SINUMERIK 810T/820T
Basic Version 3
Software Version 3

Part 2: Programming

User Documentation
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fundamentals of Programming</td>
<td>1</td>
</tr>
<tr>
<td>Directions of Movement, Dimensional Notation</td>
<td>2</td>
</tr>
<tr>
<td>Programming of Motion Blocks</td>
<td>3</td>
</tr>
<tr>
<td>Miscellaneous, Switching and Auxiliary Functions</td>
<td>4</td>
</tr>
<tr>
<td>Subroutines</td>
<td>5</td>
</tr>
<tr>
<td>Parameters</td>
<td>6</td>
</tr>
<tr>
<td>Contour Definition</td>
<td>7</td>
</tr>
<tr>
<td>Tool Offsets</td>
<td>8</td>
</tr>
<tr>
<td>Tool Nose Radius Compensation (TNRC)</td>
<td>9</td>
</tr>
<tr>
<td>Cycles</td>
<td>10</td>
</tr>
<tr>
<td>Programming of Cycles</td>
<td>11</td>
</tr>
<tr>
<td>SINUMERIK 810T/820T Program Key</td>
<td>12</td>
</tr>
</tbody>
</table>
# Contents

1  **Fundamentals of Programming** ........................................ 1-1
   1.1  Program structure .................................................. 1-1
   1.2  Block format ....................................................... 1-2
   1.3  Block elements .................................................... 1-3
   1.3.1  Main blocks and subblocks ................................... 1-3
   1.3.2  Skippable blocks ................................................ 1-3
   1.3.3  Remarks (comments) ............................................ 1-4
   1.4  Word format ........................................................ 1-4
   1.5  Character set ........................................................ 1-7
   1.6  Tapes ................................................................. 1-7
   1.6.1  Tape reader ....................................................... 1-7
   1.6.2  Tape code ........................................................ 1-8
   1.6.3  Leader ............................................................ 1-8
   1.6.4  Read-in stop ...................................................... 1-8
   1.7  Program format for input/output ................................ 1-9
   1.8  Code table .......................................................... 1-13
   1.9  Input/output formats .............................................. 1-16
   1.10  Diagrams ............................................................. 1-20
   1.10.1  Revolutional feedrate limit data ............................ 1-20
   1.10.2  Spindle speed as a function of turning radius with \( v = \text{constant} \) ........ 1-21
   1.11  Channel structure ............................................... 1-22

2  **Directions of Movement, Dimensional Notation** .................. 2-1
   2.1  Coordinate system .................................................. 2-1
   2.1.1  Flexible plane selection ....................................... 2-2
   2.2  Position data, preparatory functions ............................ 2-4
   2.3  Dimension systems: absolute and incremental position data G90/G68/G91 2-4
   2.4  Reference points .................................................... 2-6
   2.5  Zero offset .......................................................... 2-7
   2.6  Path calculation .................................................... 2-11
   2.7  Workpiece dimensioning, input system G70/G71 .................. 2-12
   2.8  Mirroring ............................................................. 2-13
   2.9  Programmable working area limitation G25/G26 .................. 2-15
   2.10  Software cam ....................................................... 2-17
   2.11  Coordinate rotation .............................................. 2-19
   2.12  Scale modification: Selection G51, cancellation G50 .......... 2-22

3  **Programming of Motion Blocks** ..................................... 3-1
   3.1  Axis commands ..................................................... 3-1
   3.1.1  Axis motion without machining G00 .......................... 3-3
   3.1.2  Axis duplication ................................................ 3-4
   3.1.2.1  Function ...................................................... 3-4
   3.1.2.2  Complete machining ......................................... 3-7
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.6</td>
<td>Data transfer, system memory to R parameters</td>
<td>11-11</td>
</tr>
<tr>
<td>11.7</td>
<td>Data transfer, R parameters to system memory</td>
<td>11-17</td>
</tr>
<tr>
<td>11.8</td>
<td>Mathematical functions</td>
<td>11-23</td>
</tr>
<tr>
<td>11.9</td>
<td>NC-specific functions</td>
<td>11-29</td>
</tr>
<tr>
<td>11.10</td>
<td>I/O functions</td>
<td>11-37</td>
</tr>
<tr>
<td>11.11</td>
<td>Operator guidance macro (OGM)</td>
<td>11-42</td>
</tr>
<tr>
<td>11.12</td>
<td>@ code table</td>
<td>11-44</td>
</tr>
<tr>
<td><strong>12</strong></td>
<td><strong>SINUMERIK 810T/820T Program Key</strong></td>
<td>12-1</td>
</tr>
<tr>
<td>12.1</td>
<td>Internal G groups for @36b</td>
<td>12-1</td>
</tr>
<tr>
<td>12.2</td>
<td>Program key</td>
<td>12-2</td>
</tr>
</tbody>
</table>
1 Fundamentals of Programming

1.1 Program structure

The program structure is based on DIN 66025.
A part program comprises a complete string of blocks which define the sequence of operations of a machining process on a numerically controlled machine tool.

A part program comprises:

- The character for program start
- A number of blocks
- The character for program end.

The character for program start precedes the first block in the part program. The character for part program end is contained in the last block of the part program.

<table>
<thead>
<tr>
<th>Block:</th>
<th>Leader</th>
<th>%</th>
<th>MPF&lt;No.&gt;</th>
<th>L_F</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd block</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>L_F</td>
</tr>
<tr>
<td>3rd block</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>4th block</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>L_F</td>
</tr>
<tr>
<td>last block</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>M02/M30</td>
</tr>
</tbody>
</table>

W = Word
L_F = Line Feed
MPF = Main program file
MPF<No.> = Main program number

Subroutines and cycles may be components of the program. Cycles are subroutines which have been created either by the machine manufacturer or by Siemens. They can be specially protected against misuse.

Up to 200 part programs and subroutines may be stored simultaneously in the program memory.

The input sequence is arbitrary. For part programs, a total of
0 – 9999 machining programs and
1 – 9999 subroutines are available.
If the program is entered by means of operator prompting via the operator panel, when the “Block number” softkey is actuated, the block numbers are generated automatically in steps of five. The “Cancel” key can be used to delete the entered block number; the “Edit” key can be used to overwrite it.

### 1.2 Block format

A block contains all data required to implement an operating procedure. The block comprises several words and the “block end” character.

The block length is max. 120 characters. The block is displayed in its entirety over several lines.

The block number is entered under address N or with “:”. Block numbers are freely selectable. A defined block search and defined jump functions can only be guaranteed if a block number is used no more than once in a program.

Programming without a block number is permissible. In this case, however, no block search or jump functions will be possible.

The block format should be made as simple as possible by arranging the words of a block in the program key sequence.

**Block example:**

```
```

- **N** Address of block number
- **925** Block number
- **G..** Preparatory function
- **X.. Z..** Position data
- **F..** Feedrate
- **S..** Spindle speed
- **T..** Tool number
- **M..** Miscellaneous function
- **H..** Miscellaneous function
- **LF** Block end

If the value for an address letter is programmed more than once, the last value to have been programmed applies.

Each block must be terminated with the “LF” end-of-block character. This character appears on the screen as the special character LF. When the program is printed out, this character does not appear.
1.3 Block elements

1.3.1 Main blocks and subblocks

There are two types of blocks: Main blocks and subblocks.

The main block must contain all words required to start the machining cycle in the program section beginning there. A main block may only be located in the part program (main program). A main block may be identified by means of the “:” character instead of address character “N” for the subblock.

Block example:

:10 G1 X10 Z-15 F200 S1000 M03 

A subblock contains only those functions which differ from the functions in the previous block.

Block example:

N15 Z20 

A main block and several subblocks together form a program section.

Example: :

N105  
N110  
N115  

1.3.2 Skippable blocks

Program blocks which must not be executed during every program run can be skipped by entering the slash character “/” in front of the word with the block number. Skipping blocks is activated via the “SKIP YES-NO” softkey or via the interface controller. The skipped blocks must form a loop (with start and end at the same point) or the program may be executed incorrectly. A section can be skipped by skipping several consecutive blocks.

Skipped block:

N...  
N...  
/N...  
N...  
N...  

Processing sequence

Skipping blocks
The rapid block change times mean that several blocks must be buffered. If the machine stops on account of M00 (programmed stop), the next blocks will already have been read in advance. The “skip” function is only active on those blocks which have not been buffered. This buffering can be prevented by programming L999 (disable pre-reading in @ 714) after the block containing M00.

1.3.3 Remarks (comments)

The blocks in a program can be explained by means of remarks. A remark permits instructions for the operator to be displayed on the screen. The text of a remark is enclosed between the start-of-remark character “("and the end-of-remark character")”.

The remark must not contain the percent sign %, an end-of-block character LF, or bracket "(".""").

A remark may be up to 120 characters in length. Up to 41 of these may be displayed in the comment line of the screen.

It is advisable to write the remark at the end of the block or in a separate line. The remark must never be located between the address and a digit or between a word and the corresponding parameter!

Right:

| N05 | G00 X100 Z200 (Position) LF |
| N10 | G01 X100+R1 Z200 (Machine) LF |
| N15 | .... |
|x | Address |
| 100 | Digits |
|R1 | R parameter |
| ( | Start of remark |
|MACHINE | Remark |
| ) | End of remark |

Wrong:

| N05 | X (Position) 100 Z200 LF |
| N10 | X100+ (Machine) R1 Z200 LF |

1.4 Word format

A word is an element of a block. It comprises an address character and a string of digits. The address character is normally a letter. The string of digits may be specified with a sign and with decimal points. The sign is written between the address letters and the string of digits. A positive sign may be omitted.
**Word format**

**Examples:**

- **G91** or **M30**
  - **G** Address
  - **91** Numerical value, signifying "Incremental dimensions"
  - **M** Address
  - **30** Numerical value, signifying "Program end"

**Extended address**

<table>
<thead>
<tr>
<th>Address characters</th>
<th>Digits</th>
<th>=</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended address</td>
<td></td>
<td></td>
<td>Word</td>
</tr>
<tr>
<td>Address characters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Axis</td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>- Spindle</td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>- M function (group 5)</td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>Number:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Axis no.</td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>- Spindle no.</td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>- Channel no.</td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>&quot;=&quot;</td>
<td></td>
<td></td>
<td>to separate the address extension digit from the numerical value</td>
</tr>
</tbody>
</table>

**Example:**

- **Q1= 100** 1st auxiliary axis
- **M1= 19** Oriented spindle stop of 1st spindle
- **M2= 100** M function 100 is output for channel 2

**Note:**

- **G1 X1 = Z L_F** means: **G1 X1=0 Z0 L_F**
- **G16 X Y1 = Z ±L_F** means: **G16 X0 Y1 = 0 Z0 ± L_F**
  - Plane selection with extended address (Y1)

"±" specifies the direction of the tool offset and must be placed after the axis.
"=" must be written to define the figure after the address as an address extension.

The word format is based on DIN 66025.
Shorthand notation of words:

%4 N04 G02/G03 D03 XL+053 ZL+053 QL+053 AL053 ID053 KD053
F05 L03/L04 S05 T08 R03 RL+053 BD033 M04 H08 P02 LF

Definitions:

First letter Address
Second letter L Absolute/incremental
Second letter D Incremental
Character ± Absolute dimensions with positive or negative sign
First digit 0 Leading zeros may be omitted: Variable word length
          (G01=G1)
Second digit Decades Positions in digit string
Second and Decades Digit string positions before and after decimal point
and third digit (coordinate values X,Z,I,K in mm)
Character LF Block end

Example:

x Address
L Absolute/incremental
± Sign
0 Leading zeros may be omitted
5 Number of positions before decimal point
3 Number of positions after decimal point

Word examples:

x–12345.531 G9

x Address g Address
– Sign 9 Digit
12345 Digits . Decimal point
531 Digits

Decimal point input:

<table>
<thead>
<tr>
<th>Value</th>
<th>Programmed value with decimal point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 µm</td>
<td>X.0001</td>
</tr>
<tr>
<td>1 µm</td>
<td>X.001</td>
</tr>
<tr>
<td>10 µm</td>
<td>X.01</td>
</tr>
<tr>
<td>100 µm</td>
<td>X.1</td>
</tr>
<tr>
<td>1000 µm</td>
<td>X1 or X1.</td>
</tr>
<tr>
<td>10200 µm</td>
<td>X10.2</td>
</tr>
</tbody>
</table>

Decimal point input is permissible for the following addresses:


For address “R” only the notation with an extended address is valid: R10 = 50.0
(see Section 12 Program key for restrictions on S).

Leading and trailing zeros need not be written when decimal point notation is used.
1.5 Character set

It is always possible to choose between two codes for programming:

- DIN 66025 (ISO)
- EIA-244-B.

The examples used in these instructions are based on the ISO code.

The following characters are available in ISO code for formulating program, geometric and process statements:

**Address letters:**

**Lower-case letters**
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

**Digits**
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

The 5th axis must be entered with extended address, e.g. Q1 = 5.

**Hexadecimal digits with CL 800 machine code**
a, b, c, d, e, f (Cf. CL 800 Configuring Instructions)

**Letter**
D (Input of tool offset [TO - TOOL OFFSET])

**Printable special characters**
%, (, ), +, -, /, :, ., =, *, @

**Data input**
The following characters are not processed or stored:

- HT = Horizontal tabulator
- SP = Space
- DEL = Delete character
- CR = Carriage return

Other control characters are shown in the code table.

**Data output**
The following characters are generated:

- SP (after every word)
- CR generated twice after LF or once before LF (setting data)

1.6 Tapes

1.6.1 Tape reader

The tape reader must be matched to the controller. The data transfer rate and the transfer format (see Universal Interface) are defined via the setting data.
### 1.6.2 Tape code

The data on tape is coded according to fixed rules, i.e., each hole combination corresponds to a particular character. **Two tape codes** are used: ISO or EIA (see code table).

All characters of a code have a common identification:

- ISO always an even number of holes
- EIA always an odd number of holes

The controller automatically recognizes the correct code as soon as it reads the first % (ISO) or EOR (EIA). The criterion relating to an odd or even number of holes is used - starting at the second character of the program - for a character parity check, which has an error detection rate for single errors of 100%.

Each tape must be written in one of the permissible codes. It is not permissible to change the code within a tape or to splice tapes together; this will cause the character parity check to be initiated.

As a further check a complete program comparison is performed if a program already stored in the program memory is read in again. On detection of an error the read-in process is halted and the error displayed on the CRT display unit.

### 1.6.3 Leader

The leader is used to identify the programs. The tape leader may include all characters except the start-of-program character (% character). The leader is not stored, and is ignored by the controller during program processing.

<table>
<thead>
<tr>
<th>SHAFT</th>
<th>%</th>
<th>MPF</th>
<th>1579</th>
<th>L_E</th>
</tr>
</thead>
</table>

### 1.6.4 Read-in stop

The read-in process is halted by M02, M30 or M17 if no central end-of-transmission character has been defined.

If an end-of-transmission character has been specified in the setting data, the program or data block end (M02, M17, M30) will not stop the reader during reading in of the tape. The read-in process is not halted until the end-of-transmission character is reached.

**Setting-Data:**

- Individual stop
- Central stop

[End character] e.g. $
1.7 Program format for input/output

<table>
<thead>
<tr>
<th>Program</th>
<th>Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>%MPF1235 LF</td>
<td>Part program 1235 (MAIN PROGRAM FILE)</td>
</tr>
<tr>
<td>(Perform measurem.)</td>
<td>Remark</td>
</tr>
<tr>
<td>N...LF</td>
<td>Part program</td>
</tr>
<tr>
<td>N...LF</td>
<td></td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>Part program end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subroutines</th>
<th>Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>%SPF234 LF</td>
<td>Subroutine 234 (SUB PROGRAM FILE)</td>
</tr>
<tr>
<td>N5...LF</td>
<td>Subroutine</td>
</tr>
<tr>
<td>N10...LF</td>
<td></td>
</tr>
<tr>
<td>(Bore cycle)...LF</td>
<td>Remark</td>
</tr>
<tr>
<td>M17 LF</td>
<td>Subroutine end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%ZOALF</th>
<th>Settable zero offsets (ZERO OFFSET ACTIVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G154 X=... Y=... LF</td>
<td>1st to 4th settable offset (coarse)</td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>G157 X=... Y=... LF</td>
<td></td>
</tr>
<tr>
<td>G254 X=... Y=... LF</td>
<td>1st to 4th settable offset (fine)</td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>G257 X=... Y=... LF</td>
<td></td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>Zero offset block data block end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%TEA1 LF</th>
<th>NC machine data (TESTING DATA ACTIVE 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N...=...LF</td>
<td>Machine data</td>
</tr>
<tr>
<td>N...=...LF</td>
<td></td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>Machine data block data block end</td>
</tr>
<tr>
<td>%TEA2 LF</td>
<td>PLC machine data (TESTING DATA ACTIVE 2)</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>N...=...LF</td>
<td>Machine data</td>
</tr>
<tr>
<td>N...=...LF</td>
<td>Machine data block data block end</td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>Machine data block data block end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%RPA0..2 LF</th>
<th>(R PARAMETER ACTIVE ) Channel No. ( 0=central R parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R...=...LF</td>
<td>Parameter numbers with value assignments (channels 1 and 2, channel specific)</td>
</tr>
<tr>
<td>R...=...LF</td>
<td>R parameter data block end</td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>R parameter data block end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%TOA LF</th>
<th>Tool offsets (TOOL OFFSET ACTIVE )</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO range (MD)</td>
<td>Tool offsets (number of parameters in acc. with MD)</td>
</tr>
<tr>
<td>D1 P0=...P1=...P9=...LF</td>
<td>Tool offset block data block end</td>
</tr>
<tr>
<td>D2 P0=...P1=...LF</td>
<td>Tool offset block data block end</td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>Tool offset block data block end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%SEA LF</th>
<th>(SETTING DATA ACTIVE) (general setting data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N...=...LF</td>
<td>Address with value assignment (0...9, 2000...2003, 3000...3171, 4000...4033, 5000...5771)</td>
</tr>
<tr>
<td>N...=...LF</td>
<td>Setting data data block end</td>
</tr>
<tr>
<td>M02 LF or M30 LF</td>
<td>Setting data data block end</td>
</tr>
</tbody>
</table>

---

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
## %PCA L_F

<table>
<thead>
<tr>
<th>%PCA L_F</th>
<th>PLC alarm texts/operator messages (PROGRAMMABLE LOGIC CONTROL ALARM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N6000 (Text ...) L_F</td>
<td>PLC alarm texts (6000 - 6063)</td>
</tr>
<tr>
<td>N6063 (Text ...) L_F</td>
<td></td>
</tr>
<tr>
<td>N7000 (Text ...) L_F</td>
<td>PLC operator messages (7000 - 7063)</td>
</tr>
<tr>
<td>N7063 (Text ...) L_F</td>
<td></td>
</tr>
<tr>
<td>M02 L_F or M30 L_F</td>
<td>PLC text data block end</td>
</tr>
</tbody>
</table>

## %PCP L_F

<table>
<thead>
<tr>
<th>%PCP L_F</th>
<th>PLC program (PROGRAMMABLE LOGIC CONTROL PROGRAM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7070 8005...</td>
<td>Machine code</td>
</tr>
<tr>
<td>M02 L_F or M30 L_F</td>
<td>PLC program data block end</td>
</tr>
</tbody>
</table>

## %UMS L_F

<table>
<thead>
<tr>
<th>%UMS L_F</th>
<th>User Memory Submodule</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Hexcode</td>
<td>Configured data</td>
</tr>
<tr>
<td>:Hexcode</td>
<td>Configured data</td>
</tr>
<tr>
<td>M02 L_F or M30 L_F</td>
<td>User memory submodule data block end</td>
</tr>
</tbody>
</table>
Memory areas:
The memory areas of the SINUMERIK 810T/820T controller are addressed by means of the following identifiers:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPF</td>
<td>Part program (Main Program File)</td>
</tr>
<tr>
<td>SPF</td>
<td>Subroutine (Sub Program File)</td>
</tr>
<tr>
<td>TOA</td>
<td>Tool offsets (Tool Offset Active)</td>
</tr>
<tr>
<td>ZOA</td>
<td>Zero offsets (Zero Offset Active)</td>
</tr>
<tr>
<td>TEA1</td>
<td>NC machine data (Testing Data Active 1)</td>
</tr>
<tr>
<td>TEA2</td>
<td>PLC machine data (Testing Data Active 2)</td>
</tr>
<tr>
<td>PCA</td>
<td>PLC Alarm Texts</td>
</tr>
<tr>
<td>PCP</td>
<td>PLC program (machine code) (Programmable Control Program)</td>
</tr>
<tr>
<td>RPA</td>
<td>R parameter numbers with value assignments (R Parameter Active)</td>
</tr>
<tr>
<td>SEA</td>
<td>Addresses with value assignments (Setting Data Active)</td>
</tr>
<tr>
<td>CLF</td>
<td>Clear statement (Clear File)</td>
</tr>
<tr>
<td>ASM</td>
<td>User memory submodule</td>
</tr>
</tbody>
</table>

- **Deleting programs:**

These functions permit part programs and subroutines to be deleted in any sequence via the input/output interface.

```plaintext
DELETE PROGRAM Leader
%CLF L F Delete program identifier (CLEAR FILE)

MPF1234 L F Delete part program %1234
MPF 1, 1200 L F Delete part program %1 to %1200
MPF 0, 9999 L F Delete all part programs
SPF 10 L F Delete subroutine L10
SPF 11, 79 L F Delete subroutines L11 to L79
SPF 1, 9999 L F Delete all subroutines
M30, M02 oder M17 L F End identifier M30, M02 or M17
```

- **Deleting text data:**

```plaintext
%PCA L F
M02 or M30 L F
```
### 1.8 Code table

**ISO/DIN 66024 extended**

<table>
<thead>
<tr>
<th>Character</th>
<th>Hole Combination</th>
<th>Only leader and remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>P 7 6 5 4 T 3 2 1</td>
<td><img src="image" alt="Hole Combination" /></td>
<td><img src="image" alt="Legend" /></td>
</tr>
<tr>
<td>NUL</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>SOH</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>STX</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>ETX</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>EOT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ENQ</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ACK</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>BEL</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>BS</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>HT</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>VT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FF</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>CR</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>SO</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>SI</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>DLE</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>DC1</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>DC2</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>DC3</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>DC4</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>NAK</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>SYN</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>ETB</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>CAN</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>EM</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>SUB</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>ESC</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>FS</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>GS</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>RS</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>US</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>SP</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>LF</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>!</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>&quot;</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>$</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>%</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>&amp;</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>'</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>(</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>)</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>+</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>.</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>/</td>
<td>*</td>
<td>-</td>
</tr>
</tbody>
</table>
## ISO/DIN 66024 extended

<table>
<thead>
<tr>
<th>Character</th>
<th>Hole Combination</th>
<th>Only leader and remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>`</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>a</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>h</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>i</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>j</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>k</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>l</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>m</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>n</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>o</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>p</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>q</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>r</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>s</td>
<td>* * * *</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>u</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>v</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>w</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>y</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>z</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>{</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>:</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>)</td>
<td>* * * *</td>
<td>x</td>
</tr>
<tr>
<td>DEL</td>
<td>* * * *</td>
<td></td>
</tr>
</tbody>
</table>

% is not allowed in the tape leader.
Not all ISO characters can be represented in EIA code. Consequently, discrepancies may occur when comparing a program generated in ISO code and stored in the NC with its equivalent program converted to EIA code.

