx ε {0 through 15}) directory.

Source program structure

- The first line contains an identification string.
- Next follows the NC header. It begins with the characters ‘%NPG’, and ends with the characters ‘%NPG’.
- Next follow the NC blocks. They must adhere to the following rules:
  - The source file may not contain more than 10,000 lines.
  - Each NC block may contain a maximum of 230 characters that can be split up into a maximum of four lines.
  - Including the preceding characters for block numbering and for the preceding blanks that mark a subsequent line, a line of an NC block may not be longer than 71 characters.
  - An NC block begins with the source block number that consists of an 'N', a four-digit consecutive number, and a trailing blank.
  - Subsequent lines of an NC block must be marked by six preceding blanks.
  - A comment is marked by a ‘;’.
  - After the consecutive block number, the last line must always contain PROGRAMMENDE.

CAUTION

⇒ Adhering to the above-mentioned rules is mandatory for proper operation of the user interface and the controller.

Example: Contents of an NC source file

```
iMTc/002004/04.16/0000002737/07.08.1996/14:44:42/N0010001.NCO/4
%NPG:0:01:0:001
!01.00
#01Shaft with H7 fit        814229.07.9615:44:20
$
$
$
$
*Shaft with H7 fit
%NPG
N0000 G0 X0 Y0 Z0
N0001 G2 X200 Y0 I100 J0 F2000
N0008 M18
          .
          .
N0114 G3 X200 Z300 I200 K150
N0115 M30
N0116 PROGRAMMENDE
```

11 Object File

Name The compiled NC program (or object file) has the name of the associated NC program assigned, but with the extension '.NCO' (NC object file).

Directory Like the source files, the object files are to be stored in the '...\ANLAGExx\NCPRG\' directory.

Creating an object file

- The source program's identification string and the header (from %NPG through %NPG) must completely and unmodified by copied to the object file. The user interface automatically updates the identification at the end of the compilation process.
- The rules that apply to the subsequent NC blocks are nearly the same as the rules that apply to the source file:
  - The object file may not contain more than 10,000 lines.
  - Each NC block may contain a maximum of 230 characters that can be split up into a maximum of four lines.
  - Including the preceding characters for block numbering and for the preceding blanks that mark a subsequent line, a line of an NC block may not be longer than 71 characters.
  - An NC block in the object file begins with the source block number, the enhanced block number, and a blank. The source block number must be taken from the source file. The enhanced block number consists of a '.' and a consecutive four-digit number. With each new source block, that number begins at '0000', and is incremented by '1' for each inserted intermediate block.
  - Like in the source file, subsequent lines of an NC block in the object file must be marked by six preceding blanks.
  - A comment is marked by a ';'. Comments need not be taken from the source file.
  - After the consecutive block number, the last line must always contain PROGRAMMENDE.

CAUTION

⇒ Adhering to the above-mentioned rules is mandatory for proper operation of the user interface and the controller.

Note: The blanks that are contained in the source file must also be transferred.

Example: Contents of an NC object file

```
iMTc/002004/04.16/0000002737/07.08.1996/14:44:42/N0010001.NCO/4
%NPG:0:01:0:001
!01.00
#01Shaft with H7 fit        814229.07.9615:44:20
$
$
$
$
*Shaft with H7 fit
%NPG
N0000:0000 G0 X0 Y0 Z0
N0001:0000 G2 X200 Y0 I100 J0 F2000
N0008:0000 M18
          .
          .
N0114:0000 G3 X200 Z300 I200 K150
N0115:0000 M30
```

N0116:0000 PROGRAMMENDE

Creating and clearing the object file

During the compiler run, the INDRAMAT compiler generates an object file.

The user interface clears the related object file and resets the status bit after each modification of the NC program, in each compiling process in which at least one error has been encountered, whenever an external program or cycle is read, and whenever a loaded macro is modified. Furthermore, changing the compiler settings causes the user interface to clear all existing objects.