The following functions are no longer capable of operating when read into the SINUMERIK controller once more:

- Parameter calculation
- Extended address
- @ commands with HEX digits (@ 36 a)
- Special characters
- Comments.

The EIA code for "@" and ":" can be set in setting data (see Section 6.2 “Setting data for description of the interfaces” of Part 1 “Operating”).
## 1.9 Input/output formats

The input/output formats depend on the machine manufacturer's machine data setting.

### Input resolution:
- 0.01 mm or position control resolution
- 0.005 mm
- 0.001 inch
- 0.0005 inch
- 0.001 degrees
- 0.005 degrees

<table>
<thead>
<tr>
<th>Significance Addresses</th>
<th>Metric</th>
<th>Inch</th>
<th>Degrees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Range</td>
<td>Unit</td>
<td>Range</td>
</tr>
<tr>
<td>Position data (linear axes)</td>
<td>±0.01 to 9999.99</td>
<td>±0.001 to 9999.99</td>
<td>—</td>
</tr>
<tr>
<td>Interpolation parameters</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Position data for G91 (rotary axes)</td>
<td>—</td>
<td>—</td>
<td>0.001 to 9999.999</td>
</tr>
<tr>
<td>Position data for G90 (rotary axes)</td>
<td>—</td>
<td>—</td>
<td>±0.001 to 359.999</td>
</tr>
<tr>
<td>Chamfer (U-); radius (U)</td>
<td>0.01 to 9999.99</td>
<td>0.001 to 9999.999</td>
<td>—</td>
</tr>
<tr>
<td>Zero offset</td>
<td>±0.01 to 9999.999</td>
<td>±0.001 to 9999.999</td>
<td>±0.001 to 359.999</td>
</tr>
<tr>
<td>Thread lead</td>
<td>0.01 to 4000.00</td>
<td>0.001 to 160.000</td>
<td>—</td>
</tr>
<tr>
<td>Spindle speed S (value determined via commissioning setting)</td>
<td>1-16000</td>
<td>1 min⁻¹</td>
<td>1-1600.0</td>
</tr>
<tr>
<td>Linear feedrate (F) (G94)</td>
<td>0.1 to 450000</td>
<td>mm/min</td>
<td>0.01 to 17700</td>
</tr>
<tr>
<td>Feedrate per revolution (F) (G95)</td>
<td>0.01 to 500.00 ¹</td>
<td>mm/rev</td>
<td>0.001 to 20.000 ¹</td>
</tr>
<tr>
<td>Tool offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>±0.01 to 9999.99</td>
<td>±0.001 to 999.999</td>
<td>inch</td>
</tr>
<tr>
<td>Radius</td>
<td>±0.01 to 999.99</td>
<td>±0.01 to 999.99</td>
<td></td>
</tr>
<tr>
<td>Dwell</td>
<td>X</td>
<td>0.01 to 99999.999</td>
<td>sec</td>
</tr>
<tr>
<td>F</td>
<td>0.01 to 99999.999</td>
<td>0.01 to 99999.999</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>0.1 to 99.9</td>
<td>Revolutions</td>
<td>0.1 to 99.9</td>
</tr>
<tr>
<td>Angle in contour definition (A)</td>
<td>—</td>
<td>—</td>
<td>0 to 359.99999</td>
</tr>
<tr>
<td>Angle in oriented spindle stop (M19)</td>
<td>—</td>
<td>—</td>
<td>0.1 to 359.9</td>
</tr>
<tr>
<td>R parameters</td>
<td>Dimension depending on association (internal floating point) all combinations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

1) The maximum speed with linear feed (G94) must not be exceeded.
2) The limit values apply to MD 155 = 2
### Input resolution:
- 0.001 mm or position control resolution
- 0.0005 mm
- 0.0001 inch
- 0.00005 inch
- 0.001 degrees
- 0.0005 degrees

<table>
<thead>
<tr>
<th>Significance Addresses</th>
<th>Metric</th>
<th>Inch</th>
<th>Degrees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Range</td>
<td>Unit</td>
<td>Range</td>
</tr>
<tr>
<td>Position data (linear axes)</td>
<td>±0.001 to 99999.999</td>
<td>±0.0001 to 99999.999</td>
<td>(\pm 0.001) to 359.999</td>
</tr>
<tr>
<td>Interpolation parameters</td>
<td>0.001 to 99999.999</td>
<td>0.0001 to 99999.999</td>
<td>-</td>
</tr>
<tr>
<td>Position data for G91 (rotary axes)</td>
<td>-</td>
<td>-</td>
<td>0.001 to 99999.999</td>
</tr>
<tr>
<td>Position data for G90 (rotary axes)</td>
<td>-</td>
<td>mm</td>
<td>(\pm 0.0001) to 359.999</td>
</tr>
<tr>
<td>Chamfer (U-); radius (U)</td>
<td>0.001 to 99999.999</td>
<td>0.0001 to 99999.999</td>
<td>(\pm 0.001) to 359.999</td>
</tr>
<tr>
<td>Zero offset</td>
<td>±0.0001 to 99999.999</td>
<td>±0.001 to 99999.999</td>
<td>(\pm 0.0001) to 359.999</td>
</tr>
<tr>
<td>Thread lead</td>
<td>0.001 to 400.000</td>
<td>0.0001 to 16.000</td>
<td>-</td>
</tr>
<tr>
<td>Spindle speed S (value determined via commissioning setting)</td>
<td>1 - 16000</td>
<td>1 min (^{-1})</td>
<td>1 - 16000</td>
</tr>
<tr>
<td></td>
<td>0.1-1600.0</td>
<td>0.1 min (^{-1})</td>
<td>0.1-1600.0</td>
</tr>
<tr>
<td>Linear feedrate (F) (G94) (^2)</td>
<td>0.01 to 45000</td>
<td>mm/min</td>
<td>0.001 to 1770</td>
</tr>
<tr>
<td>Feedrate per revolution (F) (G95)</td>
<td>0.001 to 50,000 (^1)</td>
<td>mm/rev</td>
<td>0.0001 to 2,0000 (^1)</td>
</tr>
<tr>
<td>Tool offset</td>
<td>Length</td>
<td>±0.001 to 99999.999</td>
<td>±0.0001 - 99999.999</td>
</tr>
<tr>
<td></td>
<td>Radius</td>
<td>±0.001 to 999.999</td>
<td>±0.0001 - 99.999</td>
</tr>
<tr>
<td>Dwell</td>
<td>X</td>
<td>0.01 to 99999.999</td>
<td>0.01 to 99999.999</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>0.01 to 99999.999</td>
<td>0.01 to 99999.999</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>0.1 to 99.9</td>
<td>Revolutions</td>
</tr>
<tr>
<td>Angle in contour definition (A)</td>
<td>-</td>
<td>-</td>
<td>0 to 359.99999</td>
</tr>
<tr>
<td>Angle in oriented spindle stop (M19)</td>
<td>-</td>
<td>-</td>
<td>0.1 to 359.9</td>
</tr>
<tr>
<td>R parameters</td>
<td>Dimension depending on association (internal floating point) all combinations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) The maximum speed with linear feed (G94) must not be exceeded.
\(^2\) The limit values apply to MD 155 = 2.
## Input Resolution

- **Metric**: 0.0001 mm or position control resolution 0.00001 mm
- **Inch**: 0.0001 inch or position control resolution 0.000001 inch
- **Degrees**: 0.001 degrees or position control resolution 0.00001 degrees

| Significance Addresses | Metric | | Inch | | Degrees |
|------------------------|--------|----------|----------|----------|
| **Position data (linear axes)** | Range | Unit | Range | Unit | |
| Interpolation parameters | ±0.0001 to 99999.999 | mm | ±0.0001 to 999.9999 | inch | |
| **Position data for G91** | | | | | Degree |
| (rotary axes) | 0.001 to 99999.999 |
| **Position data for G90** | | | | | Degree |
| (rotary axes) | 0.001 to 99999.999 |
| **Chamfer (U-); radius (U)** | 0.0001 to 99999.999 | mm | ±0.001 to 359.999 |
| **Zero offset** | ±0.0001 to 9999.9999 | inch | ±0.001 to 359.999 |
| **Thread lead** | 0.0001 to 40.0000 | mm | ±0.00001 to 999.9999 |
| **Spindle speed S** | 1 - 16000 | 1 min⁻¹ | ±0.001 to 9000.000 | Degrees/ min |
| (value determined via commissioning setting) | 0.1-1600.0 | 0.1 min⁻¹ | 0.1-1600.0 | |
| **Linear feedrate (F) (G94)** | 0.001 to 9000.000 | mm/min | ±0.0001 to 99999.999 |
| (G95) | 0.0001 to 350.0000 | inch/min | ±0.001 to 99999.999 |
| **Feedrate per revolution (F) (G95)** | 0.0001 to 5.0000 | mm/rev | ±0.0001 to 99999.999 |
| | 0.0001 to 02.0000 | inch/rev | ±0.0001 to 99999.999 |
| **Tool offset** | Length | | Radius | | |
| | ±0.0001 to 999.9999 | mm | ±0.0001 to 99999.999 | inch |
| | ±0.0001 to 999.9999 | mm | ±0.0001 to 99999.999 | |
| **Dwell** | X | sec | 0.01 to 99999.999 | sec |
| | F | sec | 0.01 to 99999.999 | |
| | S | Revolutions | 0.1 to 99.9 | Revolutions |
| **Angle in contour definition (A)** | | | | | |
| | Degree |
| **Angle in oriented spindle stop (M19)** | | | | | |
| | Degree |
| **R parameters** | Dimension depending on association (internal floating point) all combinations | |

---

1. The maximum speed with linear feed (G94) must not be exceeded.
2. The limit values apply to MD 155 = 2.
## Input resolution

<table>
<thead>
<tr>
<th>Metric</th>
<th>Inch</th>
<th>Degrees</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00001 mm</td>
<td>0.00005 inch</td>
<td>0.001 degrees</td>
</tr>
<tr>
<td>0.0001 mm</td>
<td>0.00005 inch</td>
<td>0.001 degrees</td>
</tr>
<tr>
<td>0.001 mm</td>
<td>0.00005 inch</td>
<td>0.001 degrees</td>
</tr>
</tbody>
</table>

### Significance Addresses

<table>
<thead>
<tr>
<th>Position data (linear axes)</th>
<th>Interpolation parameters</th>
<th>±0.0001 to 9999.9999</th>
<th>±0.00001 to 999.99999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Position data for G91 (rotary axes)</th>
<th>-</th>
<th>-</th>
<th>0.001 to 99999.999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Position data for G90 (rotary axes)</th>
<th>-</th>
<th>-</th>
<th>±0.001 to 359.999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Chamfer (U); radius (U)</th>
<th>mm</th>
<th>inch</th>
<th>±0.0001 to 9999.9999</th>
<th>±0.00001 to 999.99999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Zero offset</th>
<th>mm</th>
<th>inch</th>
<th>±0.00001 to 9999.9999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Thread lead</th>
<th>mm</th>
<th>inch</th>
<th>0.0001 to 200.0000</th>
<th>0.00001 to 8.00000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Spindle speed S (value determined via commissioning setting)</th>
<th>1 - 16000</th>
<th>1 min⁻¹</th>
<th>1 - 16000</th>
<th>1 min⁻¹</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Linear feedrate (F) (G94)</th>
<th>mm/min</th>
<th>inch/min</th>
<th>0.001 to 4500.000</th>
<th>0.0001 to 69.000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Feedrate per revolution (F) (G95)</th>
<th>mm/rev</th>
<th>inch/rev</th>
<th>0.0001 to 5.0000</th>
<th>0.0001 to 0.20000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tool offset</th>
<th>Length</th>
<th>mm</th>
<th>±0.0001 to 999.9999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dwell</th>
<th>X</th>
<th>sec</th>
<th>0.01 to 99999.999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>0.1 to 99.9</th>
<th>Revolutions</th>
<th>0.1 to 99.9</th>
<th>Revolutions</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Angle in contour definition (A)</th>
<th>-</th>
<th>-</th>
<th>0 to 359.99999</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Angle in oriented spindle stop (M19)</th>
<th>-</th>
<th>-</th>
<th>0.1 to 359.9</th>
</tr>
</thead>
</table>

### Notes

1) *The maximum speed with linear feed (G94) must not be exceeded.*

2) *The limit values apply to MD 155 = 2.*
1.10 Diagrams

1.10.1 Revolitional feedrate limit data

Assignment of lead and spindle speed (Thread cutting G33)
Assignment of feedrate per revolution and spindle speed

\[ n_{\text{max}1} \] Attainable with encoder 1024 pulses/revolution
\[ n_{\text{max}2} \] Attainable with encoder 512 pulses/revolution
1.10.2 Spindle speed as a function of turning radius with \( v = \text{constant} \)

Rule of thumb:
Given a radius of 160 mm (320 mm dia.)
the circumference is 1 m;
Thus the speed corresponds to the
cutting rate.

Spindle speed \( n \) as a function of the turning radius
(mm at constant cutting rate (\( V = \text{constant} \)).
1.11 Channel structure

The SINUMERIK 810T/820T is fitted with 3 channels. These channels permit the simultaneous processing of two different programs in addition to other structural operations such as program editing and interface operation at the same time as processing in "AUTOMATIC" mode.

The three channels have the following significance:

**Channel 1:** Main channel for processing programs and spindle programming.

**Channel 2:** Auxiliary channel for processing programs for auxiliary axes or for mathematical functions in the background.

**Channel 3:** Graphic simulation for on-screen representation of programs.

In principle all three channels can be operated simultaneously. However, problems of collision arise with a small number of functions.

Functional scope of channels:

The auxiliary channel (channel 2) is a full-grade channel, with the exception of the unavailable functions. Its main function is to perform background calculations or auxiliary motions (tool changes etc.).
The assignment of the axes (in automatic mode) to be traversed in each channel must be performed in the program. The same axis can be moved in channel 1 and channel 2 if the outputting of a travel command simultaneously from the 1st and 2nd channels is excluded (... alarm 180° axis programmed in both channels).

However, the main function of the auxiliary channel is to operate loading axes under PLC control at the same time as the main channel. Given the above-mentioned conditions, however, it is also possible to achieve other options using the auxiliary channel, giving rise to numerous potential applications.

However, since only M functions can be transferred from channel 2 to the PLC, the opportunities for data transfer with the PLC are limited.

Channel 3 is used exclusively for the graphic simulation of a part program. Another part program can be executed concurrently.

(See the notes in Part 1 “Operating”, Section 3.1.13.5 “Shift program” and Section 3.1.14 “SIMULATION”).

Spindle control can be made possible for the 2nd channel via machine data. For this purpose, all functions regarding spindles are also available in the 2nd channel.
2 Directions of Movement, Dimensional Notation

2.1 Coordinate system

The directions of movement of a machine tool are based on a coordinate system allocated to the axes of motion of the machine.

The coordinate system used is clockwise and perpendicular, and has X and Z axes. The system is based on the main axes of the machine.

The coordinate system is defined as follows:

- The second axis is perpendicular to the first axis.

Clockwise coordinate system

The program is the same irrespective of whether the workpiece or the tool is moved during machining.

The default allocations for turning machines are as follows: Main axes X and Z;

For all machines:
Axis addresses are freely selectable via machine data.
2.1.1 Flexible plane selection

Axes can be freely assigned to the G17, G18 and G19 planes via machine data. This assignment must be made by the machine manufacturer. You can define the plane initial setting channel-specifically via machine data.

An example of axis assignment with flexible plane selection

Definition of the planes in the initial setting to DIN:

<table>
<thead>
<tr>
<th>M version</th>
<th>T version</th>
</tr>
</thead>
<tbody>
<tr>
<td>G16</td>
<td>Plane selection with free axis selection</td>
</tr>
<tr>
<td>G17</td>
<td>Plane X – Y (1st axis – 2nd axis)</td>
</tr>
<tr>
<td>G18</td>
<td>Plane X – Z (1st axis – 3rd axis)</td>
</tr>
<tr>
<td>G19</td>
<td>Plane Y – Z (2nd axis – 3rd axis)</td>
</tr>
<tr>
<td></td>
<td>Plane X – Z (1st axis – 2nd axis)</td>
</tr>
<tr>
<td></td>
<td>Plane X – Z (1st axis – 3rd axis)</td>
</tr>
<tr>
<td></td>
<td>Plane Z – Z (2nd axis – 3rd axis)</td>
</tr>
</tbody>
</table>

Find out the three axis names for the axes which form the G17, G18 or G19 planes from the machine manufacturer.

The assignment of the axes to the G17, G18 and G19 planes is also defined by the machine manufacturer. If the axis names are entered incorrectly, alarm 3003 “invalid address” appears after NC start.

You can select the predefined planes G17, G18 and G19 by calling the corresponding G function in the program.

You can also set the initial setting for the plane selection via machine data (NC MD 110*) separately for each channel (see Part 1 “Operating”, Section 7.3.4 “Definition of the initial setting of the G groups”).
Plane selection via G16

Program G16 if the required plane is no longer determined by the axes defined in G17, G18 or G19 or additional definitions are required. This applies to the following functions.

- Parallel axis assumes the function of a main axis
- Additional definition of the tool offset
- Specification of directions of angle heads

**Example 1:** Parallel axis

G16 X Y W
Let G17 plane be defined by axes X,Y,Z,
let the W axis (parallel axis) be parallel to the Z axis.

**Example 2:** Angle head

G16 X Y Z Y
The first two axis addresses (X, Y) define the plane in which the cutter/tool nose radius compensation applies. The fourth axis address specifies the axis with the additional length correction in the previously defined plane.
The 3rd and 4th axis address can be given a negative sign to reverse the direction of the CRC/TNRC.
2.2 Position data, preparatory functions

The position data comprises an axis address and a numerical value, which describes the path on the addressed axis.
If a sign is specified, it is written between the address and the numerical value.
In order to start the positioning procedure, the position data must be supplemented by the preparatory function (G function) and the feedrate (F) data.
The preparatory functions describe the type of machine movement, the type of interpolation and the method of dimensioning.
The **G functions are subdivided into groups** (see Program key, Section 12). A program block may only contain one function from each group.
The G functions are either **modal** or **non modal**:

- The G functions which remain active until they are replaced by a new G function in the same group are said to be modally active.
- The G functions which are only active in the block in which they are contained are said to be active block-by-block. The resets take effect after powering up the controller, a reset or a program end. They need not be programmed.

2.3 Dimension systems: absolute and incremental position data G90/G68/G91

The traversing movement to a particular point in the coordinate system can be described by means of **ABSOLUTE** or **INCREMENTAL** position data.

**Absolute position data input G90**
If absolute position data input is selected, all dimensional inputs refer to a fixed zero, which is normally the **workpiece zero W**.
The value of the associated position data specifies the target position in the coordinate system.

**Absolute dimensioning on direct way (rotary axis only) G68**
With G68 the end of block value is approached on the shortest way (< 180°).
This function is modal. This function has effect only on those axes which have been assigned the "Modulo programming" part function. Except for the part function, G68 has an effect equivalent to G90.

**Incremental position data input G91**
If incremental position data input is selected, the value of the position data corresponds to the path to be traversed. The direction of movement is specified by the sign.
Depending on its current position the tool moves on by the programmed values.
It is possible to switch between absolute and incremental position data input from block to block as desired, since the controller actual value is always referred to the zero point.
A zero offset is calculated for both absolute and incremental programming.

---

Depending on machine data, mixed programming using G90 and G91 in the same block is also possible.
Example: Absolute and incremental position data input

Absolute position data input:

```
N...G00 G90 X40 Z60 LF
```

The tool moves from any position to P2.

Incremental position data input:

```
N...G00 G91 X30 Z40 LF
```

The tool moves from P1 to P2.

Example: Programming with absolute position and incremental position data

Changing between absolute dimensioning G90 and incremental dimensioning G91:

You can change between absolute and incremental dimensioning from block to block.

```
%10 LF
N5  G00 G90 G94 X20 Z30 LF  (P1)
N10 G01 G91 Z40 F100 LF        (P2)
N15 X25 LF                     (P3)
N20 Z-40 LF                    (P4)
N25 X-25 LF                    (P1)
N30 M30 LF                     (P1)
```
2.4 Reference points

The zeros and various reference points are defined on all numerically controlled machine tools.

The machine zero M is the design zero of the machine coordinate system.

The workpiece zero W is the zero defined for programming the workpiece dimensions. It can be freely selected by the programmer. The relationship to the machine origin is defined by the zero offset.

The reference point R is a point defined by the machine manufacturer which is approached when the controller is powered up and which synchronizes the NC control with the machine tool.

The reference dimension is defined in the machine data.

Example: Longitudinal turning (machining before turning centre)

P is the reference point for setting the tool.
Example: "Longitudinal turning (machining before turning centre)"

| P | Tool setting point |
| M | Machine zero       |
| W | Workpiece zero     |
| R | Machine reference point |
| F | Slide reference point |
| WR | Workpiece reference point |
| XMR | Reference point coordinate X |
| ZMR | Reference point coordinate Z |
| XMW | Sum of zero offsets X |
| ZMW | Sum of zero offsets Z |
| XFP | Tool geometry L1 |
| ZFP | Tool geometry L2 |

2.5 Zero offset

Zero offset is the distance between the workpiece (W) (on which the dimensions are based) and the machine zero (M).

The following types of zero offset (ZO) can be activated:

- Settable zero offset (G54 to G57),
- Programmable zero offset (G58, G59),
- External zero offset (from PLC).

The zero offsets apply to the current program.

*It may, however, be defined in machine data that a selected ZO remains active in programs started subsequently.*
Sum of zero offsets =
settable zero offset (G54 to G57) + programmable zero offset (G58, G59) + external zero offset (from PLC).

Settable zero offset G54, G55, G56, G57

The settable zero offset values for each axis can be entered in the controller via the operator panel or via the universal interface.

The values are calculated in absolute and incremental position data blocks for the block end point if the relevant axis is programmed.

G54 to G57 permit 4 settable zero offsets each with two settings to be selected for the various axes.

The various settable zero offsets subdivide into 2 ZOs (coarse ZO and fine ZO) which are calculated additively.

The fine ZO is used as an additional fine offset (compensation) of the zero point.
For reasons of compatibility, the format \texttt{G54 X = 250 LF}, for example, can be read in, the values then being entered in the settable coarse ZO.

**Programmable zero offset G58/G59**

An additional zero offset can be programmed with G58 and G59 under the axis address for all existing axes. When calculating the path, the programmed values are added to the settable zero offset and external zero offset values.

- **Settable** zero offset (coarse and fine) 
  Input values XMW1, ZMW1
- **Programmable** zero offset 
  Input values XMW2, ZMW2
- Total effective zero offset
  \[ XMW = XMW1 + XMW2 \]
  \[ ZMW = ZMW1 + ZMW2 \]

Programming:

\[
\begin{align*}
\text{N30} & \ldots \\
\text{N35} & \text{G54 LF} \\
\text{N40} & \text{G59 X600 Z600 LF} \\
\text{N45} & \ldots
\end{align*}
\]
A block containing G58 or G59 must not include any functions other than the zero offsets. The G58/G59 functions can apply to a maximum of five axes within one block.

**Application example with G59:**

The contour has been exclusively programmed using absolute position data. In order to obtain a **finishing allowance**, the total contour can be offset in the X coordinate by means of a **programmable zero offset**.

**Zero offset with G59**

To select: `N..G59  X...LF`
To select: `N..G59  X0  LF`

The programmable zero offset values set in this program are automatically reset each time the program is terminated with M02 or M30 or aborted. After RESET all programmable zero offsets are reset.

**G53 Cancelling zero offsets**

The zero offsets which are to be suppressed by G53 are determined by the machine tool manufacturer via NC machine data.
The offset of the coordinates from the machine zero to the workpiece zero obtained by means of
• settable zero offsets (G54 to G57)
• programmable zero offsets (G58, G59)
• external zero offsets (from PLC)
can be cancelled block by block using G53.

If the relevant machine data is set, you can also use G33 to cancel:
• DRF offset
• PRESET offset.
In this case G53 has the same effect as @706 (see Section 11.9).

The tool offset must be cancelled separately.
In the block following G53 all zero offsets will be active again.

Reference to machine zero:

\[
\begin{align*}
\text{Path calculation} & \\
\text{The path calculation determines the distance to be travelled within a block, taking all offsets} & \\
\text{and compensations into consideration.} & \\
\text{The formula is generally as follows:} & \\
\text{Path} & = \text{setpoint} - \text{actual value} + \text{zero offset (ZO)} + \text{tool offset (TO)}. \\
\end{align*}
\]

If incremental position data input is used, the zero offset is incorporated normally in the first
block: Path = incremental position data + zero offset + tool offset

If a new zero offset and a new tool offset are programmed in a new program block, the formula
is as follows:

- With absolute position data input
  Path = absolute position data P2 - absolute position data P1 + ZOP2 - ZOP1 +TOP2 - TOP1.

- With incremental position data input
  Path = incremental position data + ZOP2 - ZOP1 +TOP2 - TOP1.
2.7 Workpiece dimensioning, input system G70/G71

The dimensions can be entered in the program in either mm or inches.

The input system can be changed by selecting the preparatory functions G70 or G71:

G70 input system: inches
G71 input system: metric (mm)

The controller converts the entered value into the input system of the initial setting. When this type of block is processed, the value will be displayed already converted in the initial setting of the system.

It is essential to ensure that the units of measurement are the same before selecting subroutines or cycles.

The unit of measurement which is different from the initial setting can be fixed for one or more blocks or for an entire program.

The first block must then contain the necessary G function; the initial setting must be written again following the last block (the initial setting is written automatically following a program end with M02 or M30).

The following are dependent on the initial setting of the input system:

- Actual value display (including distance to go)
- Zero offset
- Feedrate/cutting speed G94/G95
- Tool offset

The following are dependent on the programmed G70 or G71:

- Position data X, Z
- Interpolation parameters I, J, K
- Chamfers/radii B-/B
- Parameters related to position data, interpolation parameters and chamfers/radii.
**Example:** G71 - Initial setting (metric)

![Diagram of initial setting G71](image)

Input in inches for initial setting G71

N05   
N10 G91 250 Lf  
N15 G03 G70 X - 1 Z1 K1 Lf  
N20 G01 G71 X - 30 Lf  
N25   

Circular arc programmed in inches

Straight line programmed in metric dimensions

**2.8 Mirroring**

**Mirroring of one axis**

Mirroring of a coordinate axis permits machining of a contour

- with the same dimensions,
- at the same distance from the other axes,
- on the other side of the mirror axis and as a mirror image.

When mirroring an axis the controller inverts

- the sign of the coordinates of the mirrored axis,
- the direction of rotation in the case of circular interpolation (G02 G03, G03 G02),
- the direction of machining (G41 G42, G42 G41).

There is no mirroring of:

- tool length offsets
- zero offsets.

In the case of a milling machine it must be taken into consideration that for the facing axis the tool length offset and the position of the tool cutting point are mirrored as a function of a machine data. This does not apply when mirroring the Z axis.
If the bit of MD 572*.3 has been set, the tool offset of the facing axis is mirrored too.

Mirroring of Z axis

Mirroring is always about the coordinate axis. In order for the contours to be mirrored to the exact position where they are to be machined, the position of the program start when mirroring is called must be such that the axes of the coordinate system are located exactly between the programmed contour and the mirrored contour.

If necessary, the zero of the coordinate system can be offset to the correct position before mirroring is called in the program (W to W prime). To miror the workpiece onto the right position the zero must be offset by the value “MO”. This ensures that the distance of both workpiece from the zero is equal. After mirroring the zero can be set back to its original position.

Offset of workpiece zero
The "mirroring" function is selected via the PLC.

**Example:** Selection of mirroring

```
N10 G90 G54 G00 X0 Z0 L_F
N20 Z30 L_F
N30 G1 Z0 F500 L_F
N35 M... L_F Mirror Z axis
N36 G04 F... L_F Dwell possibly on account of PLC cycle time
N37 L999 P1 L_F Empty buffer
`

Special function @ 714 (buffer empty) makes it possible to stop an additional block increment calculation until the buffer is empty.

The use of @ 714 or L999 is required for all displacements requiring external influencing, e. g. mirroring.

### 2.9 Programmable working area limitation G25/G26

Programmable working area limitation provides machine protection in the event of programming and operating errors.

The slide reference point F must only move in the limited range (light area). As soon as the tool leaves this limited area or is located outside this area on program start, or as soon as a position outside the working area limitation is programmed, the path setting is terminated or a travel command is not accepted (program stop, no program start, alarm).

The current following error is eliminated. Programmable working area limitation is active in automatic mode with the values in the setting data.

Programmable working area limitation is called using G25 and G26:

- **G25** minimum working area limitation
- **G26** maximum working area limitation
Example:

N10 G25 X-30 Z200 L F
N20 G26 X200 Z500 L F

No more data are allowed in this block. With G25/G26 the values in the setting data are overwritten. Working area limitation is no longer active when -99999.999 and +99999.999 respectively are input for the minimum and maximum values per axis in the setting data.

Example: Turning machine
2.10 Software cam

You can reduce the retooling times during a machining operation on a workpiece by dividing the machining table into two working areas. While in the first area workpiece is being machined a tool can be changed in the second. You can use the “software cam” function to divide the machining table up in this way.

The “software cam” function is an option. It can be activated in all modes except PRESET and REFPOINT and only refers to linear axes.

The “software cam” function generates cam signals and can be parameterized via R parameters and machine data. The R parameters contain the axis positions of the individual cams (cam positions) and are grouped into a cam parameter block. With five reals axes, as with SINUMERIK 810T/M, 820T/M, up to ten cams can be set up. Two cams form a cam pair. Please consult the machine manufacturer for the numbers of the R parameters for the cams.

Cam signals:

Cam signals are control signals from the NC. They emulate a cam of infinite length which is activated in direction of approach at the cam position. The cam signals are evaluated by the PLC program.

- Cam signals are only output after repositioning the axes.  
  Exception: axes for which no start disable is programmed before reference point approach.
- The cam positions relate to the machine system.

Cam pair and cam range:

A cam pair consists of a plus cam and a minus cam. The axis range of the plus cam is greater than its cam position and the axis range of the minus cam is less than its cam position. The cam positions must relate to the machine system (metric or inch). The machine related axis position can be read with @361.

No check is made to whether the cam position lies within the maximum traverse range. The axis range assigned to the cam is designated the cam range.

A cam pair can only ever be assigned to one NC axis, but Several cam pairs can be activated for one axis.
2.10 Software cam

Minus cam < plus cam

Plus cam < minus cam
Cam parameters:

All cam parameters are grouped into an R parameter block. The R parameter block is designated cam parameter block.

In five consecutive R parameter pairs (cam pairs) the values of the axis positions for the ten software cams are stored. There is one parameter for the negative cam direction and one for the positive cam direction per cam pair. A cam parameter block can be formed from global or channel-specific R parameters. The beginning of the cam parameter block (Rxxx), i.e. the number xxx of the first R parameters, is set by the machine manufacturer.

The assignment of axes to cam pairs is also performed by and can be obtained from the machine manufacturer.

Please see the machine manufacturers instructions for how to activate the software cams.

<table>
<thead>
<tr>
<th>Cam pair 1</th>
<th>Cam pair 2</th>
<th>Cam pair 3</th>
<th>Cam pair 4</th>
<th>Cam pair 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R xxx</td>
<td>R xxx+1</td>
<td>R xxx+2</td>
<td>R xxx+3</td>
<td>R xxx+4</td>
</tr>
<tr>
<td>R xxx+5</td>
<td>R xxx+6</td>
<td>R xxx+7</td>
<td>R xxx+8</td>
<td>R xxx+9</td>
</tr>
</tbody>
</table>

Definition of cam parameters

2.11 Coordinate rotation

By coordinate rotation the coordinate system of the workpiece can be adapted to the coordinate system of the machine. Execution of a part program (machining) takes place in the rotated coordinate system.

The centre of rotation (DZ) is the point defined by the sum of the zero offset (ZO).
The following coordinate rotations can be activated:

- Settable coordinate rotation G54 to G57
- Programmable coordinate rotation G58 A... and G59 A...

The sum of "settable coordinate rotation" and "programmable coordinate rotation" gives the "effective coordinate rotation".

**Settable coordinate rotation:**

- There is no distinction between coarse and fine setting
- The adjustable coordinate rotation is defined using setting data
- Input of the value for angle of rotation "A" via keyboard into the input display coordinate rotation.
- Via the universal interface the angle of rotation "A" can be input for G54...G57 or for G154...G157 (input for G254...G257 is not permissible)

By G54 the Z0 entered in the setting data for G54 and the coordinate rotation G54A... entered in the setting data are activated.

**Example:** Settable coordinate rotation

```
N...
N... G90 G54 LF
N...
```

*While coordinate rotation is active, the axes concerned may only be programmed with path feed.*
Programmable coordinate rotation:

- Coordinate rotation can be programmed with G90 (absolute dimensions) or G91 (incremental dimensions)
- The angle of rotation is programmed under address "A..."

Example: Programmable coordinate rotation

```
N...  G90 G58 X20 Z50 A45 LF
N....
```

![Diagram of coordinate rotation](image)

Program control

- G53 bzw. @706: Coordinate rotation and zero offset are suppressed block-by-block
- M02, M30: Programmable coordinate rotation and zero offset are deleted, settable coordinate rotation and zero offset are retained (setting data).

Further characteristics:

- Coordinate rotation can run in "AUTOMATIC" mode
- Coordinate rotation is channel-specific; for "SIMULATION" the coordinate rotations entered for channel 3 are effective.
- Coordinate rotation is not taken into account in "ACTUAL VALUE DISPLAY"
- Circular interpolation must not be programmed immediately after coordinate rotation
- If cutter radius compensation is selected:
  - The turning centre (sum of zero offsets) must not be modified
  - Angle of rotation "A" of settable and programmable coordinate rotation can be modified.
- Coordinate rotation is associated with the function "Empty buffer until coordinate rotation" (@715). This function is initiated internally by block preparation and need not be programmed.

You must set or program changes in the zero offset and plane selection before selecting coordinate rotation.
2.12 Scale modification: Selection G51, cancellation G50

This function can considerably reduce the programming effort for geometrically similar parts.

- Scale modification is effective when G52 is programmed.
- G51 is modal and is only cancelled when G50 is programmed (scale modification deselection).
- Scale modification is based on reference point PB (scale centre). The coordinates (Z,X) of PB are specified on programming. If Z and X are not specified, the control generates the value Z = 0 and X = 0 (reference point = workpiece zero).
- The value for the scale factor is specified under address "P..." in the range 0.0001 to 99.9999.
- With scale modification the following values are converted
  - Axis coordinates
  - Interpolation parameters
  - Radius
  - Programmable zero offset
  - Thread lead, increase and decrease.
- Scale modification is connected with the function "Empty buffer" @ 714. This function is initiated internally.

Example: Possible notation in programming

a) N...
   N... G51 Z70 X30 P1.5 L
   N...
   OR

b) N...
   N... G51 P1.5 L
   N... G51 Z70 X30 L
   N...

Scale factor (channel-specific) and reference point are displayed on the screen under "SETTING DATA".
The scale modification is enabled for the respective axis via setting data 560*, bit 2.
3 Programming of Motion Blocks

3.1 Axis commands

The addresses of the axis command determines the axis in which the following numeric value must be traversed, e.g. X, Z.

Addresses A, B, C, U, V, W and Q are optionally available for further axes. For these extended address notation is possible (e.g. C1 = ...).

The address letters for the axes, interpolation parameters, radius and chamfer are freely selectable.

Use is made below of the address letters used in preference for turning machines.

Rotary axis:

All axes can also be declared rotary axes.

Defined via machine data. The designation preferably used for the rotary axis is “C”.

- The traversing range with absolute position data programming (G90) is ± 360.000 degrees.
  
The sign specifies the direction in which the axis is positioned at the absolute position within a revolution.

  The traversing range by the shortest path (G68) with absolute position data programming is 0 to+360.000 degrees.

  The control finds the shortest path from the current position to the programmed position so that the direction of travel is determined automatically.

  The first time the rotary axis is programmed with “G90” in the part program “G68” is automatically activated for this block. This automatic generation can be deselected with “G91 C0 Lf”.

  The above notes on G68 are also valid when a “block search with calculation” is applied to a part program for the first time.

With the setting modulo 360.000 (machine data), the actual value at 360.000 degrees is set to zero.
With incremental position data programming (G91) the traversing range of +99999.999 must be observed. The sign only specifies the direction of travel.

It is possible to define several axes as rotary axes simultaneously. The rotary axes can generally be rotated infinitely.

**Program section:**

```plaintext
N5 G91 C99999.999 LF
```

**Auxiliary axes:**

The term “auxiliary axis” designates an axis used for workpiece or tool handling (loader, turret, magazine etc.) as opposed to actual workpiece machining.

Auxiliary axes may be located in the machining channel or in a separate NC channel. They have the same range of functions as an NC main axis (linear interpolation, circular interpolation, contour definition, tool offset, etc.).

Providing the auxiliary axis is not programmed in the part program as the “NC axis”, synchronization with the NC main axes will also be possible via the interface controller. The position measurements, traversing range, input resolution and geometrical and position control resolution correspond to those of an NC axis.

The axis addresses are freely selectable from the available addresses. They can be further distinguished using the default address Q and an extended address notation, e.g. Q1, Q2...

If the auxiliary axis is located in a separate channel, it is not possible to program contour definition or tool offset. The NC axes and the auxiliary axes are then synchronized by the interface controller (application program).
3.1.1 Axis motion without machining G00

- Rapid traverse motions are programmed by means of the position data G00 and a target position specification. The target position can be reached by means of either an absolute position data input (G90) or an incremental position data input (G91).

- The path programmed using G00 is traversed at the maximum possible speed (rapid traverse) along a straight line without machining the workpiece (linear interpolation).

The maximum permissible axis speed is monitored by the controller. This speed is defined as machine data for each axis.

If the rapid traverse is executed simultaneously in several axes, the traversing speed is determined by the lowest axis speed specified in the MACHINE DATA. The preparatory function G00 automatically initiates "coarse exact positioning".

When G00 is programmed, the feedrate programmed under address F remains stored; it can be made active again, for example, by means of G01.

Example:

\[ \text{N5 G00 G90 Z30 X20 L}_p \] Rapid traverse, absolute position specified

![Diagram showing axis motion without machining G00](image-url)
3.1.2 Axis duplication

The function “axis duplication” is an ordering data extension.

The function “axis duplication” duplicates the programmable main axes without exceeding the maximum number of axes (on the 810 T/M, 820 T/M: five axes).
In this way it is possible to machine two identical workpieces on a machine tool with two tool systems with one part program running on one channel of the NC.
The function “axis duplication” is only available in the “AUTOMATIC”, “MDI AUTOMATIC” and “AUTOMATIC interrupted” modes.
With axis duplication there is a measuring circuit for each axis, for the duplicated axes as well.
For this reason, no more than two axes can be duplicated. The fifth axis is the available as an independent axis.
Please consult the machine manufacturer for the axis designations with the “axis duplication” function.

3.1.2.1 Function

With the function “axis duplication” the NC executes one part program in two separate tool systems simultaneously. In the part program only one tool system is programmed (tool system 1).
For example, if on a turning machine the leading (programmed) axes are the X1 axis and the Z axis in tool system 1, the following (duplicated) axes are the X2 and Z2 axis in tool system 2.
The axes of tool system 2 must on no account be programmed in the part program. The names and traverse paths for the axes in tool system 2 are derived from the program for tool system 1. The same goes for the plane selection and the geometry assignment of the tools. Please see the machine manufacturers instructions for how to select tool system with “axis duplication”.
After NC start the function selected via the PLC (tool system 1 or tool system 2 or both active) is activated and stored. The selected function is modal until the next end of program or RESET and cannot be changed during program execution (causes alarm).
Part program with or without axis duplication:

<table>
<thead>
<tr>
<th>Tool system 1 active, 5 axes defined</th>
<th>up to 3 axes can be traversed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool systems 1 and 2 active, 5 axes defined</td>
<td>up to 5 axes can be traversed</td>
</tr>
</tbody>
</table>

Part program

%1234  
G0 X1=0 Z0 LF  
G01 X1=100 F1000 LF  
Z20 C80

What happens when tool system 1 is active:

**Part program**

| N0010 G0 X1=0 Z D5 L_F  
N0020 G01 X1=100 F2000 L_F  
N0030 Z30 C50 L_F  
N0040 G33 X1=50 I2 L_F  
N0050... |
|--------------------------------------------------|

**Result**

Axes X1 and Z move to 0 at rapid traverse rate, the selected tool is D5.
Axis X1 moves to position 100 mm at feedrate 2000 mm/min.
Axis Z and C (rotary axis) interpolate to position 30 mm and 50 degrees respectively at feedrate 2000 mm/min.
Thread cutting with axis X1.

What happens when tool system 1 and 2 are active (axis duplication):

**Part program**

| N0010 G0 X1=0 Z D5 L_F  
N0020 G01 X1=100 F2000 L_F  
N0030 Z30 C50 L_F  
N0040 G33 X1=50 I2 L_F  
N0050... |
|--------------------------------------------------|

**Result**

Axis X1/X2 and Z/Z2 move to 0 at rapid traverse rate, the selected tool is D5 in tool system 2.
Axis X1/X2 move to position 100 mm at feedrate 2000 mm/min.
Axis Z/Z2 and C (rotary axis) interpolate to position 30 mm and 50 degrees respectively at feedrate 2000 mm/min.
Thread cutting with axis X1/X2.
Characteristics of the “axis duplication” function:

- The programmed set velocity applies to both tool systems.
- Handwheels and software limit switches are axial.
- The working area limitations G25/G26 are duplicated so that they apply in tool systems 1 and 2.
- The programmable zero offsets G58, G59 also apply to the duplicated axes.
  In the “programmed zero offsets” display the value of the zero offset is always shown for both axes, the programmed and the duplicated axis.
- With the functions “second reference point” and “mirror zero offset” complete machining of a workpiece is also possible (see next page).
- A scale factor in tool system 1 is also valid in tool system 2.

While the function “axis duplication” is active, differing axial PLC signals (mirror, feed enable, follow-up, axis disable, park) cause alarm 2192 “axis duplication active”.

Tool management, tool offset:

When “axis duplication” is being used, the tool memory is divided into two. Tool numbers D1 to D49 apply to tool system 1 and tool numbers D51 to D99 apply to tool system 2. The offset between tool system 1 and tool system 2 is thus 50.

Example:

D3 is programmed for tool system 1. The control automatically selects D53 for tool system 2 when the command “D3” in the program is processed.

The tool radius must be identical for both tools but the tool length corrections can be different. When the length correction is travelled, the axis is always positioned with “exact stop”.

Note:

While axis duplication is active and MD 5011 bit 3 = 1 or bit 4 = 1, length P2 and length correction P5 are calculated as diameters for P1 = 0.

1) The tool radius of tool system 1 applies both to tool system 1 and tool system 2.
@ commands:

With the exception of @440 (programmed axis position) all axis-related @ commands function as previously. With @720 (in-process measurement) the programmed path is duplicated but only one actual value is measured. Both axes stop when the probe is triggered.

It is essential to ensure that at axis duplication a collision is not possible for measuring.

Setting data (SD):

The option bits for the scale factor applying to tool system 1 also apply to tool system 2. As for the zero offsets, the notes under “characteristics” apply.

3.1.2.2 Complete machining

For complete machining of a workpiece there are machine data not only for mirroring the zero offsets but also for a “second fictitious reference point value”. The difference between the “reference point value” and the “second fictitious reference point value” is set by the PLC as an external zero offset. The part of the NC program which follows must then refer to this reference point.

![Diagram of machining process with coordinate positions and reference points]
3.2 Axis motions with machining

The controller implements linear interpolation or circular interpolation, depending on the type of axis motion:

- **Linear interpolation:**
  - Linear motion
  - Paraxial
  - In two axes
  - In three axes

- **Circular interpolation:**
  - Circular motion in 2 axes (plane)

3.2.1 Linear interpolation G01

The tool must travel at a set feedrate along a straight line to the target position whilst machining the workpiece at the same time. The controller calculates the tool path by means of linear interpolation.

**Linear interpolation** effects motion

- In one axis direction (linear axis or rotary axis),
- From the starting position to the target position programmed using absolute or incremental position data,
- At the programmed feedrate,
- At the programmed spindle speed.

Paraxial motions and movements at any angle can be performed.

---

**Absolute position data input:**