CAUTION

⇒ The user interface does not clear the related object files of the station concerned if the user modifies the machine parameters or the setup list.

12 Miscellaneous

Run time program

Textual and graphical user interface, and SOT display the run time program in the position display and in the base screens.

Current source program

Textual and graphical user interface show the current source program in the 'Current NC program' screens. In the base screens, the graphical user interface provides the option of displaying the source program instead of the run time program.

MDI

Within the textual user interface and at the SOT, neither key words nor macros may be programmed in MDI mode.

Within the graphical user interface, the user may also program the global macros during MDI operation. Any other key word will lead to an error message.

NC program upload

When an NC program is uploaded, the user interface loads the NC programs (source programs or compiled programs) from the controller into the NC. The user interface always creates '*.DAT' files in this process, irrespective of whether compiled programs or source programs are concerned. The user interface does not transfer the enhanced block numbering of compiled programs into the '*.DAT' file.

Key words

The specified key words of the INDRAMAT compiler are listed below:

- CF ;Insert chamfer
- RD ;Insert rounding
- DEFINE ... AS ... ;Define macro
- MODF_ON (...) ;Activate modal function
- MODF_OFF ;De-activate modal function
- LA_ON ;Activate enhanced look-ahead function
- LA_OFF ;De-activate enhanced look-ahead function
- METB ;Minimum execution time of an NC block
- VFBT ;Block transition velocity factor
- BBTRC ;Block buffer to tool radius compensation
- TL_RADIUS [..., ...] ;Specify tool radius
- TRC ;Tool radius compensation
- ADTRC ;Approach distance for tool radius compensation
- ACD [...] ;Access to current controller data at compilation time
- WINDOW_01 (..., ..., . . .) ;Define window size for turning
- WINDOW_02 (..., ..., . . .) ;Define window size for milling
- CONT (..., ..., . . .) ;Definition of the initial part contour or of the final part contour

  :

  :

  END_CONT
- START ... ;Definition of the start point
- LINE ... ;Straight line

- CW ... ;Circle clockwise
- CCW ... ;Circle counter-clockwise
- RD ... ;Rounding
- CF ... ;Chamfer
- RELIEF ... ;Relief groove
- FORM_20 (..., ..., . . .) ;Recess - turning
- FORM_50 (..., ..., . . .) ;Straight elongated hole - milling
- FORM_51 (..., ..., . . .) ;Round elongated hole - milling
- FORM_52 (..., ..., . . .) ;Circle - milling
- FORM_53 (..., ..., . . .) ;Polygon - milling
- FORM_54 (..., ..., . . .) ;Straight text - milling
- FORM_55 (..., ..., . . .) ;Round text - milling
- FORM_56 (..., ..., . . .) ;Rectangle - milling
- FORM_57 (..., ..., . . .) ;Rectangle centered - milling
- CYCLE_10 (..., ..., . . .) ;Contour cut - turning
- CYCLE_11 (..., ..., . . .) ;Roughing - turning
- CYCLE_12 (..., ..., . . .) ;Residual cut - turning
- CYCLE_40 (..., ..., . . .) ;Contour cut - milling

13 Index

14 Figures

List of Customer Service Points

Germany

Sales area Center	Sales area East	Sales area West	Sales area North
INDRAMAT GmbH D-97816 Lohr am Main Bgm.-Dr.-Nebel-Str. 2 Phone: 09352/40-0 Fax: 09352/40-4885	INDRAMAT GmbH D-09120 Chemnitz Beckerstraße 31 Phone: 0371/3555-0 Fax: 0371/3555-230	INDRAMAT GmbH D-40849 Ratingen Hansastraße 25 Phone: 02102/4318-0 Fax: 02102/41315	INDRAMAT GmbH D-22085 Hamburg Fährhausstraße 11 Phone: 040/227126-16 Fax: 040/227126-15
Sales area South	**Sales area South-West**		**INDRAMAT Service Hotline**
INDRAMAT GmbH D-80339 München Ridlerstraße 75 Phone: 089/540138-30 Fax: 089/540138-10	INDRAMAT GmbH D-71229 Leonberg Böblinger Straße 25 Phone: 07152/972-6 Fax: 07152/972-727		INDRAMAT GmbH Phone: D-0172/660 040 6 -or- Phone: D-0171/333 882 6