```
N5 . . .
N10 G01 G90 Z-30 F10 Lf
N15 X5 Z-15 Lf
```

**Incremental position data input:**

```
N5 . . .
N10 G01 G91 Z-30 F10 Lf
N15 X30 Z-45 Lf
```
3.2.2 Circular interpolation G02/G03

The tool must traverse between two points on the contour in a circular arc, whilst simultaneously machining the workpiece.

The controller calculates the tool path by means of circular interpolation.

Circular interpolation effects tool motion

- along a circular arc in a clockwise direction with G02, in a counterclockwise direction with G03,
- about the programmed centre point of the circle,
- from the starting position on a circular path to the programmed end position.

The action of preparatory functions *G02* and *G03* is modal.

The interpolation parameters determine the circle or circular arc together with the axis commands:

- The starting position “KA” on the circle or circular arc is defined by the preceding block
- The end position “KE” is defined by axis values X and Z
- The centre point of the circle “KM” is defined in one of the following ways:
  - either by the interpolation parameters,
  - or directly using the radius (B).
3.2.2.1 Interpolation parameters I, K

The interpolation parameters are the paraxial coordinates of the distance vector from the starting position to the centre point of the circle.

According to DIN 66025 interpolation parameters I and K are allocated to axes X and Z. Depending on whether X or Z are programmed as absolute position data or incremental position data, the interpolation parameters must always be entered as incremental position data or absolute position data (selectable via MD) and in the correct sequence. Selection is made by the machine manufacturer via machine data.

The sign is based on the direction of the coordinates from the starting position to the centre point of the circle.

No programming is necessary
• if the value of an interpolation parameter is zero
• if the end position coordinates are the same as for the starting point of the circular path

At least one axis must be programmed for a full circle (X0 or Z0).

Circular interpolation with interpolation parameters

**Circle end position monitoring:**
Prior to processing a circle block, the NC checks to ensure that the programmed values are correct by determining the difference in radii for starting position A and end position E.

Definition of tolerance range of circle end point monitoring
If, due to inaccurate input of radius or interpolation parameters, the difference (F) exceeds the tolerance range (T) (machine data), the circle block is not processed. An alarm “circle end position error” is issued.

The tolerance range is defined by means of machine data.

If the difference in the radii is within the tolerance range, the centre point parameters are corrected since it is assumed that the circle end position has been “properly” programmed.

The circle block is then executed with the new, compensated centre point.

3.2.2.2 Radius programming

In many cases the dimensions of a drawing are such that it is easier to specify radius B when defining the circular path. Extended address notation for the radius is not possible. The end point and radius B are programmed in radius programming. G02 or G03 determines the direction of movement about the circle defined by means of the circle end position and the interpolation parameters or by means of radius B. Because the radius specified gives an unambiguous circular path only within a semicircle, it is necessary to specify whether the angle is less than or greater than 180 degrees.

Thus the radius is given the following sign:

+ B: Angle less than or equal to 180°
– B: Angle greater than 180°.
It is not permissible to program the radius if the traversing angle is $0^\circ$ or $360^\circ$. Full circles must therefore be programmed using interpolation parameters.

**Example: Radius programming**

```
N5  G03 G90 X15 Z-15 B15 F500 Lf  Tool machining from point 1 to point 2
N10 G02 G90 X30 Z-30 B15 Lf  Tool machining from point 2 to point 1
```

Action of G02 and G03 before and after turning centre

*In front of turning centre*

*Behind turning centre*
Programming example: Turning machine

Interpolation parameters:

N5  G03 G90 X40 Z80 X15 I0 F500 L_F
N10 G02 X70 Z65 K0 I15 L_F

Tool traverse from P1 to P2
Tool traverse from P2 to P1

or radius programming:

N5  G03 G90 X40 Z80 B+15 F500 L_F
N10 G02 X70 Z65 B+15 L_F

Tool traverse from P1 to P2
Tool traverse from P2 to P1
3.2.3 Cylindrical interpolation

Cylindrical interpolation permits machining of cylindrical paths with one rotary axis and one linear axis at a constant rotary table diameter. Both linear and circular contours may be programmed, but circles can only be programmed with circle radius programming (see Section 3.2.2.2). It is not possible to input the interpolation parameters I, J and K. SPLINE interpolation programmed, but circles can only be programmed with circle radius programming (see Section 3.2.11) is not possible either. The position of the rotary axis is entered in degrees. It is converted to the circumferential dimensions of the working diameter internally in the control. The ratio is programmed under G92 P for this purpose.

The control forms the ratio from the machining diameter and the unit diameter as follows:

\[ P = \frac{\text{machining diameter}}{\text{unit diameter}} \]

**Input system:**

Unit diameter

- 114.592 mm in the metrical system with input resolution 10\(^{-3}\) mm
- 114.592 inches with input resolution 10\(^{-3}\) mm.

The unit diameter is derived from the relation \( \pi \cdot d = 360 \):

(\( \cdot \) = Multiplication character).

\[ \text{Unit diameter} = \frac{360}{\text{in mm or inches}} \]

No characters other than the axis name must be written in a block containing G92P...

N..   G92  P..  C  LF
P..  Factor for unit circle, \( P = \text{working diameter/unit diameter} \)
C  Rotary axis.

The input resolution for P is 10\(^{-5}\) and the value of P is < 100. The action of the factor for the unit circle is modal until reprogrammed or reset with M02/M30 or RESET. The programmed feedrate is adhered to on the contour.

As long as the machining diameter is not equal to the unit diameter, this axis (e.g. C) can only be interpolated with one further axis, i.e. for interpolation with more than 2 axes, the machining diameter must be equal to the unit diameter (factor \( P = 1 \)). For \( P < 1 \), it is important to note that the interpolation resolution of the rotary axis is \( 1/p \) greater with cylindrical interpolation than without it.
While cylindrical interpolation is selected, the rotary axis must not be programmed with G68 (only G90, G91).

You can use cylindrical interpolation together with CRC/TNRC. It is possible to select and to deselect cylindrical interpolation while CRC/TNRC is selected. The function “soft approach/exit” cannot be used together with cylindrical interpolation.

**Example:** Cylindrical interpolation

![Cylindrical interpolation diagram]

Cylindrical interpolation

```plaintext
%10 LF
N10 G92 P3 C LF
N15 G01 C40 Z... F1000 S800 M3 LF
N20 G03 C60 Z216 B+60 LF
N25 G01 C100 Z456 LF
N30 G02 C150 Z549 B+165 LF
N35 G01 C260 LF

; Cancellation of cylindrical interpolation
N60 G92 P1 C LF
N70 M02 LF
```

Selection of cylindrical interpolation

Radius R60 (B60)

Radius R165 (B165)
3.2.4 Feedrate F/G94/G95/G96/G97/G98

The feedrate F is programmed in mm/min or mm/rev:

- **G94 F..**: Feedrate in mm/min
- **G95 F..**: Feedrate in mm/rev
- **G96 F..**: Feedrate in mm/rev
- **G96 F.. S..**: Constant cutting speed S (in m/min) at preset feedrate F (mm/rev) (applies for the leading spindle only).
- **G98 F..**: Feedrate in rev/min (for rotary axes only)

The feedrate determines the machining speed (tool path feedrate) and is adhered to with all types of interpolation, even when tool offsets on the contour are taken into consideration. The value programmed under address F remains in the program until a new F value is programmed. The F value is deleted at the end of the program or on a reset. An F value must therefore be programmed in the first program block.

If the G functions G94, G95 or G96 are changed, the new F value must be programmed in the next block at the latest.

The programmed feedrate F can be modified between 1% and 120% by means of a feedrate override switch at the operator panel. The 100% setting corresponds to the programmed value.

For every G98 active traversing block exactly one rotary axis with a traversing path > 0 must be programmed.

![](image)

**The maximum values are defined via the machine data.**

### Constant cutting speed G96 S..

A constant cutting speed can be entered under address S with G96. A feedrate F can also be programmed at the same time.

- **G96 F.. S..**: Constant cutting speed S (in m/min) at preset feedrate F (mm/rev).

**Example**: Turning machine

The **controller determines** the **spindle speed** for the current turning diameter in accordance with the programmed cutting speed.

```
N5.. G96 F.. S..
```

The relationship between the turning diameter, the spindle speed and the feed motion permits optimum matching of the program to the machine, the material being machined and the tool.

The zero of the X axis must be the turning centre. This is ensured by the reference point approach.
When calculating the spindle speed for the constant cutting speed, the following values are taken into consideration:

- Actual value
- Tool length compensation
- Zero offset in the X direction
  - Settable zero offset G54, G55, G56, G57
  - Programmable zero offset G59, G58
  - External supplementary compensation (PRESET, DRF).

The workpiece must not be displaced from the turning centre as a result of zero offset in the X axis. Zero offset in the X axis may be used to move the tool carrier. Reference is always made to the workpiece zero when determining the spindle speed.

One gear step only is used for a constant cutting speed. It is not permissible to change the gear step. The relevant gear step must be selected in advance.

**Spindle speed limitation G92**

If you are working with a constant cutting rate (G96), the spindle speed S (rev/min) can get too great for small radial values. You can avoid this by limiting the spindle speed S with the command G92 S... . The spindle speed limitation is only active if you have first selected G96. The function G92 is modal and must be in a block of its own.

**Example:**
G92 S500 means that the spindle speed remains limited to 500 rev/min at a constant cutting rate (G96 S...).
If higher values are programmed for the spindle speed in a following block, the spindle only runs at the limit speed regardless. You can also define a spindle speed limitation G92 directly in the “SETTING DATA SPINDLE” screen form.

**Constant speed G97**

The constant cutting speed is cancelled with G97. The final speed is retained as the constant speed.
An undesirable speed change can be avoided with G97 for motions in the X direction without machining.

**G97** Cancellation of constant cutting speed and storage of final set speed of G96

**Feedrate reduction ratio M37**

The programmed feedrate can be reduced by 1:100 with M37. M37 can be cancelled with M36 (reset)
3.2.5 Thread cutting G33/G34/G35

There are various types of thread as follows:

- Threads with a constant lead
- Threads with a variable lead
- Single or multiple threads
- Threads on cylindrical or tapered blocks
- External or internal threads
- Transversal threads

The following preparatory functions are available for machining threads:

- **G33** Thread cutting with constant lead
- **G34** Thread cutting with linear lead increase
- **G35** Thread cutting with linear lead decrease

The G functions G33, G34 and G35 are assigned to the leading spindle.

The thread length is entered under the corresponding path address; start-stop and overrun sections at which the feedrate is increased or reduced must be taken into consideration. The values can be entered using absolute or incremental position data.

The thread lead is entered under addresses I, K.

The lead is entered under K for longitudinal threads, under I for transversal threads and under I and K for tapered threads. I and K must always be entered using incremental position data and without a sign.

The standard input resolution of the thread lead is 0.001 mm/revolution.

The thread lead can be programmed between 0.001 mm and 400.000 or 2000.000 mm. If a thread lead of 1 mm is programmed as the input resolution, it is possible to obtain a resolution of 0.01 mm/revolution with M37.

Right and left-hand threads are programmed by specifying the spindle direction of rotation functions M03 and M04.

The spindle direction of rotation and the speed must be programmed in the block prior to the actual thread cutting operation to permit the spindle to run up to its nominal speed.

**Example:**

```
N10 S500 M03 L F
```

Spindle speed S=500 rev/min, clockwise

```
N15 G33 Z... K... L F
```

Thread cutting with constant lead

The feed start does not begin until the zero mark is reached on the pulse encoder in order to permit threads to be cut in several steps. This ensures that the tool always enters the workpiece at the same point on the circumference of the workpiece. The cuts should be implemented at the same spindle speed to prevent discrepancies in the following error.

The feedrate override switch, the FEED OFF key, the spindle speed override switch and the SINGLE BLOCK key have no effect during thread cutting.

The feedrate programmed under F remains stored however, and is effective again when, for example, G01 is next programmed.

The lead of the tapers at which the thread is cut can be altered in steps. This ensures that longitudinal threads run out gently.
3.2.5.1 Thread with constant lead

The feedrate \( F \) is not programmed here since the feedrate is linked directly to the spindle speed via a pulse encoder.

**Example:** Thread on a cylindrical block (longitudinal thread)

lead \( h = 2 \text{ mm} \)

thread depth \( t = 1.3 \text{ mm} \)

radial infeed direction

absolute position data input:

\[
\begin{align*}
  \%387 & \text{ (G33, G90 THREAD CUTTING, LONGITUDINAL) } L_F \\
  N5 & \ G90 \ S400 \ M3 \ L_F \quad \text{Spindle speed} \\
  N10 & \ G00 \ X46 \ Z78 \ L_F \quad \text{(P1) Rapid traverse to P1} \\
  N15 & \ X38.7 \ L_F \quad \text{(P2) Infeed to 1st cutting depth} \\
  N20 & \ G33 \ Z22 \ K2 \ L_F \quad \text{(P3) Thread cutting, 1st run from P2 to P3} \\
  N25 & \ G00 \ X46 \ L_F \quad \text{(P4) Exit to P4 in rapid traverse} \\
  N30 & \ Z78 \ L_F \quad \text{(P1) Rapid traverse to P1} \\
  N35 & \ X37.4 \ L_F \quad \text{(P5) Infeed to 2nd cutting depth} \\
  N40 & \ G33 \ Z22 \ K2 \ L_F \quad \text{(P6) Thread cutting, 2nd run from P5 to P6} \\
  N45 & \ G00 \ X46 \ L_F \quad \text{(P4) Exit to P4 in rapid traverse} \\
  N50 & \ M30 \ L_F
\end{align*}
\]

incremental position data input:

\[
\begin{align*}
  \%388 & \text{ (G33, G90 THREAD CUTTING, LONGITUDINAL) } L_F \\
  N5 & \ G91 \ S400 \ M3 \ L_F \quad \text{Spindle speed} \\
  N10 & \ G00 \ G90 \ X46 \ Z78 \ L_F \quad \text{(P1) Rapid traverse to P1} \\
  N15 & \ X-3.65 \ L_F \quad \text{(P2) Infeed to 1st cutting depth} \\
  N20 & \ G33 \ Z-56 \ K2 \ L_F \quad \text{(P3) Thread cutting, 1st run from P2 to P3} \\
  N25 & \ G00 \ X3.65 \ L_F \quad \text{(P4) Exit to P4 in rapid traverse} \\
  N30 & \ Z56 \ L_F \quad \text{(P1) Rapid traverse to P1} \\
  N35 & \ X-4.3 \ L_F \quad \text{(P5) Infeed to 2nd cutting depth} \\
  N40 & \ G33 \ Z-56 \ K2 \ L_F \quad \text{(P6) Thread cutting, 2nd run from P5 to P6} \\
  N45 & \ G00 \ X4.3 \ L_F \quad \text{(P4) Exit to P4 in rapid traverse} \\
  N50 & \ M30 \ L_F
\end{align*}
\]
Example: Thread on a tapered block (taper thread)

Lead \( h = 5 \) mm
Thread depth \( t = 1.73 \) mm; \( \alpha = 15^\circ \)
Radial infeed direction
Both end position coordinates must be written.
The lead \( h \) is entered under \( K \).

Calculation of thread start and end position coordinates

\((A, B, C \text{ etc. are diameters})\)

1st cut \( P2 \ldots P3, t = 1 \) mm
2nd cut \( P5 \ldots P6, t = 1.73 \) mm

Calculation of points \( P2 \) and \( P3 \)

\[
X (P2) = C + 2 \text{ mm} = 65.86 \text{ mm} \\
X (P3) = D + 2 \text{ mm} = 103.366 \text{ mm}
\]

Absolute position data input:

```gcode
%389 (G33 THREAD CUTTING, TAPER) L_F
N30 G90 S200 M3 L_F
N35 G00 X110 Z170 L_F (P1) Rapid traverse to P1
N40 X65.86 L_F (P2) Infeed to 1st cutting depth
N45 G33 X103.366 Z100 K5 I1.34 L_F (P3) Thread cutting, 1st run from P2 to P3
N50 G00 X110 L_F (P4) Exit to P4 in rapid traverse
N55 Z170 L_F (P1) Exit to P1 in rapid traverse
N60 X63.86 L_F (P5) Infeed to 2nd cutting depth
N65 G33 X101.366 Z100 K5 I1.34 L_F (P6) Thread cutting, 2nd run from P5 to P6
N70 G00 X110 L_F (P4) Exit to P4 in rapid traverse
N75 M30 L_F
```

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
If $\theta > 45^\circ$, the thread lead in X direction (with the parameter I) must be programmed instead of the thread lead in Z direction.

**Example:** Transversal thread

**Diagram:**

- Lead $h = 2$ mm
- Thread depth $t = 1.3$ mm
- Infeed direction perpendicular to cutting direction

**Absolute position data input:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N40</td>
<td>G90</td>
<td>S100 M3 L_F</td>
</tr>
<tr>
<td>N45</td>
<td>G00</td>
<td>X4 Z82 L_F</td>
</tr>
<tr>
<td>N50</td>
<td>G00</td>
<td>X82.7 Z78.7</td>
</tr>
<tr>
<td>N55</td>
<td>G33</td>
<td>X36 I2 L_F</td>
</tr>
<tr>
<td>N60</td>
<td>G00</td>
<td>Z82 L_F</td>
</tr>
<tr>
<td>N65</td>
<td>X4</td>
<td>L_F</td>
</tr>
<tr>
<td>N70</td>
<td>G90</td>
<td>S100 M3 L_F</td>
</tr>
<tr>
<td>N75</td>
<td>G33</td>
<td>X36 I2 L_F</td>
</tr>
<tr>
<td>N80</td>
<td>G00</td>
<td>Z82 L_F</td>
</tr>
<tr>
<td>M85</td>
<td></td>
<td>M30 L_F</td>
</tr>
</tbody>
</table>

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
3.2.5.2 Thread with variable lead

The thread lead per turn is altered by the value programmed under address F until the maximum or minimum possible value is obtained.

**G34 Lead increase:**

\[
\text{N25 G34 G90 Z217 K2 F0.1 LF}
\]

K2 Initial lead 2 mm  
F0.1 Lead change + 0.1 mm per turn, 
i.e. after 5 turns the lead is 2.5 mm

**G35 Lead decrease:**

\[
\text{N45 G35 G90 Z417 K10 F0.5 LF}
\]

K10 Initial lead 10 mm 
F0.5 Lead change 0.5 mm per turn, i.e. after 10 turns the lead is 5 mm.

Value F is calculated from the initial and final leads:

\[
F = \frac{\text{Initial lead}^2 - \text{Final lead}^2}{2 \times \text{Thread length}}
\]

The value should be used without a sign.
3.2.5.3 Infeed options

The tool may be advanced perpendicular to the cutting direction or along the flank.

Infeed perpendicular to cutting direction

Flank infeed $Z = X \cdot \tan \frac{e}{2}$ - cutting with one side of the tool
Example: Longitudinal thread with constant lead, flank infeed

Lead \( h = 2 \text{ mm} \)

Thread depth \( t = 1.3 \text{ mm} \)

Infeed angle \( \theta = 60^\circ \)

\[
X_1 = \frac{(46-38.7)}{2} = 3.65 \\
Z_1 = X_1 \tan \frac{\theta}{2} = 2.136 \\
X_2 = 3.65 + \frac{1.3}{2} = 4.3 \\
Z_2 = X_2 \tan \frac{\theta}{2} = 2.483
\]

Absolute position data input:

%391 (G33 THREAD CUTTING) \( L_F \)  
N05 G90 S50 M3 \( L_F \)  
N10 G00 X46 Z80.483 \( L_F \)  (P1)  
N15 X38.7 Z78.347 \( L_F \)  (P2)  
N20 G33 Z22 K2 \( L_F \)  (P3)  
N25 G00 X46 \( L_F \)  (P4)  
N30 Z80.483 \( L_F \)  (P1)  
N40 X37.4 Z78 \( L_F \)  (P5)  
N45 G33 Z22 K2 \( L_F \)  (P6)  
N50 G00 X46 \( L_F \)  (P4)  
N55 M30 \( L_F \)  

Absolute dimensioning, spindle speed

Rapid traverse to P1

Infeed to 1st cutting depth

Thread cutting, 1st run

Rapid traverse to P4

Rapid traverse to P1

Infeed to 2nd cutting depth

Flank infeed

Thread cutting, 2nd run

Exit to P4 in rapid traverse
3.2.5.4 Multiple threads

Thread cutting always begins at the synchronization point of the zero mark of the pulse encoder. The feed is not enabled unless this signal is received from the digital rotary transducer. The starting position for cutting the thread can be offset with the aid of the program. This makes it possible to cut multiple threads. A turn of a multiple thread is programmed in the same manner as a single thread. When the first turn has been fully machined, the starting position is offset by \( h' \) and the next turn is machined.

\[
h' = \frac{\text{Thread lead}}{\text{No. of turns}}
\]

The various turns must be executed at the same spindle speed in order to avoid discrepancies in the following error.

**Example:** Multiple thread with constant lead

![Diagram of multiple thread](image)

Lead \( h = 6 \) mm  
Thread depth \( t = 3.9 \) mm  
Radial infeed direction  
2-turn

In this example each turn is machined in two steps. When the first turn has been fully machined, the second turn is machined by offsetting the starting position by \( h' \).

\[
h' = \frac{\text{Thread lead}}{\text{number of turns}}
\]

\[
h' = \frac{6}{2} = 3 \text{ mm}
\]
The Z coordinate is thus for the 2nd turn Z118 instead of Z115.

%392 (G33, MULTIPLE THREAD)  L_F
N05  G90  S50  M3  L_F
N10  G00  X66  Z115  L_F  (P1)
N15  X56.1  L_F  (P2)
N20  G33  Z30  K6  L_F  (P3)
N25  G00  X66  L_F  (P4)
N30  Z115  L_F  (P1)
N35  X52.2  L_F  (P5)
N40  G33  Z30  K6  L_F  (P6)
N45  G00  X66  L_F  (P4)
N50  Z118  L_F  (P7)
N55  X56.1  L_F  (P8)
N60  G33  Z30  K6  L_F  (P3)
N65  G00  X66  L_F  (P4)
N70  Z118  L_F  (P7)
N75  X52.2  L_F  (P9)
N80  G33  Z30  K6  L_F  (P6)
N85  G00  X66  L_F  (P4)
N90  M30  L_F

1st turn

2nd turn
3.2.6 Tapping without encoder G63

Preparatory function G63 is used to tap threads using a **tap in the compensating chuck**. There is no functional relationship between the spindle speed and the feedrate.

The spindle speed is programmed under address S and a suitable feedrate under address F. The length compensating chuck must allow for the tolerances between the feedrate and the speed as well as the spindle overrun when the position is reached.

With G63 the **feedrate override switch is set to 100%**. The spindle may also be stopped in conjunction with “Feed hold” depending on the design of the interface controller. The spindle speed override switch is active.

G63 can only be used in blocks with **linear interpolation G01**.

3.2.7 Tapping without G36 compensating chuck

Precondition for this function is rotary axis operation of the spindle.

Preparatory function G63 is used to tap threads (or to cut in the case of turning machines) using a tap without compensating chuck. The spindle in rotary axis operation and the infeed axis/axes interpolate. Thus a compensating chuck is not required for tapping or thread cutting on turning machines.

G36 can be used for the following types of thread:

- Cylindrical threads (spindle in rotary axis operation and one linear axis)
- Tapered thread (spindle in rotary axis operation with two linear axes)

Other programmings with G36 are not permitted.

Threads with lead increase and lead decrease (as in the case of G34, G35) cannot be implemented with G36.

On writing G36, the feed for the rotary axis in rev/min (G98) is automatically activated. No other feedrate for the rotary axis may be selected in a program part where G36 is active.
G36 is not deactivated before another G function of the G group 0 (e.g. G0) is selected. On
deselecting G36, G98 is also deselected. The feed active beforehand becomes active again.

Programming conditions:

- Rotary axis operation of the spindle must be selected
- In every block where G36 is active, only the rotary axis and the relevant linear axes may
  be written.
  A numerical value behind the rotary axis is ignored. The feed value F in rev/min for the
  rotary axis must be written in the first G36 active block.
- The thread lead is entered under addresses I, J, or K (please observe the machine
  manufacturer's specifications). The sign of the values for I, J, or K indicate the sense of
  rotation of the rotary axis. Plus means right-hand thread and minus means left-hand
  thread.
- One rotary axis and one linear axis must be programmed for cylindrical thread and one
  rotary axis and two linear axes for tapered threads.
  Other axis programmings generate alarm 3006. In the case of a tapered thread the lead
  increase parameter I, J or K is assigned to the linear axis with the longer traversing path.
  This enables programming of tapered threads with an inclination angle of max. 45°C.

**Example 1**: Cylindrical thread assigning K for the infeed axis Z

```
N10 M1=70 LF
N15 G0 C30 Z2 LF
N20 G36 C Z-30 K5 F10 LF
N25 C Z2 K-5 LF
```

**Example 2**: Tapered thread assigning K for the infeed axis

```
N10 M1=70 LF
N15 G0 C30 X10 Z2 LF
N20 G36 C Z-30 X20 K5 F10 LF
```

Behavior on reset:

In the case of M02/M30 or reset of operator panel, the delete position stored in the machine
data becomes active (G36 or G98).
Display:

- G36 and G98 are displayed in the "G functions" display field
- While G98 is active, the feedrate is displayed related to the rotary axis in rev/min (identifier U).
- The "rev/min" feedrate is displayed in the feed display with identifier U.

**Behavior after block search with calculation**

When starting a G36 block following a block search with calculation, the spindle must be in rotary axis operation. Otherwise, the axis-specific reset alarm 196* "Follow-up/Park for axis" is output. If no error occurs, the programmed G98 value (rev/min) is used as feed value.

From a technological point of view, a positioning block in front of the tapping block should be selected or all accumulated paths and offsets should be traversed in "REPOS" mode.

**Note:**

G36 enforces block end velocity 0. Thus a dwell time is no longer required from technological point of view.

**3.2.8 Exact positioning G09/G60/G00, continuous path operation G62/G64**

**3.2.8.1 Fine and coarse exact stop tolerance ranges G09/G60/G00**

<table>
<thead>
<tr>
<th>G09/G60</th>
<th>Fine exact stop tolerance range</th>
</tr>
</thead>
<tbody>
<tr>
<td>G00</td>
<td>Coarse exact stop tolerance range</td>
</tr>
</tbody>
</table>

G09, G60 and G00 can be used to approach a target position within a specified **exact stop tolerance range**. When the **exact positioning window** is reached, the feedrate of the traversed axis (from P1 to P2, see next page) is reduced to 0. The following error is eliminated. At the same time a block change is initiated and the axis motion programmed in the next block (from P2 to P3) begins.

The action of G09 is block by block, whilst that of G60 is modal.

Functions G09 and G60 can be used, for example, for machining sharp corners, for plunge-cutting or when reversing the direction of movement.
Fine and coarse exact stop tolerance ranges

Fine exact positioning: 10 μm (window F1)
Coarse exact positioning: 250 μm (window F2).

The limit values can be defined in machine data.

If the two exact stop tolerance ranges are identical (window F1 = window F2), the effect of G00 will be the same as that of G09/G60. The rapid traverse motion generally has a larger exact positioning window. This saves time during rapid traversing (earlier block change).

The thin line shows the velocity control of the controller. The position controller in the NC ensures a smooth characteristic (thick line).
3.2.8.2 Continuous path operation G62/G64

Function G62 reduces the feedrate $F$ to a rate defined in NC MD 3 towards the end of the block. With G64 the feedrate is not reduced (initial setting for continuous-path operation). The action of G62 and G64 is modal.

**Application:**
In woodworking operations the feedrate must not be allowed to become zero during a block transition, or scorch marks will result on the workpiece.

**Block transition without feedrate reduction G64:**
Preparatory function G64 is used if relief cutting must be avoided during transitions from one block to the next. It also permits smooth transitions when changing the direction of movement.
3.2.8 Exact positioning G09/G60/G00, continuous path operation G62/G64

Continuous path operation with G64 and with a change in direction in one axis

Change in direction with (G09/G60) and without (G64) a reduction in feedrate

3.2.9 Dwell G04

Dwells are required for relief cutting, possibly for speed changes and machine switching functions (steadyrest, tailstock, etc.). The dwell is always entered without a sign.

The dwell is entered under address X or F. The time range is as follows:
- 0.001 to 99999.999 sec for X,
- 0.001 to 99.999 sec for F and
- 0.1 to 99.9 spindle revolutions.

The action of G04 is block by block. No other functions may be written in a block containing a dwell time.
3.2.9 Dwell G04

Example:

```
N10 G04 X11.5 LF
```

Dwell 11.5 sec always without a sign

A further option is to program the dwell in spindle revolutions. The dwell is then programmed under address S in the range between 0.1 and 99.9 revolutions.

Example:

```
N10 G04 S5 LF
```

Dwell 5 spindle revolutions

3.2.10 Soft approach to and retraction from contour

To avoid cutting marks approach to and retraction from a contour is tangential.

Soft approach to and retraction from a contour can be used with all tool types. Radius and wear are taken into account.

**Approach** to or **retraction** from the contour can be programmed with the following functions:

- **G147** Linear approach
- **G247** Approach in quadrant
- **G347** Approach in semi-circle
- **G148** Linear retraction
- **G248** Retraction in quadrant
- **G348** Retraction in semi-circle
- **G48** Retraction from contour in the same way as the contour was approached

Example: Straight line to circle

![Diagram](image-url)
Example: Approach to straight contour in quadrant (with programming example)

Approach to straight contour in quadrant

\[
\begin{align*}
N05 & \ G01 \ G41 \ X10 \ Z20 \ D1 \ F1000 \ \text{LF} & \text{Selection TNRC counter-clockwise (G41), selection of tool offset (D1)} \\
N10 & \ G247 \ X40 \ Z70 \ B25 \ \text{LF} & \text{Selection “Approach with quarter circle”(G247), } P_s, P_o \\
N15 & \ G01 \ Z70 \ \text{LF} & \text{Travel along the linear contour up to } P_E \\
N20 & \ M30 \ \text{LF} & \text{End of program}
\end{align*}
\]

Programming features

- The functions for approach to and retraction from the contour are effective block-by-block.
- The approach block must contain the following information:
  - The coordinates of the starting point P0 of the contour
  - The value of B (approach path without touching the contour)
- The retraction block must contain the following information:
  - The coordinates of end point PE after leaving the contour
  - The value of B (retraction path without touching the contour)
- In an approach and retraction block containing G47 or G48 no other traversing movements must be programmed.
- Auxiliary functions can be programmed in both approach and retraction blocks.
- After an approach block or before a retraction block, a block only containing auxiliary function must not be programmed.
- Note that when approach and retraction blocks are combined G40 (deselection of CRC) is generated by the control in a retraction block: G41 or G42 must be programmed before each new approach block.
- With part programs which are generated block-by-block (TEACH IN / PLAYBACK) the final part program can be completed retrospectively.

Note:

“Soft approach to and retraction from a contour” can run in “AUTOMATIC” mode. In “Single block operation” don’t forget that the control inserts further blocks in “Soft approach to and retraction from the contour”. Depending on the number of blocks inserted, the NC start key must be pressed several times.
3.2.11 Polar coordinates G10/G11/G12/G13

Drawings dimensioned with angles and radius can be entered directly into the program using polar coordinates.

The following preparatory functions are available for programming with polar coordinates:

- **G10** linear interpolation rapid traverse
- **G11** linear interpolation feed (F)
- **G12** circular interpolation, clockwise
- **G13** circular interpolation, counter-clockwise

The preparatory functions are modal.

In order to define the path the control requires values for the centre, the radius and the angle. The centre is entered in absolute dimensions for right angle coordinates (X, Z) and on first programming. Any later incremental dimension input (using G91) is always based on the last centre programmed. The definition of the centre is modal and can be deleted with M02/M30.

- The radius is programmed under address B without sign.
- The angle is entered under address A without sign (input resolution 10⁻⁵ degrees). It always refers to the first positive axis programmed of the centre coordinates (reference axis).
  - The positive direction of this axis is equivalent to an angle of zero degrees.
  - The angle given is absolute or incremental and positive.

*Mixed programming using G90 and G91 in one block is possible (settable via machine data).*

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
Example: G10, G11 turning machine

\[
\begin{align*}
&N10 \text{ G90 G10 Z200 X0 B40 A+135 LF} \quad \text{(P0)} \quad \text{Approach start position P0 in rapid traverse.} \\
&N15 \text{ G11 B56.56 LF} \quad \text{(P3)} \quad \text{Machine external contour of the workpiece at feed F...} \\
&N20 \text{ B58.2 A+149 LF} \quad \text{(P4)}
\end{align*}
\]

### 3.2.11.1 Polar coordinates G110/G111

The functions G110 and G111 are used to adopt a new **centre point** or **zero point** when programming polar coordinates.

Using the new centre point, the angles are again taken from the horizontal and the radius is calculated from the new centre point. G110 and G111 have the following meanings:

- **G110**: Adopt the setpoint reached as the new centre point
- **G111**: Centre point programming with angle and radius without axis movement (example: setting the arc center of a hole circle) (the following traversing movement must be programmed using G110)

**Example**: Polar coordinates G110

\[
\begin{align*}
&\%385 \\
&(G110 \text{ Polar coordinates}) \quad LF \\
&N5 \text{ G90 G10 Z0 X0 B0 A0 F1000 LF} \\
&N10 \text{ G11 B30 A45 LF} \\
&N15 \text{ G110 B20 A30 LF} \\
&N20 \text{ M30 LF}
\end{align*}
\]
Polar coordinates G110

The block before a block with G110 or G111 must be programmed using G10 or G11.

Functions G110/G111 are non modal. They only apply to linear interpolation. The feedrate is the last F value (G11) active or rapid traverse rate (G10) to have been programmed.

Example: Polar coordinates G111

%386
(G111 Polar coordinates) \( L_F \)
N5 G11 Z40 X40 A B F1000 \( L_F \) \( P_1 \)
N10 G111 A225 B28 \( L_F \) Set centre point without traversing
N15 G110 A90 B20 \( L_F \) \( P_2 \)
N20 A180 \( L_F \) \( P_3 \)
N25 M30 \( L_F \)

Polar coordinates G111

© Siemens AG 1990 All Rights Reserved
3.2.12 SPLINE interpolation G06

A spline is the linking of curves which have the same function value, identical gradient and identical curvature.

SPLINE interpolation reduces the amount of programming when processing complex work-piece contours and makes it possible to machine shapes that are not described by standard geometries. Spline interpolation programming is described in a separate publication "SINUMERIK System 800, Spline Interpolation, Programming Guide" (Order no. 6ZB5 410-7BA02-0BA1). G06 is modal and is cancelled by another preparatory function of G group 0.
3.2.13 Coordinate transformation TRANSMIT

The "TRANSMIT" function is an ordering data option.

Definition: TRANSMIT = Transformation Milling Into Turning.

The TRANSMIT coordinate transformation function enables the face milling of turning parts on turning machines (810/820T). The program is written in a fictitious (Cartesian) coordinate system. Machine motions are performed in a real machine coordinate system. The axes for the fictitious coordinate system are defined in machine data by the machine manufacturer. A fictitious axis can only be moved when the transformation is selected. The real axes which form the transformation grouping are moved by the NC as the fictitious axes are controlled. In addition to the five real axes you can have two fictitious axes. The fictitious axes are handled by the NC in the same way as real axes. The only exception is the position control. Only the five real axes are processed connected to the measuring circuits.

The TRANSMIT function is defined by a transformation record, which contains the following information:

- G function for TRANSMIT: G131
- the names of the fictitious axes involved in the transformation (programmed axes)
- The names of the axes which are generated and traversed by the transformation (real axes)
- The name of the infeed axis.

This transformation record is defined by machine tool manufacturer and must be known before programming.

The TRANSMIT transformation is assigned to channel 1, i.e. it only runs in this channel.
3.2.13 TRANSMIT function

The plane definition defined in the channel-specific machine data applies to the real system. For TRANSMIT, G17 is defined as the basic plane (fictitious), i.e. G17 is automatically set after transformation selection. When transformation is deselected, the control automatically returns to the plane which was active before transformation was selected.

The fictitious plane is defined in the transformation data by assignment of the fictitious axes. You can program deviations form the basic planes with G16. The axis designations for fictitious and real axes must be unambiguous, i.e. the multiple use of an axis name is not permissible.

You select the TRANSMIT function with G function G141. This function must be in a block of its own, i.e. no other motions or other functions can be programmed in the same block. Otherwise the control outputs the alarm 2043 “programming error during transformation”.

You can deselect TRANSMIT with a block containing the function G130. Fictitious axes must not be programmed in the initial setting (after G130). The real axes involved in the transformation, on the other hand, can only be programmed in the initial setting (after G130). If you do not observe this condition, the alarm 2043 will be output.

When the TRANSMIT function is selected

The real axes of the transformation grouping are disabled. The machining plane is now defined by fictitious system. Linear and circular motions are combined to form a defined path. The working area limitations (G25/G26) can only be programmed in the fictitious coordinate (Cartesian) system. Tool offsets and zero offsets also relate to the fictitious coordinate system.

If you use the TRANSMIT tools with the tool length correction L1, you must specify the axis in which the correction is to be active.

You can either:

- set it up using free plane selection with the command:
  G16<1st fictitious axis><2nd fictitious axis> <infeed axis>
  or
- specify it directly in MD 5064 (name of the real infeed axis).

In both cases the tool length correction L1 is on a real axis.

Turning machines often work with a revolutionary feedrate. If this is the case, you must cancel it with G94 after selecting TRANSMIT. Please also observe Section 3.3 (G74).
Example: plane definition