Customer service points in Germany

Europe

Austria	Austria	Belgium	Denmark
G.L.Rexroth Ges.m.b.H. Geschäftsbereich INDRAMAT A-1140 Wien Hägelingasse 3 Phone: 1/9852540-400 Fax:1/9852540-93	G.L.Rexroth Ges.m.b.H. Geschäftsbereich INDRAMAT A-4061 Pasching Randlstraße 14 Phone: 07229/4401-36 Fax: 07229/4401-80	Mannesmann Rexroth N.V.-S.A. Geschäftsbereich INDRAMAT B-1740 Ternat Industrielaan 8 Phone: 02/5823180 Fax: 02/5824310	BEC Elektronik AS DK-8900 Randers Zinkvej 6 Phone: 086/447866 Fax: 086/447160
England	**Finland**	**France**	**France**
Mannesmann Rexroth Ltd. INDRAMAT Division Cirencester, Glos GL7 1YG 4 Esland Place, Love Lane Phone: 01285/658671 Fax: 01285/654991	Rexroth Mecman OY SF-01720 Vantaa Riihimiehentie 3 Phone: 0/848511 Fax: 0/846387	Rexroth - Sigma S.A. Division INDRAMAT F-92632 Gennevilliers Cedex Parc des Barbanniers 4, Place du Village Phone: 1/41475430 Fax: 1/47946941	Rexroth - Sigma S.A. Division INDRAMAT F-69634 Venissieux - Cx 91, Bd 1 Joliot Curie Phone: 78785256 Fax: 78785231
France	**Italy**	**Italy**	**Netherlands**
Rexroth - Sigma S.A. Division INDRAMAT F-31100 Toulouse 270, Avenue de lardenne Phone: 61499519 Fax: 61310041	Rexroth S.p.A. Divisione INDRAMAT I-20063 Cernusco S/N.MI Via G. Di Vittoria, 1 Phone: 02/92365-270 Fax: 02/92108069	Rexroth S.p.A. Divisione INDRAMAT Via Borgomanero, 11 I-10145 Torino Phone: 011/7712230 Fax: 011/7710190	Hydraudyne Hydrauliek B.V. Kruisbroeksestraat 1a P.O. Box 32 NL-5280 AA Boxtel Phone: 04116/51951 Fax: 04116/51483
Spain	**Spain**	**Sweden**	**Switzerland**
Rexroth S.A. Centro Industrial Santiago Obradors s/n E-08130 Santa Perpetua de Mogoda (Barcelona) Phone: 03/718 68 51 Telex: 591 81 Fax: 03/718 98 62	Goimendi S.A. División Indramat Jolastokieta (Herrera) Apartado 11 37 San Sebastion, 20017 Phone: 043/40 01 63 Telex: 361 72 Fax: 043/39 93 95	AB Rexroth Mecman INDRAMAT Division Varuvägen 7 S-125 81 Stockholm Phone: 08/727 92 00 Fax: 08/64 73 277	Rexroth SA Département INDRAMAT Chemin de l`Ecole 6 CH-1036 Sullens Phone: 021/731 43 77 Fax: 021/731 46 78
Switzerland	**Russia**		
Rexroth AG Geeschäftsbereich INDRAMAT Gewerbestraße 3 CH-8500 Frauenfeld Phone: 052/720 21 00 Fax: 052/720 21 11	Tschudnenko E.B. Arsenia 22 153000 Ivanovo Rußland Phone: 093/22 39 633		