```
N05 G17             Machining plane X-Y
M10 G00 X Y        Plane X-Y
M15 G16 Q1 B       Plane selection Q1-B
M20 G01 Q1=20 B=30 Plane Q1-B
M25 G131           Selection block for TRANSMIT transformation
                    Machining plane A1F, A2F
                    Deselection block for TRANSMIT transformation
                    Machining plane Q1-B
M60 G17            Machining plane X-Y
```

Explanation:
A1F ... A2F: 1st ... 2nd fictitious axis (A1F and A2F must be replaced by meaningful axis names).

Every selection/deselection of TRANSMIT performs the function “clear buffers” (@714). This function is activated internally by block preparation and need not be programmed. You must cancel the cutter/tool nose radius compensation (CRC/TNRC) before selecting TRANSMIT (because of @714). Otherwise the alarm 2081 “CRC/TNRC not allowed” is output.
You must cancel CRC/TNRC again before deselecting TRANSMIT because when TRANSMIT is deselected, the machining plane is changed. After deselection TRANSMIT the control returns to the plane which was active before TRANSMIT was selected. Only after selection or deselection of TRANSMIT can you reselect the CRC/TNRC.
TRANSMIT cannot be selected or deselected within a series of contour blocks. Block search with calculation to a part of program in which TRANSMIT is active is permitted (see Section 3.2.13.2).
3.2.13.2 Block search with calculation and the TRANSMIT function

The “block search with calculation” function can be used without restriction when TRANSMIT is active. The control generates a linear block from the current position to the target position after “block search with calculation” and “NC start” (selected block).

If the target position is behind the turning centre from the point of view of the tool, there is a risk of collision between the tool and the workpiece when the selected block is executed. In this case the rotary axis must be traversed in the “REPOS” mode until the target position is in front of the turning centre. If the tool is in the turning centre, the alarm 2191 “transformation in zero” is output when TRANSMIT is selected. Then you must move the tool out of the turning centre in JOG mode.

Manual approach to the target position
3.2.13.3 Principle of the TRANSMIT coordinate transformation

Two types of coordinate system are used in NC controls:

a) **Fictitious coordinate system:**
   Cartesian coordinate system for producing e.g. part programs for milling on turning machines.
   (Workpiece oriented coordinate system, control coordinate system, interpolation coordinate system).

b) **Real coordinate system:**
   Coordinate system based on the real arrangement of machine axes (machine coordinate system).

A coordinate transformation describes the mathematical relation between two different coordinate systems by system of transformation equations. This system of equations must be valid for both directions of transformation. The coordinates are entered and the interpolation is performed in the Cartesian (fictitious) coordinate system. The setpoints are output to the position controller in the machine (real) coordinate system. The transformation converts between these two coordinate systems.

Using the TRANSMIT function you can perform milling operations on turning machines. Motions which would be difficult to program in the real coordinate system can be implemented with relatively simple motion blocks in the fictitious coordinate system.

**Initialization of the coordinate systems**

The fictitious and real coordinate systems have the same zero.
Selection/deselection of TRANSMIT in “AUTOMATIC” or “MDI AUTOMATIC” mode:

**AUTOMATIC mode:**

<table>
<thead>
<tr>
<th>Selection of TRANSMIT (G131)</th>
<th>Deselection of TRANSMIT (G130)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = X_1$</td>
<td>$X = 0$</td>
</tr>
<tr>
<td>$Y = 0$</td>
<td>$Y = 0$</td>
</tr>
</tbody>
</table>

*If TRANSMIT is selected and deselected several times in a part program, the rotary axis must be repositioned every now and again to machine the same part of the workpiece.*

**MDI AUTOMATIC mode:**

<table>
<thead>
<tr>
<th>Selection of TRANSMIT (G131)</th>
<th>Deselection of TRANSMIT (G130)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = X_1 \cos (C_M - C_A)$</td>
<td>$X = 0$</td>
</tr>
<tr>
<td>$Y = X_1 \sin (C_M - C_A)$</td>
<td>$Y = 0$</td>
</tr>
</tbody>
</table>

$C_A$: Current angle of the rotary axis

When TRANSMIT selected in "MDI AUTOMATIC" mode the angle $C_A$ of the rotary from the last use of TRANSMIT is used in "AUTOMATIC" mode.
**Example:** Axis combinations for milling

Fictitious coordinate system: \( X,Y \)
Real coordinate system: \( X1,C',Z \)

<table>
<thead>
<tr>
<th>Progr. axis</th>
<th>Controlled axis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>after selection</td>
</tr>
<tr>
<td></td>
<td>from TRANSMIT</td>
</tr>
<tr>
<td>1 dimensional</td>
<td></td>
</tr>
<tr>
<td>( X )</td>
<td>( X1 + C' )</td>
</tr>
<tr>
<td>( Y )</td>
<td>( X1 + C' )</td>
</tr>
<tr>
<td>( X1 )</td>
<td>-</td>
</tr>
<tr>
<td>( C' )</td>
<td>-</td>
</tr>
<tr>
<td>( Z )</td>
<td>( Z )</td>
</tr>
<tr>
<td>2 dimensional</td>
<td></td>
</tr>
<tr>
<td>( X,Y )</td>
<td>( X1 + C',Z )</td>
</tr>
<tr>
<td>( X,Z )</td>
<td>( X1 + C',Z )</td>
</tr>
<tr>
<td>( Y,Z )</td>
<td>( X1 + C',Z )</td>
</tr>
<tr>
<td>( C',X1 )</td>
<td>-</td>
</tr>
<tr>
<td>( C',Z )</td>
<td>-</td>
</tr>
<tr>
<td>( X1,Z )</td>
<td>-</td>
</tr>
<tr>
<td>3 dimensional</td>
<td></td>
</tr>
<tr>
<td>( X,Y,Z )</td>
<td>( X1 + C',Z )</td>
</tr>
<tr>
<td>( X1,C',Z )</td>
<td>-</td>
</tr>
</tbody>
</table>

The combinations marked "-" cause a machine stop and the alarm 2043 “programming error during transformation” is output.
Programm example: face machining with TRANSMIT

Face machining with TRANSMIT

%100 L_f
N15 G01 F5000 X1=50 Z0 L_f  (P1)  Approaching the starting point in the real system
N20 G131 L_f
N25 G01 G94 F1000 X40 Y0 G42 D1 L_f  (P2)  Selection of the TRANSMIT transformation
N30 X20 Y30 L_f  (P3)  Selection of CRC/TNRC
N35 X=-50 L_f  (P4)
N40 G02 X=-50 Y=-30 J=-30 I0 L_f  (P5)
N45 G01 X20 L_f  (P6)
N50 X40 Y0 L_f  (P2)
N55 G40 X50 L_f  (P1)  Deselection of CRC/TNRC
N60 G130 L_f
N65 G00 X1=60 L_f  (P1)  Deselection of the TRANSMIT transformation
N70 M02 L_f  Real system retract tool

1) Since C’ cannot be entered, C is used
3.2.13.4 Machining accuracy with TRANSMIT

The machining accuracy with TRANSMIT is determined by
• the feed programmed for the fictitious axes
• the velocity of the real axes, i.e. the machining rate
• the measuring system used (encoder).

Programmed feed, machining rate:
The size of the feedrate is the path velocity of the programmed motion in the fictitious plane X,Y. However, the velocity of the real axes, i.e. the machining rate, is constantly changing (see Section 3.2.13.5). This entails accelerations which make it impossible to guarantee the accuracy of a pure milling machine.

Measuring system used:
The machining accuracy in the linear axis (X1 axis) depends on the resolution of the encoder used. The influence of the rotary axis (C axis) on the machining accuracy depends on the resolution of the angle position encoder by the following rule of thumb:
The real linear deviation \( s \) therefore is up to:

\[
\text{s} = \frac{R}{180}
\]

where \( R = \sqrt{X^2 + Y^2} \) : distance of the programmed position (radius) for the fictitious coordinate system zero.

\( \theta \); smallest, settable angle value in degrees, i.e. the resolution of the angle position encoder.

The influence of the angular resolution on the deviations \( s \) to be expected with various radii \( R \) is shown by following diagram and table. However, the real deviations are usually smaller than those calculated.

---

**Diagram:**

Deviation \( s \) on the surface of a rotary axis at an angular resolution of \( 10^{-4} \) degrees

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
3.2.13 Coordinate transformation TRANSMIT

### 3.2.13.5 Velocity monitoring with TRANSMIT

In "AUTOMATIC" mode the programmed velocity, i.e. the velocity in the fictitious coordinate system, is monitored.

To prevent the rotary axis turning faster than is defined in MD 280*, the feedrate might have to be reduced. If this is the case, the block related alarm 3083 “feedrate limitation fictitious axis” is output. The velocity of the other real axis is also a function of the programmed contour and is therefore not constant.

**Working area for the fictitious coordinate system:**

The fictitious coordinate system is divided into a **main working area** and a **prohibited area**. If the fictitious axes enter the **prohibited area** the monitoring system responds. Controlled movements of the fictitious axes are possible in the prohibited area if the override switch is used. The response of the monitoring system is affected by the override switch.

The angular velocity of the rotary axis is monitored and if the maximum velocity (defined in a machine data) is violated, the velocity is limited to this maximum value. At the same time the alarm 2035 “feedrate limitation” is output and except in JOG, INC and REPOS modes the alarm 3083 “feedrate limitation fictitious”. These alarms must be acknowledged, alarm 2035 with a RESET and alarm 3083 with the acknowledgement key. An NC stop is not triggered. Because of the limitation, form errors occur on the contour when the alarm 2035 is output.

**Contour errors can occur with the TRANSMIT function. These can occur when the distance between the workpiece centre and the turning centre of the milling cutter is too small (limitation of the angle velocity of the rotary axis). Use a cutter with a greater radius. This increases the distance and reduces the error. It is advisable to work with the greatest possible cutter radius.**
The boundary between the **main working area** and the **prohibited area** is the reference contour.
Please consult the machine manufacturer for the data of the areas.
Also consult the machine manufacturer for the maximum axis velocities.
A path which runs through the centre cannot be traversed. Angular velocities of up to 180°
sampling cycle would occur and the maximum rotary axis velocity would be exceeded. The
rotary axis would have to be rotated through 180° at the turning centre. To avoid this, contours
of this kind must be spread over at least three blocks (2nd to 4th block):

1st block: Contour up to the turning centre.
2nd block: Retraction of cutter.
3rd block: Rotation of rotary axis through 180°.
4th block: Infeed of cutter.
5th block: Contour behind the turning centre.
3.3 Reference point approach in part program (G74)

3.3.1 Function description

The function “reference point synchronization in the part program” enables the user to approach the reference point of a real programmed NC axis from the part program using a G function. The machine control (servo and traverse behaviour) and the communication interface of the CNC (actual value, service display) are used unchanged. The function “REF in PP” is a standard function which is applicable to both linear and rotary axes.

3.3.2 Starting the function

The function “REF in PP” can be activated in the “AUTOMATIC” and “MDI AUTOMATIC” modes. Syntax of the function in the part program:

N <No.> G74 < axis name> LF

Explanations:

<No.> = Block number
<Axis name> = Address of a generated NC axis

Example:

N1000 G74 C LF

With an extented address name of an NC axis, e. g. Y1 the following applies:

N <No.> G74 Y1=LF

Example:

N1000 G74 Y1= LF

Remarks:

1. For the 810/820 T/M a restriction to one NC axis per G74 block applies.
2. Rotary and linear axes can be specified.

The G74 function must be in a block of its own without other functions!
3.4 Second spindle

The "Second spindle" function is an ordering data option.

The second spindle can be used as main spindle or auxiliary spindle, e.g. as driven tool for the TRANSMIT function.

In the case of two spindles, one spindle must be defined as leading spindle for each channel (1 or 2). For the exact specification refer to the machine manufacturer. Setpoints and actual values for the valid leading spindle are displayed in the basic display operating modes.

The following G functions are assigned to the leading spindle:

- G33, G34, G35  Tapping, thread cutting
- G04 Sn=  Dwell time related to spindle revolutions, n being the number of the leading spindle
- G95  Feedrate per revolution
- G96 Sn=  Constant cutting velocity, n being the number of the leading spindle
- G97  Extended number of spindles from G96
- G63  Thread without encoder

M and S functions can be programmed with or without address extension (with the exception of G04, G96). The programmed M and S functions refer to the leading spindle when programming without address extension. The address extension is programmed directly behind the address character S or M, then "=" followed by the auxiliary function value. Spindle functions may be programmed only for one spindle in a part program block.

Programming setting data for the spindle

<table>
<thead>
<tr>
<th>Setting Data</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD 4010 (spindle 1)</td>
<td>Spindle speed limitation for G92</td>
</tr>
<tr>
<td>SD 4011 (spindle 2)</td>
<td>Spindle speed limitation for G92</td>
</tr>
<tr>
<td>SD 4020 (spindle 1)</td>
<td>Oriented spindle stop (M19)</td>
</tr>
<tr>
<td>SD 4021 (spindle 2)</td>
<td>Oriented spindle stop (M19)</td>
</tr>
<tr>
<td>SD 4030 (spindle 1)</td>
<td>Spindle speed limitation</td>
</tr>
<tr>
<td>SD 4031 (spindle 2)</td>
<td>Spindle speed limitation</td>
</tr>
<tr>
<td>SD 4040 (spindle 1)</td>
<td>Smoothing constant for thread</td>
</tr>
<tr>
<td>SD 4041 (spindle 2)</td>
<td>Smoothing constant for thread</td>
</tr>
</tbody>
</table>

With G26 S1=500, for example, the spindle speed limitation is entered in setting data 4030.
With G92 S1=500, for example, the spindle speed limitation for G92 is entered in setting data 4010.
With M1 = 19 S1 = 180, the setpoint position of the spindle in degrees is entered in setting data 4010 in addition to the oriented spindle stop.
Setting data 4040 and 4041 replace setting data SD1.
3.5 On-the-fly synchronization of rotary axis

The functions:
- Transmit for turning machines
- Tapping without compensating chuck (interpolating)
- Thread cutting without compensating chuck (interpolating)
- Cylindrical interpolation for turning machines

cause switchover from spindle operation to rotary axis operation and vice versa.

By means of the "On-the-fly synchronization of rotary axis" function, synchronization of the rotary axis is performed already when braking the spindle.

Switching from spindle operation to rotary axis operation is performed by means of the auxiliary function.

\[ M_n = m \]

Specified by machine manufacturer (e.g. 70)

Specification of the spindle number (when 0 the leading spindle is addressed)

**Example:** \( M_1 = 70 \)

Switching from spindle operation to rotary axis operation is performed by means of the auxiliary function.

\[ M_n = q \]

Specified by machine manufacturer (e.g. 71)

Specification of the spindle number (when 0 the leading spindle is addressed)

**Example:** \( M_1 = 71 \)

Internally, command @ 714 is executed.

The switching commands must be programmed in an NC block. In the NC reset state switching can be effected by means of overstore. Before selecting block search, select or deselect rotary axis operation by means of overstore.

**Notes:**
- Spindle operation
  Position display for the rotary axis is "0". Rotary axis operation may be selected repeatedly.
  The rotary axis must neither be programmed nor traversed manually (otherwise alarm 1961 is generated).

- Rotary axis operation
  The spindle speed remains "0". Repeated selection of the rotary axis operation may be programmed.
  The spindle must not be programmed.

- Overstore of the changeover in an interrupted program is not possible.

- Depending on the machine manufacturer’s specifications, rotary axis operation remains active or is deselected with RESET.

- Selection and deselection of the rotary axis operation must be performed in the same channel.
4 Miscellaneous, Switching and Auxiliary Functions

4.1 M, S, T, H

The miscellaneous, switching and auxiliary functions contain primarily technological specifications which are not programmed in the words provided with address letters F, S, and T:

- Miscellaneous functions M (2-digit)
- Spindle speed S (5-digit)
- Tool number T (4-digit)
- Auxiliary function H (4-digit)

Each block may contain up to three M functions, one S function, one T function and one H function. They are output to the interface controller in the following sequence: M, S, T, H.

For further specifications refer to the machine manufacturer.

If the functions are output during axis movement, the following applies:

If a new value must be active before the axes are traversed, the new function must be written in the preceding block.

4.2 Miscellaneous functions M

Some miscellaneous functions are defined in DIN 66025, Part 2; others are defined by the machine tool manufacturer.

M functions can be programmed with or without extended addressing, M** or M* = *** represent a figure in the range of 0 to 9

Miscellaneous functions from M0 to M99 and from M1 = 0 to M9 = 99 result. M functions with the extended address 0 are treated just as functions without extended address and decoded regularly.

M00 or M*=0 Programmed stop (unconditional)

Function M00 permits the program to be interrupted, e. g. in order to perform a measurement. On termination of the measurement, machining can be continued by pressing the "NC START" key. The information entered is retained. Miscellaneous function M00 is active in all automatic operating modes. Whether or not the spindle drive is also stopped must be determined from the specific Programming Instructions for each machine. M00 is also active in a block without position data.
M01 or M*=1 Programmed stop (conditional)
Function M01 is the same as M00 but is active only if the “Conditional stop (M01) active”
function has been activated by means of the softkey or at the interface. Text in reverse video
cannot be displayed with M01.

M02 or M*=2 Program end
M02 signals the end of the program and resets the program to the beginning. It is written in
the last block of the program. The controller is reset. M02 may be written in a separate block
or in a block containing other functions. The read-in process can be stopped with M02 (setting
data).

M17 or M*=17 Subroutine end
M17 is written in the last block of a subroutine. It may be written in a separate block or in a
block containing other functions. M17 must not be written in the same block as a subroutine
(nesting), otherwise the subroutine will not be executed.

M30 or M*=30 Program end
Function M30 is the same as M02.

M36, M37 or M*=36, M*=37 feed reduction ratio 1:100
M36 The programmed feedrate is valid again
Deselect M37
M37 The programmed feedrate is reduced by 1:100. Is active for G94, G95 and G96.
Applies for path feed and all simulation feedrates.

Example:
N10 G94 X30 P1000 L_F
: Feedrate 1000 mm/min active
N30 M37 L_F
: Feedrate 10 mm/min active
N50 M36 L_F
: Feedrate 1000 mm/min active

M1=3, M2=3, M1=4, M2=4, M1=5, M2=5, M1=19, M2=19 Main spindle control
(M19 only with pulse encoder at main spindle).
The extended addressing establishes the relation to spindle 1 or spindle 2.

The oriented spindle stop function is available as an option.

In the version with analog spindle speed output the following M words are fixed for spindle
control:
M1=3, M2=3 Clockwise spindle direction of rotation Spindle 1, spindle 2
M1=4, M2=4 Counter-clockwise spindle direction of rotation Spindle 1, spindle 2
M1=5, M2=5 Non-oriented spindle stop Spindle 1, spindle 2
M1=19, M2=19 Oriented spindle stop Spindle 1, spindle 2
M0=... Applies to the leading spindle
M19 S.. can be used to perform an oriented main spindle stop. The relevant angle is programmed under S in degrees. The angle is measured from the zero mark in the clockwise direction of rotation. M19 S.. must be in a block of its own.

The action of the angle programmed under address S is modal. If M19 is programmed without S, the value stored under S is valid for the angle, i.e. a repeated stop can be effected simply by programming M19.

The angle may also be input via the operator panel under “Spindle setting data”. A MACHINE DATA can specify whether the spindle has to be at rest before the axis motion programmed in the next block is started or whether the next block is enabled during spindle positioning.

M19 wählt M03 bzw. M04 nicht ab.

**Freely assignable miscellaneous functions**

All miscellaneous functions with the exception of M00, M01, M02, M03, M04, M05, M17, M19, M30, M36 and M37 are freely assignable.

The extended address notation must be used and the channel number specified for all freely assignable miscellaneous functions:

**Example:**

M3=124

3... NC channel number

124... M function 124.

Further details on the use of the various functions can be found in the specific machine program key (see Section 12). The meaning of some of these functions is defined in DIN 66025.

### 4.3 Spindle function S

The data below can optionally be entered under address S:

- Spindle speed in rev/min or 0.1 rev/min *
- Cutting speed in rev/min or 0.1 rev/min *
- Spindle speed limitation in rev/min or 0.1 rev/min *
- Spindle stop in degrees (M19)
- Spindle dwell in revolutions (see G04).

*) The speed and cutting speed must be programmed in the same input unit.
The extended address notation must be used and the spindle number specified for the S word:

Example:

S2=1000
2... Specification of spindle no.
1000... Spindle speed (rev/min).

For not existing spindles the S word can also be used as auxiliary function.

S3=... S9=12345 (max. 5 digits for the value)

With one spindle: 2, 3, ..., 9
With two spindles: 2, 3, 4, ..., 9

4.4 Auxiliary functions H

One auxiliary function per block can be entered under address H for machine switching functions or movements not covered by numerical control. H can be programmed with 4 decades. The meaning of the functions is described in the programming instructions issued by the machine-tool manufacturer.

The extended address notation must be used for the H word:

H=...

4.5 Tool number T

The tool number determines the tool required for a machining operation:

Example:

T1234
T Address
1234 Tool number (max. 4 decades).

The extended address notation must be used and the channel number specified for the T word:

T1=1234 Selection of tool No. 1234 for processing in channel 1.
# 5 Subroutines

## 5.1 Application

If the same machining operation must be performed repeatedly when a workpiece is machined, it can be entered as a subroutine and called as often as desired in the part program or by means of manual data input.

The program memory can be used to store 200 part programs and subroutines at any one time. Subroutines should preferably be programmed using incremental position data. The tool is set to the start position in the part program before the subroutine call. The machining sequence at the workpiece can then be repeated at various points on the workpiece without modifying the dimensions in the subroutine.

## 5.2 Subroutine structure

A subroutine comprises:

- the subroutine start (header),
- the subroutine blocks,
- the subroutine end.

![Subroutine structure](image)

The subroutine start comprises address L and the max. four-digit subroutine number (see program key, Section 12).

> The subroutine start is not in tape format. The subroutine end is used to return to the part program and is defined by the end character M17. M17 is written in the last block of the subroutine. It is permissible to write other functions (except address L) in this block.
5.3 Subroutine call

The subroutine is called in a part program via address L with the subroutine number and the number of passes with address P. If a subroutine number is programmed without address P, P1 is assumed (1 pass).

Example:

L123 P1 LF
L123 Subroutine number (1...9999)
P1 Number of passes (1...99)

The following should be noted during programming:

- The subroutine call must not be written in a block together with M02, M30 or M17, otherwise the subroutine is not executed.
- If the subroutine is called whilst the cutter radius compensation (CRC) function is selected, the section on special cases for CRC ("Blocks without path addresses") should be referred to.
- If the subroutine call is written in a block containing other functions, the subroutine is called at the end of the block.
- The special functions L81 to L89 can be called with G functions G81 to G89. G80 cancels G81 to G89 (initial setting).
  
  G81 to G89 and L81 to L89 in in a block trigger alarm 3006 “wrong block structure”.
- G81 to G89 are modal.
Subroutines can be called not only from a part program, but also from other subroutines. This process is referred to as subroutine nesting. Nesting of the subroutine to a **depth of four** is possible.
Example: Subroutine nesting two-deep

Program execution:

% 4011 L_F
N5 G90 G94 F500 S2000 M3 T3 L_F
N10 G00 X52 Z60 L_F
N15 L230 P1 L_F
End of program

Program header % 4011 (main program)
Select absolute dimensioning, feed mm/min, S, D, T, M function
Rapid traverse to P1
Call subroutine L230 with 1 pass

End of program

Program header L230 (subroutine)
Program part for turning off the segment
Program part for turning off the segment
Call subroutine L240 with 2 passes
End of subroutine L230

Program header L240 (subroutine)
Program part for turning off the segment
Program part for turning off the segment
Program part for turning off the segment
End of subroutine L240
6 Parameters

6.1 Parameter programming

Parameters are used in a program to represent the numeric value of an address. They are assigned values within the program, and can thus be used to adapt a program to several similar applications (e.g. different feedrates, different spindle speeds for various materials, different operating cycles).

A parameter comprises address R and a number with up to 3 digits. In the basic configuration 700 channel-specific and a total of 300 parameters are available to the control; they are subdivided into transfer parameters, computing parameters, parameters defined as being channel-dependent or channel-independent and central parameters (see CL800 language description).

<table>
<thead>
<tr>
<th>Channel 1 to Channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R00</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>R49</td>
</tr>
<tr>
<td>R50</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>R99</td>
</tr>
</tbody>
</table>

R0 to R49
Typical application/channel:
Assign parameters to cycles and subroutines.

R50 to R99
Typical application/channel:
For calculations within cycles and subroutines. For nested subroutines, the same local parameters can be used. With cycles or subroutine call by @040 ... 043, a R parameter stack saves the data used up to that point and stores them after return to the calling program.

R110 to R199
Typical application for each channel: Memory for data which must be accessible for part programs and subroutines. R110 to R199 are reserved if Siemens measuring cycles are used.

R200 to R219
Assigned internally (cycle converter)

R220 to R239
WS800 compiler

R240 to R299
Reserved for internal assignment.

R300
Stack pointer for @040, @041, @042, @043

R301 to R499
Stack area for @040, @041, @042, @043

R500 to R699
Reserved for user

R700 to R999
Typical application: General memory for all NC channels, e.g. as buffer for target positions which are used by a different channel.

Structuring and application of R parameters
A parameter can be assigned instead of a value to all addresses with the exception of N.

\[ N5 \text{ Z} = R5 \text{ L} \text{ F} \]

The word structure of the various address must be observed (decimal notation or use of sign does not apply to all addresses).

Incorrect:

\[ R1 = 51120.98 \quad H = -R1 \]

This numerical value is not possible at address H (Alarm 3000 general programming error).

### 6.2 Parameter definition

Parameter definition is used to assign certain numeric values with signs to the various parameters.

The parameters can be defined either in part programs or in subroutines.

\[ R1 = 10 \text{ L} \text{ F} \]

Parameter definition, subroutine call and switching functions may be written in a single block. The value defined for a parameter is assigned direct to the address.

**Example:**

\[
\begin{align*}
\% &\text{ 5772 L} \text{ F} \\
N5 &\ldots \\
N35 &R1=10 \quad R29=-20.05 \quad R5=50 \quad \text{ L} \text{ F} \\
N40 &L51 \quad P2 \quad \text{ L} \text{ F} \\
N45 &M02 \quad \text{ L} \text{ F} \\
L51 \\
N5 &Z=-R5 \quad B=-R1 \quad \text{ L} \text{ F} \\
N10 &X=-R29 \quad \text{ L} \text{ F} \\
\ldots \\
N50 &M17 \quad \text{ L} \text{ F} \\
\end{align*}
\]
6.3 Parameter calculation

Parameter linking

All four basic arithmetic operations are possible with parameters. The sequence of operations is however crucial to the result of the calculation. The rule whereby multiplication and division are performed before addition and subtraction does not apply here.

<table>
<thead>
<tr>
<th>Arithmetic operation</th>
<th>Programmed arithmetic operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>R1 = 100</td>
</tr>
<tr>
<td>Assignment</td>
<td>R1 = R2</td>
</tr>
<tr>
<td>Negation</td>
<td>R1 = – R2</td>
</tr>
<tr>
<td>Addition</td>
<td>R1 = R2 + R3</td>
</tr>
<tr>
<td>Subtraction</td>
<td>R1 = R2 – R3</td>
</tr>
<tr>
<td>Multiplication</td>
<td>R1 = R2 * R3</td>
</tr>
<tr>
<td>Division</td>
<td>R1 = R2 / R3</td>
</tr>
</tbody>
</table>

The result of an arithmetic operation is written in the first parameter of a logic operation; its initial value is thus overwritten and lost when the logic operation is performed. The values of the second and/or third parameters are retained.

Value assignment amongst parameters

If the value of one parameter is to be assigned to another parameter, then:

\[
R1 = R3 \text{ LF}
\]

Calculation using numbers and parameters

- Addition and subtraction of numbers and parameters in conjunction with addresses

It is possible to add a parameter to the value of an address or to subtract it from it. Calculation signs must be written. No sign signifies a positive number.

The "+" sign must always be entered.

\[
X = 10 + R100 \text{ LF}
\]

or

\[
X = R100 + 10 \text{ LF}
\]
Example:
N05  R1=9.7  R2 = – 2.1  LF
N10  X=20.3 + R1  LF
N15  Z=32.9 – R2  LF

Result:
X=30
Z=35.

• Calculations using numbers and parameters

It is possible to multiply, divide, add and subtract absolute numbers and R parameters.

\[ R10=15+ R11 \]  \[ LF \]

• several separate equations can be programmed in one block.

\[ R1 = R2 + 23 \quad R50 = R37 \cdot 3 \quad R99 = R27 / R13 \quad LF \]

6.4 Parameter string

\[ R1 = R2 + R3 – R4 \cdot R5 / R6 \ldots \quad LF \]

All 4 basic arithmetic operations are permissible in any sequence. A parameter string is limited by the **block length of 120 characters maximum**.

The calculations are performed as follows:

Step 1  \( R1 = R2 \)
Step 2  \( R1 = R1 + R3 \)
Step 3  \( R1 = R1 – R4 \)
Step 4  \( R1 \cdot R5 \)
Step 5  \( R1 / R6 \)

\[ \begin{align*}
\text{Step 1} & \quad R1 = R2 \\
\text{Step 2} & \quad R1 = R2 + R3 \\
\text{Step 3} & \quad R1 – R4 \\
\text{Step 4} & \quad R1 \cdot R5 \\
\text{Step 5} & \quad R1 / R6 \\
\end{align*} \]

Instead of a link R parameter (not a result parameter), constants and pointers (pointers to R parameters) are allowed with address P in the parameter string.
Example:
\[ R1 = R2 + 10.5 - P3 \cdot R5/R6 \ldots \]

The result parameter must be an R parameter.

\[ P3: \]
\begin{itemize}
  \item \textbf{P} Address of pointer
  \item \textbf{3} Pointer to R parameter R3, i.e. the contents of R3 are the address of an R parameter whose value is included in the parameter string.
\end{itemize}

Any number of arithmetic operations may be performed in a block, e.g. multiplication, parameter string, addition, etc.; the maximum permissible block length of 120 characters must not however be exceeded. The individual links are calculated in the programmed sequence.

Value range:
- Minimum value: \( 1 \cdot 10^{-8} \)
- Maximum value: 99999999

Display:
- Floating point (\( \pm 1.2345678 \)) to (\( \pm 12345678. \))

### 6.5 Programming example with parameters

**Parameters for subroutines without values**

In the case of subroutines without values the current data are transferred by means of parameters \( R00 \) to \( R99 \).

These parameters are used in subroutines instead of numerical values. **Up to 10 parameters** may be programmed for each block.

When a subroutine is called, it must be ensured that the parameters used exhibit the correct values. The values are assigned to the parameters in the main program.
Example:

The contour to be machined is stored in a subroutine.

L47 Lg
N5 G90 Z=R0 Lg
N10 X=R1 Lg
N15 A=R2 A=R4 X=R6 Z=R5 B=R3 Lg
N20 Z=R7 Lg
N25 X=R8 Lg
N30 Z=R0 M17 Lg

Subroutine call:
N . . X . . Z . . Lg
N . . L47 P1 R0=125 R1=50 R2=170 R3=30
R4=135 R5=30 R6=120 R7=20 R8=140 Lg
7 Contour Definition

7.1 Blueprint programming

Multi-point cycles for direct programming in accordance with the workpiece drawing are provided for blueprint programming. The points of intersection of the straight lines are entered as coordinate values or via angles.

The various straight lines may be joined together directly in the form of a corner, rounded via radii or chamfered. Chamfer and transition radii are specified only by means of their size. The geometrical calculation is performed by the controller. The end position coordinates may be programmed using either absolute or incremental position data.

Angle (A): Input resolution 0.00001 corresponds to $10^{-5}$ degrees.

In the clockwise coordinate system the angle (max. $359.99999^\circ$) is always measured from the horizontal axis direction to the vertical axis direction.

In the example the axis addresses were defined with $X$, $Z$.

Clockwise system and operating area after turning centre

Clockwise system and operating area before turning centre
### 7.2 Contour definition programming

The elements described are valid for a turning machine with an operating area after the turning centre.

Examples 1 to 8 represent the basic elements of contour definition programming. These basic elements can be combined in a number of ways. The addresses for the angle (in this case A) and the radius (in this case B) are freely selectable in the controller. The addresses must not be assigned more than once. Extended addresses are not possible for angle (A) and radius/chamfer (B, B⁻).

<table>
<thead>
<tr>
<th>Function</th>
<th>Programming</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) 2-point definition</td>
<td>[N \ldots A \ldots X_2 \ldots (or \ Z_2) \text{ LF}] The second end coordinate is calculated by the controller.</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td>(2) Circular arc</td>
<td>[N \ldots \text{G02 (or G03) I \ldots K \ldots B \ldots X_2 \ldots (or \ Z_2) \text{ LF}] The circular arc is limited to one quadrant. The second end position coordinate is calculated by the controller. In the contour definition parameters I and K must both be programmed, even if one of the values is zero.</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>(3) 3-point definition</td>
<td>[N \ldots A_1 \ldots A_2 \ldots X_3 \ldots Z_3 \ldots \text{ LF}] The controller calculates the coordinates of the vertex and generates 2 blocks. Angle A₂ is referred to the second straight line.</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>Function</td>
<td>Programming</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>(4) Chamfer</td>
<td>N... X2... Z2... B-... LF</td>
<td><img src="Diagram1" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>B-... means insert a chamfer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B ... means insert a radius</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(The “minus” character is not a sign here; instead it is a special identifier for B = chamfer).</td>
<td></td>
</tr>
<tr>
<td>(5) Radius</td>
<td>N... X2... Z2... B...</td>
<td><img src="Diagram2" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>N... X3... Z3... LF 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The inserted radius must not be larger than the smaller of the two paths.</td>
<td></td>
</tr>
<tr>
<td>(6) Straight line - circular arc (tangent)</td>
<td>N.. G02 (or G03) A.. B.. X3.. Z3.. LF</td>
<td><img src="Diagram3" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>The circular arc must not exceed 180°. The sequence A (angle) followed by B (radius) must be used.</td>
<td></td>
</tr>
<tr>
<td>(7) Circular arc - straight line (tangent)</td>
<td>N.. G02 (or G03) B.. A.. X3.. Z3.. LF</td>
<td><img src="Diagram4" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>The circular arc must not exceed 180°. The sequence B, A must be used. No radius may be inserted in X3, Z3.</td>
<td></td>
</tr>
<tr>
<td>(8) Circular arc - circular arc (tangent)</td>
<td>N.. G02 (or G03) I1.. K1.. I2.. K2.. X3.. Z3.. LF</td>
<td><img src="Diagram5" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>The preparatory function is programmed for the first circular arc. The second preparatory function is always the opposite of the first one and is not programmed. The interpolation parameters of the second circle are referred to the end position of this circle. Both interpolation parameters must be programmed, even if one value is zero.</td>
<td></td>
</tr>
</tbody>
</table>

1) Second block may also be a contour definition
<table>
<thead>
<tr>
<th>Function</th>
<th>Programming</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) + (4) 2-point definition + chamfer</td>
<td>N... A.. X₂... (or Z₂...) B-.. LF N... X₃.. Z₃.. LF ¹</td>
<td><img src="image1" alt="Example 1" /></td>
</tr>
<tr>
<td>(1) + (5) 2-point definition + radius</td>
<td>N... A.. X₂... (or Z₂...) B... LF N... X₃.. Z₃.. LF ¹</td>
<td><img src="image2" alt="Example 2" /></td>
</tr>
<tr>
<td>The inserted radius must not be larger than the smaller of the two paths.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) + (4) 3-point definition + chamfer</td>
<td>N.. A₁.. A₂.. X₃.. Z₃.. B-.. LF</td>
<td><img src="image3" alt="Example 3" /></td>
</tr>
<tr>
<td>(3) + (5) 3-point definition + radius</td>
<td>N.. A₁.. A₂.. X₃.. Z₃.. B.. LF</td>
<td><img src="image4" alt="Example 4" /></td>
</tr>
<tr>
<td>(3) + (4) + (4) 3-point definition + chamfer + chamfer</td>
<td>N... A₁.. A₂.. X₃.. Z₃.. B₁-.. B₂-.. LF ¹ N... X₄.. Z₄.. LF ¹</td>
<td><img src="image5" alt="Example 5" /></td>
</tr>
<tr>
<td>Addition of a second chamfer at end position (X₃, Z₃).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example

**Function** | **Programming** | **Example**
--- | --- | ---
(3) + (5) + (5) | \( N \ldots A_1 \ldots A_2 \ldots X_3 \ldots Z_3 \ldots B_1 \ldots B_2 \ldots \text{LF} \)
3-point definition + radius + radius | Addition of a second radius at end position \((X_3, Z_3)\).

(3) + (4) + (4) | \( N \ldots A_1 \ldots A_2 \ldots X_3 \ldots Z_3 \ldots B\ldots B\ldots \ldots \text{LF} \)
3-point definition + chamfer + radius | Addition of a radius at end position \(X_3, Z_3\). The next block is always taken into consideration automatically.

(3) + (5) + (4) | \( N \ldots A_1 \ldots A_2 \ldots X_3 \ldots Z_3 \ldots B\ldots B\ldots \ldots \text{LF} \)
3-point definition + radius + chamfer | Addition of a chamfer \(B\) at end position.

**B0** must be programmed for corners where **no chamfer or radius** must be inserted if a further **radius or chamfer follows** in the contour definition.

---

**When this is programmed, the controller generates a block with a distance 0. This must be noted in relation to the action of the TNRC.**

**B-0 is interpreted as B0.**

**A radius or chamfer can only be inserted between two linear blocks.**

The sequence of addresses A, X, Z, B, F, etc. is freely selectable; angles and radii must however be entered in the **sequence** described above (first angle before second angle, first radius before second radius **in machining direction**).

---

1) Second block may also be a contour definition.
7.3 Operation of function G09, F, S, T, H, M in contour definition

If G09 is programmed in a contour definition block, it is not active until the end of the block, in other words until the end position is reached. G09 is automatically generated by the controller at irregular points (corners, edges) in the contour definition.

- If F, S, T, H or M is programmed in a contour definition, it is active at the start of the block.
- M00, M01, M02, M17 and M30 are active at the end of the block.

7.4 Linking of blocks

It is possible to link blocks with or without angle inputs and with inserted radii or chamfers in any sequence.

Example: Linking of blocks

| N10 | Z... B5 Lf | Straight line with radius |
| N15 | A... X... B7 Lf | Straight line with radius |
| N20 | A1... A2... X... Z... B9 B11 Lf | 3-point definition with radius at both ends |
| N25 | Z... Lf | Straight line |

Linking of blocks
7.5 Programming examples: turning machine

Angle refers to the starting position; angle refers to the missing vertex. The end position can be programmed using absolute position data G90 or incremental position data G91. Both end position coordinates must be specified. The controller determines the vertex from the known starting position, the two angles and the end position.

Example: External machining

Example: Internal machining
**Drawing dimensions**
The starting position can be defined anywhere outside the inner taper.

The perpendicular through the starting position and the extension of the inner taper yield point of intersection A.

The program is then as follows:

```plaintext
N10 G00 Xstart Zstart Lf
N15 G01 A90 A184 X... Z... F... Lf
```

**Example:** Contour definition programming for external machining
Blueprint programming:

% 495 Lf
N5   G00 G90 X0 Z332 Lf
N10  G01 G09 A90 X66 B-8 F0.2 LF
N15  A180 A90 X116 Z246 B8 LF
N20  G03 B40 A175 X140 Z130 LF
N25  G01 A135 A180 X220 Z0 LF
N30  M02 LF

Absolute dimensioning, starting point definition
2-point definition + chamfer (B-8) (concatenation with B-)
3-point definition + radius (B8)
Circular arc - straight line
3-point definition
End of subroutine

7.6 Miscellaneous functions in linked blocks

Blocks are said to be linked whenever they are joined together by means of radii or chamfers.

**Example:** Miscellaneous functions in linked blocks

A block with miscellaneous functions may be located between two linked blocks:

```
N10  G01 G09 A90 X66 B-8 F0.2 LF (P1)
N101 M... H... ... LF
N15  A180 A90 X116 Z246 B8 LF
```

Miscellaneous functions are active at position P11 (see above).

**Relief-cutting** is thus effected at position P11. The F value programmed in block N10 is active at the start of block N10.
8 Tool Offsets

8.1 Tool data

The geometrical tool data for the tool are stored under tool offset number D:
Length \( \pm 9999.999 \text{ mm} \), radius \( \pm 999.999 \text{ mm} \); T number 4 decades (input resolution 1\(\mu\text{m}\)).

The tool number, tool type, geometry, basic dimension and wear of all active tools are stored in the tool offset memory of the NC.

- The memory is subdivided into 99 tool offsets (D1 to D99).
- Each block consists of 10 columns corresponding to 10 tool parameters (P0 to P9) and has the following structure:

  \[
  \begin{align*}
  \text{D1 ... D99} & \qquad \text{Wear data} \\
  \text{P0} : & \quad \text{Tool number} \\
  \text{P1} : & \quad \text{Type} \\
  \text{P2} : & \quad \text{L1 Geometry} \\
  \text{P3} : & \quad \text{L2 Geometry} \\
  \text{P4} : & \quad \text{Diameter/radius} \\
  \text{P5} : & \quad \text{L1 Wear} \\
  \text{P6} : & \quad \text{L2 Wear} \\
  \text{P7} : & \quad \text{Diameter/radius} \\
  \text{P8} : & \quad \text{L1 Base} \\
  \text{P9} : & \quad \text{L2 Base} \\
  \end{align*}
  \]

- The structure of the tool offset block is defined by the tool type (P1).

**Breakdown of tool type P1:**

- Type 0 Tool not defined
- Type 1 ... 9 Lathe tools, position of tool tip
- Type 10 ... 19 Tools with active length compensation only (e.g. drills)
- Type 20 ... 29 Tools with radius compensation and one length compensation (e.g. milling cutters)
- Type 30 ... 39 Tools with radius compensation and two length compensations (e.g. angle tool noses)

- The tool offset is called in up to 3 decades via D1 to D99.
- The tool offset is cancelled with D0.

The offset is not executed until the corresponding axis is programmed. The geometry and wear are updated, for example, by measuring cycles in the NC.

*The wear and basic dimension are summated in the NC in accordance with the machine data.*

Tool offset may be entered via both the operator panel and the data input interface. No block numbers may be programmed (see tape formats).

The machine manufacturer can limit the maximum wear specification to \( \pm 0.999 \text{ mm} \) instead of \( \pm 9.999 \text{ mm} \).
8.2 Tool offset without using tool nose radius compensation (TNRC)

The active tool offset is derived from the sum of the tool length compensations and any external, additive tool length compensation. The sum corresponds to dimension XSF or ZSF.

The path of tool nose radius centre S is programmed. The length compensation is referred to the tool nose radius centre.
Incorporation of compensation

The difference between the old and new offset values is formed when the tool offset number is changed.

The question as to whether the difference

- is traversed directly following the change
- or is not taken into consideration until the programmed traverse of the corresponding axis is resolved on start-up.

*With TNRC type 1 to 9 (G41, G42) the difference in addition to the tool nose radius is traversed in both axes. No axis command is required for traversing the tool offset or a difference.*
### Structure of tool offset memory

<table>
<thead>
<tr>
<th>T no.</th>
<th>Type</th>
<th>Geometry</th>
<th>Wear</th>
<th>Basis (supp. TO)</th>
<th>Tool call</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td>P4</td>
<td></td>
</tr>
<tr>
<td>123...8</td>
<td>1...9</td>
<td>Length 1</td>
<td>Length 2</td>
<td>Radius</td>
<td>Length 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Turning tool

![Diagram of turning tool]

<table>
<thead>
<tr>
<th>T no.</th>
<th>Type</th>
<th>Geometry</th>
<th>Wear</th>
<th>Basis (supp. TO)</th>
<th>Tool call</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td>P4</td>
<td></td>
</tr>
<tr>
<td>123...8</td>
<td>1...9</td>
<td>Length 1</td>
<td>Length 2</td>
<td>Length 1</td>
<td>Length 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Grooving tool

![Diagram of grooving tool]
### 8.3 Tool offset using tool nose radius compensation

The workpiece contour can be programmed in conjunction with TNRC. The length compensation entered is referred to cutting point “P”. The tool nose radius $R_S$ and the position of the cutting point $P$ must also be entered.

The controller then computes the path to be traversed. There is no contour error. The tool nose radius compensation takes effect at the end position of the block in which it is called (G41, G42), i.e. the next block is executed correctly.

In order to calculate the tool nose radius compensation, the control requires the tool nose radius (compensation value P4) and details of how the cutting tool is clamped in the tool carrier. Identifier $P1 = 1$ to $P1 = 9$ must be entered in tool offset column P1 (tool type).

The position of point $P$ relative to tool nose radius centre point $S$ is now specified.

If tool nose radius centre $S$ is used as the reference point to determine the tool length compensation as opposed to point $P$, identifier $P1 = 9$ must be entered. Identifier $P1 = 0$ is not allowed.
The diagram shows the **possible positions of the cutting tool** with associated **identifiers** P1 = 1 to P1 = 9.

The values in **brackets** apply to machining **before the turning centre**.

Position of the theoretical tool tip P in relation to the tool nose centre point S (viewing direction always from S to P).

---

*Machining after turning centre (before turning centre)*
9 Tool Nose Radius Compensation (TNRC)

All stop positions for single blocks are marked S. The respective block numbers are shown in brackets. In the block following the selection block, a block start vector (length R) is created perpendicular to the programmed path.

Contour elements are marked with a thick line (——)

9.1 Selection of TNRC

The compensation mode is selected in the fixed plane with preparatory functions G41/G42 and offset number D.

The compensation is effected to the left of the workpiece contour (in the traversing direction) with G41 and to the right of the workpiece contour with G42 (in traversing direction).

Example of a turning tool:

When selecting TNRC, two or three program blocks are always read in for calculating the point of intersection.

The compensation mode can only be selected in a program block when G function G00, G01, G10 or G11 is active. Tool number D0 is assigned the compensation value 0; no compensation is selected. If the alarm "Option 3D interpolation not available" is generated, the "3D interpolation" function must be activated.

The diagrams below show the compensation selected for various approach angles. The diagrams are shown with G42. In the case of programs with G41, the angle transition β=360°-.
### 9.1 Selection of TNRC

<table>
<thead>
<tr>
<th>Programmed path</th>
<th>&gt;180°</th>
<th>Compensated path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S(N05)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Arc

| X               |       |                  |
| Z               |       |                  |
| G42             |       |                  |
| N05             |       |                  |
| R               |       |                  |
| S(N05)          |       |                  |
| N10             |       |                  |

**Selection of compensation mode at > 180°**
Selection of compensation mode

<table>
<thead>
<tr>
<th>90°</th>
<th>180°</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Straight line</strong></td>
<td><strong>Arc</strong></td>
</tr>
</tbody>
</table>

Selection of compensation mode at 90° 180°

<table>
<thead>
<tr>
<th>&lt;90°</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Straight line/straight line</strong></td>
</tr>
</tbody>
</table>

Selection of compensation mode at < 90°
9.2 TNRC in the program

When TNRC is selected, the control reads in two further blocks in advance during processing of the current block and calculates the intersection point of the compensated paths. The diagrams below show compensation for various transitions.

<table>
<thead>
<tr>
<th>&gt;180°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight line/straight line</td>
</tr>
<tr>
<td><img src="image1.png" alt="Diagram 1" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Diagram 3" /></td>
</tr>
</tbody>
</table>

TNRC for various transitions at > 180°
90°  180°

<table>
<thead>
<tr>
<th>Straight line/straight line</th>
<th>Straight line/arc</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arc/straight line</th>
<th>Arc/arc</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

TNRC for various transitions at 90°  180°
9 Tool Nose Radius Compensation (TNRC)

9.2 TNRC in the program

<90°

<table>
<thead>
<tr>
<th>Straight line/straight line</th>
<th>Straight line/arc</th>
</tr>
</thead>
</table>

![Diagram of straight line/straight line and straight line/arc with G42 and R](image1.png)

<table>
<thead>
<tr>
<th>Arc/straight line</th>
<th>Arc/arc</th>
</tr>
</thead>
</table>

![Diagram of arc/straight line and arc/arc with G42 and R](image2.png)

TNRC for various transitions at <90°
9.3 Cancellation of TNRC (G40)

Compensation mode is cancelled using preparatory function G40.

Compensation mode can only be cancelled in a program block when G function G00, G01, G10 or G11 is active. Tool number D0 corresponds to compensation value 0. It can also be used to cancel compensation.

Cancellation of compensation mode

<table>
<thead>
<tr>
<th>&gt;180°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight line</td>
</tr>
<tr>
<td>Arc</td>
</tr>
</tbody>
</table>

Cancellation of compensation mode at > 180°

<table>
<thead>
<tr>
<th>90° 180°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight line/straight line</td>
</tr>
<tr>
<td>Arc/straight line</td>
</tr>
</tbody>
</table>

Cancellation of compensation mode at 90° 180°
Cancellation of compensation mode at $<90^\circ$

Cancellation of compensation mode (special case)

Cancellation of TNRC in one block "Distance = 0"

N5  G91  Z100  L_F
N10  G40  Z0  L_F
N15  Z100  X+100  L_F

Contour error (hatched area).
9.4 Changing direction of compensation (G41, G42)

A perpendicular vector of length $R$ is created in the appropriate direction of compensation at the end position (A) of the block with the old G function (e.g. G42) and at the starting position (A) of the block with the new G function (e.g. G41).

9.5 Changing compensation number (G41 D.., G41 D..)

When changing the compensation number (G41 D.., G41 D..), the following applies:

No block start intersection is calculated with the old compensation; a vector perpendicular to the contour with length $R_1$ is created at the end position of the block with the old offset number D01; the block end intersection is calculated with the new compensation D02.
9.6 Changing compensation values (R1, R2)

The compensation values can be changed via the operator panel, via tape input, via the external tool offset or in the part program. The new compensation value takes effect in the next block (N20 in the example).

![Diagram showing compensation values](image)

9.7 Repetition of selected G function (G41, G42) with same compensation number

If a G41 or G42 function which has already been programmed is repeated, a vector of length \( R \) and perpendicular to the programmed path is created in the preceding block at the block end position.

The block start intersection \( S_1 \) is calculated for the following block:

\[
\begin{align*}
N5 & \quad G91 \quad D10 \quad G41 \quad X \ldots \quad Z \ldots \quad L_P \\
N10 & \quad X \ldots \quad L_P \\
N15 & \quad Z \ldots \quad L_P \\
N20 & \quad G41 \quad X \ldots \quad Z \ldots \quad L_P \\
N25 & \quad Z \ldots \quad L_P
\end{align*}
\]

**Error!** G41 repeated in N20. This causes a contour error.

![Diagram showing repeated selection](image)
9.8 M00, M01, M02 and M30 with TNRC selected

M00, M01

The NC stops at stop position S1, S2 and S3 for a single block (the positions are shown in the diagrams).

M00, M01 with TNRC selected

There are two different cases:

Case 1: 
N10 G01 F100 Z0 X0 LF
N20 G41 Z50 X50 LF
N25 H111 M00 LF
N30 X0 LF

H and M functions are executed at point S3.

Case 2: 
N10 G01 F100 Z0 X0 LF
N20 G41 Z50 X50 LF
N25 H111 M00 Q-5 LF
N30 X0 LF

H/M functions and Q movements are executed at point S1.

M02, M30

- The compensation is retracted one block ahead of the block in which it was cancelled using G40 and programmed with a least one axis address (N150 in this example).
  N150 X.. Z.. LF
  N200 G40 X.. M30 LF

- The compensation is not retracted if no path is programmed with G40 and if M30 is following:
  N150 X.. Z.. LF
  N200 G40 LF
  N250 M30 LF
Cancellation with G40:

\[
\begin{array}{c}
\text{N60} \ Z50 \ X50 \ L_F \\
\text{N70} \ Z50 \ X0 \ M30 \ L_F
\end{array}
\]

The compensation is retracted one block ahead of the block in which it was cancelled with G40 (N60 in this example).

No path in the case of G40 or programming of an axis in the last block:

\[
\begin{array}{c}
\text{N30} \ Z50 \ X50 \ L_F \\
\text{N35} \ X0 \ L_F \\
\text{N40} \ G40 \ L_F \\
\text{N45} \ M30 \ L_F
\end{array}
\]

The compensation is not retracted.

G40 in the last but one block or programming of two axes in one block:

\[
\begin{array}{c}
\text{N30} \ Z50 \ X50 \ L_F \\
\text{N35} \ G40 \ X0 \ L_F \\
\text{N40} \ M30 \ L_F
\end{array}
\]

The compensation is retracted in N35.
9.9 TNRC with combination of various block types and in conjunction with contour errors

When programming in compensation mode, special attention must be paid to the blocks without tool movements in order to prevent contour errors. There are three different types of block:

- Block with distance 0:
  Path addresses are programmed, but there is no movement since the distance is 0.
  \[ \text{N5} \ G91 \ Z0 \ L_F \]

- Block “without path”:
  Auxiliary functions or dwells are programmed in the compensation plane instead of path addresses.
  \[ \text{N5} \ M05 \ L_F \]
  \[ \text{N10} \ G04 \ X100 \ L_F \]

- Block “Not in compensation plane”:
  Axis addresses outside the compensation plane have been programmed.

Examples:

- One "auxiliary function block" between two motion blocks in the compensation plane.

![Diagram](image)

**TNRC: One "auxiliary function block" between two motion blocks**

\[ \text{N5} \ G91 \ Z100 \ L_F \]
\[ \text{N10} \ M08 \ L_F \]
\[ \text{N15} \ X-100 \ L_F \]

Block **N10** is executed at point \( S_{(N10)} \).
9.9 TNRC with combination of various block types and in conjunction with contour errors
• **Two** “auxiliary function blocks” between two motion blocks in the compensation plane.

![Diagram](image1)

**TNRC: Two “auxiliary function blocks” between two motion blocks**