European customer service points without Germany

Outside Europe

Argentina Mannesmann Rexroth S.A.I.C. Division INDRAMAT Acassusso 48 41/7 1605 Munro (Buenos Aires) Argentina Phone: 01/756 01 40 01/756 02 40 Telex: 262 66 rexro ar Fax: 01/756 01 36	Argentina Nakase Asesoramiento Tecnico Diaz Velez 2929 1636 Olivos (Provincia de Buenos Aires) Argentina Argentina Telefon 01/790 52 30	Australia Australian Industrial Machenery Services Pty. Ltd. Unit 3/45 Horne ST Campbellfield VIC 2061 Australia Phone: 03/93 59 0228 Fax: 03/93 59 02886	Brazil Mannesmann Rexroth Automação Ltda. Divisão INDRAMAT Rua Georg Rexroth, 609 Vila Padre Anchieta BR-09.951-250 Diadema-SP Caixa Postal 377 BR-09.901-970 Diadema-SP Phone: 011/745 90 65 011/745 90 70 Fax: 011/745 90 50
Canada Basic Technologies Corporation Burlington Division 3426 Mainway Drive Burlington, Ontario Canada L7M 1A8 Phone: 905/335-55 11 Fax: 905/335-41 84	China Rexroth (China) Ldt. Shanghai Office Room 206 Shanghai Intern. Trade Centre 2200 Yanan Xi Lu Shanghai 200335 P.R. China Phone: 021/627 55 333 Fax: 021/627 55 666	China Rexroth (China) Ldt. Shanghai Parts & Service Centre 199 Wu Cao Road, Hua Cao Minhang District Shanghai 201 103 P.R. China Phone: 021/622 00 058 Fax: 021/622 00 068	China Rexroth (China) Ldt. 1430 China World Trade Centre 1, Jianguomenwai Avenue Beijing 100004 P.R. China Phone: 010/50 50 380 Fax: 010/50 50 379
China Rexroth (China) Ldt. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023 P.R. China Phone: 0411/46 78 930 Fax: 0411/46 78 932	Hong-Kong Rexroth (China) Ldt. 19 Cheung Shun Street 1st Floor, Cheung Sha Wan, Kowloon, Honkong Phone: 741 13 51/-54 und 741 14 30 Telex: 3346 17 GL REX HX Fax: 786 40 19 786 07 33	India Mannesmann Rexroth (India) Ltd. INDRAMAT Division Plot. 96, Phase III Peenya Industrial Area Bangalore - 560058 Phone: 80/839 21 01 80/839 73 74 Telex: 845 5028 RexB Fax: 80/839 43 45	Japan Rexroth Co., Ltd. INDRAMAT Division I.R. Building Nakamachidai 4-26-44 Tsuzuki-ku, Yokohama 226 Japan Phone: 045/942-72 10 Fax: 045/942-03 41
Korea Rexroth-Seki Co Ltd. 1500-12 Da-Dae-Dong Saha-Gu, Pusan, 604-050 Phone: 051/264 90 01 Fax: 051/264 90 10	Korea Seo Chang Corporation Ltd. Room 903, Jeail Building 44-35 Yoido-Dong Youngdeungpo-Ku Seoul, Korea Phone: 02/780-82 07 ~9 Fax: 02/784-54 08	Mexico Motorización y Diseño de Controles, S.A. de C.V. Av. Dr. Gustavo Baz No. 288 Col. Parque Industrial la loma Apartado Postal No. 318 54060 Tlalnepantla Estado de Mexico Phone: 5/397 86 44 Fax: 5/398 98 88	
USA Rexroth Corporation INDRAMAT Division 5150 Prairie Stone Parkway Hoffman Estates, Illinois 60192 Phone: 847/645-36 00 Fax: 857/645-62 01	USA Rexroth Corporation INDRAMAT Division 2110 Austin Avenue Rochester Hills, Michigan 48309 Phone: 810/853-82 90 Fax: 810/853-82 90		

Customer service points outside Europe

Notes


Indramat