```
N5  G91  Z100  LF
N10  M08  LF
N15  M09  LF
N20  X-100  LF
N25  Z100  LF
```

Blocks **N10** and **N15** are executed at point **S(N5)**.

**Contour error** (hatched area)

• **One block** for “movement = 0” between two motion blocks in the compensation plane.

![Diagram](image2)

**TNRC: One block for “movement 0”**

```
N5  G91  Z100  LF
N10  Z0  LF
N15  X-100  LF
```

**Contour error** (hatched range)
- **Two blocks** for “movement = 0” between two motion blocks in the compensation plane.

![Diagram of two blocks](image1)

**TNRC: Two blocks for “movement 0”**

```
N5  G91  Z100  L_F
N10  Z0  L_F
N15  Z0  L_F
N20  X-100  L_F
```

Contour error (hatched range)

- **One block** for “movement = 0” and one “auxiliary function block” between two motion blocks in the compensation plane.

![Diagram of one block](image2)

**TNRC: One block for “movement 0” and one “auxiliary function block”**

```
N5  G91  Z100  L_F
N10  Z0  L_F
N15  M08  L_F
N20  X-100  L_F
```

Block **N15** is executed at point **S(N10)**.

Contour error (hatched range)
- **One** “auxiliary function block” and **one block** for “movement = 0” between two motion blocks in the compensation plane.

![Diagram of Tool Nose Radius Compensation (TNRC)](image)

**TNRC: One “auxiliary function block” and one block for “movement 0”**

```
N5  G91  Z100  LF
N10 M08  LF
N15 Z0  LF
N20 X-100  LF
```

Block **N10** is executed at point \( S_{(N05)} \).

**Contour error** (hatched range)

- **One** block "Not within the compensation plane" between two motion blocks in the compensation plane.

![Diagram of Tool Nose Radius Compensation (TNRC)](image)

**TNRC: One block “Not within the compensation plane”**

```
N5  G91  G41  G01  G17  Z100  P100  D01  LF
N10 Z500  LF
N15 Q500  LF^1)
N20 X-500  LF
N25 Z1000 X-600  LF
```

1) Block not within compensation plane

**No contour violation**
- **Two** blocks "Not within the compensation plane" between two motion blocks in the compensation plane.

![Diagram](image)

**TNRC: Two blocks "Not within the compensation plane"**

<table>
<thead>
<tr>
<th>Line</th>
<th>Tool Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N5</td>
<td>G91 G41 G01 G17 Z100 F100 D01 LF</td>
<td></td>
</tr>
<tr>
<td>N10</td>
<td>Z500 LF</td>
<td>Block not within the compensation plane</td>
</tr>
<tr>
<td>N15</td>
<td>Q500 LF</td>
<td>Block not within the compensation plane</td>
</tr>
<tr>
<td>N20</td>
<td>Q-500 LF</td>
<td></td>
</tr>
<tr>
<td>N25</td>
<td>X-500 LF</td>
<td>Contour error (hatched range)</td>
</tr>
<tr>
<td>N30</td>
<td>Z1000 X-600 LF</td>
<td></td>
</tr>
</tbody>
</table>

### 9.10 Special cases of TNRC

The control always uses the **next block** to calculate the **point of intersection of the compensated paths**. If no axes in the compensation plane are programmed in the **next block**, the controller uses the **next block but one**. **Contour errors** may occur if the **intermediate block is smaller** than the selected compensation value.

Machining is not interrupted, but an **alarm is indicated**.

![Diagram](image)

**Special case of TNRC: Intermediate block < compensation value**
The direction of compensation for TNRC is retained and the traversing direction is reversed.
The return path in N10 must exceed twice the tool nose radius, or the tool will move in the wrong direction.
The following applies to external contours with circle transitions and obtuse angles:

In order to prevent a conditional stop in continuous path mode due to intermediate blocks which are too small, paths AB and BC can be omitted by the NC. In this case, the course of the path depends on tolerance d defined on start-up (max. 32000μm).
The block numbers are exchanged if the TNRC generates intermediate blocks (also in case of selection or cancellation) and if an axis motion outside the compensation plane has been programmed in between.

The control generates three intermediate blocks between points S1-S2-S3-S4; only when the cutter (S1) is lifted, they are traversed.

```
N5  G00  Q100  L_F
N10  Z0  X10  L_F
N15  G41  D01  Z20  X20  L_F
N20  G03  Z0  X40  K-20  I0  L_F
N25  Z0  X40  K0-  I-40  L_F
N30  G01  Q50  L_F
N35  G40  Z80  X60  L_F
N40  M30  L_F
```

The points S1, S2, S3, S4 logically belong to block N25. The processing sequence (visible in the single block is:

..., N20, N25 (S1), N30 (removal of the tool from the workpiece; retraction), N25 (S2), N25 (S3), N25 (S4), N35... . This is also true if N25 is a linear block.
9.11 Effect with negative compensation values

If the compensation value is negative (tool nose radius P4), a compensated path corresponding to G42 with a positive compensation value is implemented with G41, i.e. an analog internal contour is followed instead of the programmed external contour, and vice versa.

For the cutter centre path (→←→) shown in the diagram (a) below a positive compensation value has been entered.
A negative compensation value in conjunction with the same machining program will effect the machining shown in diagram (b).

If the program is created with a positive compensation value according to diagram (b), then entering a negative compensation value will result in machining according to diagram (a). The two operations are distinguished by a positive or negative compensation value.
10 Cycles

The following machining cycles are available in the form of permanently stored subroutines in the User Memory Submodule (UMS):

- L91/92 Retract cycles for tool change
- L93 Groove cycle
- L95 Stock removal cycle (paraxial roughing)
- L97 Thread cutting cycle
- L98 Deep-hole drilling cycle
- L99 Chaining of threads (four point thread cutting cycle)

These cycles are described in a separate publication "Programming Guide, SINUMERIK System 800, Cycles".
11 Programming of Cycles

11.1 General

Using the @ codes, cycles etc. can be generated directly on the control.

Difficult problems are solved with the WS 800 programming workstation and CL800 programming language.

The sections below are designed to provide you with an introduction to programming with the @ code.

11.2 Destination code

The destination code (@ code) consists of three digits, which can be interpreted as follows:

- @1 2 3 Three-digit @ code
  - Last digit to determine specific function
  - Centre digit to split main groups into subgroups
  - First digit to differentiate between main groups

11.2.1 Main groups

The @ code is broken down into the following eight groups:

- @0 . . General statements for program structure
- @1 . . Program branchings
- @2 . . Data transfer, general
- @3 . . Data transfer: System memory to R parameters
- @4 . . Data transfer: R parameters to system memory
- @6 . . Mathematical and logic functions
- @7 . . NC-specific functions
- @a . . I/O functions
11.2.2 Operands after the destination code

The @ code requires more detailed information (operands) for its function. These operands are defined by means of the following letters:

- **K** . . . Constant
- **R** . . . R parameter (register)
- **P** . . . Pointer

The value defined by constant K is fixed in the program and cannot be changed (direct value specification).

The value contained in a R parameter can be changed by the program (indirect value specification).

The pointer refers to a parameter in which the address of the parameter is contained and to the contents of which the function is to be applied (indirect value specification).

Examples for @ code with operands:

a) @201 R13 P37 L F
   - Pointer, with address specification of the source register
   - Destination register (parameter)
   - @ code for the function "data transfer, general"

Description of function in example a):
Load the contents of the source register whose address is in register R37 into the destination register R13.

b) @201 P16 P37 L F
   - Pointer, with address specification of the source register
   - Pointer, with address specification of the destination register
   - @ code for the function "data transfer, general"

Description of function in example b):
Load the contents of the source register whose address is in register R37 into the destination register whose address is to be found in register R16.

11.2.3 Notation

The @ code requires a strict notation. In the list of the individual commands below, the three digit @ code is followed by a series of notations each in angle brackets. The individual notations have the following significance:

- **<Const>** Direct value specification (constant K)
- **<R-Par>** Indirect value specification (R parameter)
- **<Var>** Indirect value specification (R parameter or pointer)
- **<Value>** Mixed value specification (constant, R parameter or pointer)
11.3 General statements for program structure

The main group 0 is broken down as follows:

@ 0 x y

Three-digit @ code

0: Save several R parameters
1: Save parameter range
2: Retrieve several R parameters
3: Retrieve R parameter range
4: Save R parameters
0: Main group 0

Main group 0/subgroup 0: Program statements

@00f (Enable for softkey start)
If this is the only command programmed in the first block of a subroutine, this means that the subroutine is enabled for start via softkey. The command should be at the beginning of the program to save time for searching. Refer to the machine tool manufacturer for information on where the softkey is configured.

Main group 0 / subgroup 4: Save R parameters

@040 <Const> <R par 1> . . . <R par n> Saving the R parameters onto stack
The constant <Const> indicates the number of the subsequent R parameters belonging to this function. The contents of the R parameters are saved in a stack register from R300 onwards.

@041 <R par 1> . . . <R par 2> Saving a group of R parameters onto stack
The contents of the R parameters from <R par 1> to <R par 2> are saved by transferring to the stack register.

@042 <Const> <R par n> . . . <R par 1> Fetch R parameters from stack
This command takes the saved values from the stack register and loads them in the stated R parameters. The R parameters must be listed in reverse sequence as compared with @040.

@043 <R par 1> . . . <R par 2> Fetch a group of R parameters from stack
The values saved with @041 are reloaded into the R parameters.

These commands of the main group 0 / subgroup 4 are used when working in a subroutine with R parameters which may already have been used at a higher level.
A push command (@040 or @041) is written at the start of the subroutine, being used to save the values and to occupy the specified R parameters with the value 0.

At the end of the subroutine a pop command (@042 or @043) is used to restore the original status.

Example of statement formats in the program:

```
L100 LF  \hspace{1cm} \textit{Call subroutine}
@041 R61 R69 LF  \hspace{1cm} \textit{The contents of the R parameters from R61 to R69 are transferred to the stack register and have the default value "0".}
\ldots
@043 R61 R69 LF  \hspace{1cm} \textit{The saved values are reloaded into parameters R61 to R69.}
\ldots
M17 LF  \hspace{1cm} \textit{End of subroutine}
```

### 11.4 Program branchings

Main group 1 is broken down as follows:

<table>
<thead>
<tr>
<th>@</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>y</td>
</tr>
</tbody>
</table>

Three-digit @ code

- 0: No relational operator
- 1: = equal to
- 2: <> not equal to
- 3: > greater than
- 4: >= greater than or equal to
- 5: < less than
- 6: <= less than or equal to

- 0: Absolute jump
- 1: CASE branch (relational operator y=1)
- 2: IF-THEN-ELSE branch
- 3: WHILE loop
- 4: REPEAT loop
- 5: FOR-TO loop
- 6: FOR-DOWNTO loop

11-4 © Siemens AG 1990 All Rights Reserved

Main group 1 / subgroup 0: Absolute jump

@100 <Const> or @100 <R par> \hspace{1cm} \textit{Absolute jump}

The constant or R parameter indicates the jump destination (block number) and jump direction. A positive block number means that the block to which the jump is to be performed is located towards the end of the program, while a negative block number means that it is near the start of the program.
If the sign indicates the wrong direction, the control does not find the block even if it is contained in the program (Alarm 3012: “block not in memory”).

**Examples:**
- @100 K375 L_F Absolute jump to block N375 towards end of program
- @100 K-150 L_F Absolute jump to block N150 towards beginning of program

---

**Main group 1 / subgroup 1: CASE branch**

@111 <Var> <Value 1> <Const 1> CASE branch
- <Value 2> <Const 2>
- ...
- <Value n> <Const n>

The notation <Var> is constantly compared with the successive notations <Value...>. If there is agreement, the program branches to the block number programmed under the notation <Const...>. If the comparison does not agree, the program continues with the next block at which the block number is programmed for program continuation (e.g. by means of an absolute jump with @100).

**Example for CASE branch:**

```
  N475 @111 R11 K1 K480  R11=1
K2 K485  R11=2
K3 K490 L_F  R11=3
@100 K900 L_F  R11 is neither 1, 2 nor 3
N480 .......... L_F
  @100 K495 L_F
N485 .......... L_F
  @100 K495 L_F
N490 .......... L_F
  ...

Jump to block N900 (end of program)
```

**Explanation regarding the above example:**
The diagram shows how a CASE branch is used in a cycle with axis switching. Depending on whether the programmer has determined the machining plane by parameterizing R11 = 1, 2 or 3, the program branches to the N480 (R11 = 1), N485 (R11 = 2) or N490 (R11 = 3) blocks. If neither of the three values is stored in R11, this is obviously the result of a parameterizing error. The program branches to block N900 (= end of program).
Main group 1 / subgroup 2: IF-THEN-ELSE branch

An IF-THEN-ELSE branch specifies:

If (IF) the condition (third digit in the @ code) has been satisfied, THEN the instructions given in the next blocks must be carried out otherwise (ELSE) branch to this block whose number is named by the last constant.

Here too, the sign of the block number acts as search direction.

@121 <Var> <Value> <Const> Equal to
If (IF) the numerical value defined with notation <Var> is equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

Example:
@121 R13 R27 K375 L F
Continuation of program if R13 = R27, else conditional jump to block N375 in direction of program end.

@122 <Var> <Value> <Const> Not equal to
If (IF) the numerical value defined with notation <Var> is not equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

@123 <Var> <Value> <Const> “>”
If (IF) the numerical value defined with notation <Var> is greater than the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

Example:
@123 R13 R27 K-150 L F
Continuation of program if R13 > R27, else conditional jump to block N150 in direction of program start.

@124 <Var> <Value> <Const> “>=”
If (IF) the numerical value defined with notation <Var> is greater than or equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

@125 <Var> <Value> <Const> “<”
If (IF) the numerical value defined with notation <Var> is less than the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.

@126 <Var> <Value> <Const> “<=”
If (IF) the numerical value defined with notation <Var> is less than or equal to the value defined with <Value>, then (THEN) the program is continued with the next block. Otherwise (ELSE) a jump is made to the block determined with the constant.
**Example:** Program run with IF-THEN-ELSE branches

![Diagram](image)

Condition fulfilled:
- **y** = yes
- **n** = no

**Part I**
- R58 = 123
- R77 = 230
- R88 = 7
- R95 = 12.45
- K230

**Part II**
- R85 = R88 / R77
- K250
- @100

**Part III**
- R58 = 7
- R77 = 12.45
- K480
- R2 = 2
- R7 = 7

**Explanation regarding the above example:**

The program section from a cycle shown in the example illustrates how IF-THEN-ELSE branches can be used to create program branchings. If, in block N225, the contents of register R58 are greater than the contents of register 77, then the instructions in the next line are executed.

Register R88 is loaded with 7 and register R95 with 12.45. If, however, R58 is less than or equal to R77, then the program branches to block N230. But in the line preceding block N230 there is an absolute jump to block N250.

This way the IF-THEN-ELSE branch in block N225 causes either block N225 or the program section from block N230 to N250 to be processed.

The second example does not include the absolute jump, so that the instructions in block N475 are either executed or not. This part of the program can also be skipped with the IF-THEN-ELSE branch @124.
Main group 1 / subgroup 3: WHILE loop

@13y <Var> <Value> <Const>
The WHILE loop is a repeat statement with sampling of the repeat conditions at the start of the loop. The relation operators correspond to those of the IF-THEN-ELSE branch. Provided the comparison is fulfilled, the next block is processed. At the end of the block an absolute jump must be programmed with @100 <Const> which leads back to sampling.
If the comparison is not fulfilled, a jump is made to the block defined under <Const> and which is generally located after the block with the absolute jump.
The values assigned to y shall be in accordance with commands @ 121 ...@ 126.

Examples:

N300 @133 R13 R27 K375 LF       Continuation of loop - as long as loop condition  
                              R13=R27 is fulfilled.

@100 K–300 LF  
N375 . LF  
N300 @133 R13 R27 K375 LF       Continuation of loop - as long as loop condition  
                              R13>R27 is fulfilled.

@100 K–300 LF  
N375 . LF  

Main group 1 / subgroup 4: REPEAT loop

@14y <Var> <Value> <Const>
The REPEAT loop is a repeat statement with sampling of the repeat conditions at the start of the loop. The relation operators correspond to those of the IF-THEN-ELSE branch. If the comparison is not fulfilled, a jump is made back to the block defined under <Const>. If the condition is fulfilled, the loop is exited and the program is resumed.
The values assigned to y shall be in accordance with commands @ 121 ...@ 126.

Examples:

N400 . LF  
                              Repeat the following statements until condition  
                              R13=R27 has been fulfilled.

@141 R13 R27 K–400 LF  
N400 . LF  
                              Repeat the following statements until condition  
                              R13>R27 has been fulfilled.

@143 R13 R27 K–400 LF  

Main group 1/ subgroup 5: FOR TO loop

@151 <Var> <Value> <Const>
The FOR TO loop is a counting loop in which the contents of the R parameter defined under <Var> are incremented with each pass. Sampling for “equal to” is performed at the beginning of the loop. In the event of inequality, the loop is processed, otherwise a jump is made to the block defined under <Const>. At the end of the loop the variable <Var> must be incremented (@620) with an absolute jump back to the start of the loop.

Example:

R50=1 R51=5 R52=10 L_F  
@201 R50 R51 L_F  
N500 @151 R50 R52 K505 L_F  

Main group 1/subgroup 6: FOR DOWNTO loop

@161 <Var> <Value> <Const>
The FOR DOWNTO loop is a counting loop in which the contents of the R parameter defined under <Var> are decremented with each pass. Sampling for “equal to” is performed at the beginning of the loop. In the event of inequality, the loop is processed, otherwise a jump is made to the block defined under <Const>.

At the end of the loop the variable <Var> must be decremented (@621) with an absolute jump back to the start of the loop.

Example:

R50=10 R51=5 R52=1 L_F  
@201 R50 R51 L_F  
N600 @161 R50 R52 K605 L_F  

Assignment of value for R5, R51, R52
Transfer of data from R5 to R50
Start of FOR TO loop

Assignment of value for R5, R51, R52
Transfer of data from R5 to R50
Start of FOR DOWNTO loop
11.5 Data transfer, general

Main group 2 is broken down as follows:

<table>
<thead>
<tr>
<th>@2</th>
<th>x</th>
<th>y</th>
<th></th>
<th></th>
<th>Three-digit @ code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>0:</td>
<td>Delete variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1:</td>
<td>Load variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2:</td>
<td>Exchange variable contents</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3:</td>
<td>Read bits from bit pattern</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0:</td>
<td>Data transfer R parameter/R parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1:</td>
<td>Data transfer R parameter/MIB (input buffer memory) for numeric variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2:</td>
<td>Main group 2</td>
<td></td>
</tr>
</tbody>
</table>

Main group 2/subgroup 0: Data transfer R parameter/R parameter

@200 <Var> Delete variable
The value of the R parameter defined with the notation <Var> is deleted.

@201 <Var> <Value> Load variable with value
The numerical value defined under <Value> is transferred to the R parameter defined under <Var>.

@202 <Var 1> <Var 2> Swap contents of variables
The contents of the two R parameters defined under <Var 1> and <Var 2> are swapped.

@203 <Var 1> <Var 2> <Const> Read bit from bit pattern
The status of a bit of the pattern variable <Var 2> is transferred to <Var 1>. The bit number is specified in <Const.>.

Main group 2/subgroup 1: Data transfer R parameter/input buffer memory (MIB) for numeric variable

@210 <Value 3> <Value 4> Delete input buffer
The input buffer (MIB) is deleted.
Significance:
<Value 3> Start address
<Value 4> End address

@211 <Var> <Value 1> Read input buffer
The R parameter <Var> is loaded with the contents of the MIB (machine input buffer) cell <Value 1>.

Example:
@211 R50 K101 Lp The contents of the MIB (machine input buffer) cell 101 are read in R50.
@212 <Value 1 > <Value> Write machine input buffer
The MIB cell <Value 1> is loaded with the numerical variable <Value>.

Example:
@212 K102 K5 LF
The value 5 is written into the MIB (machine input buffer) cell 102.

11.6 Data transfer, system memory to R parameters
Main group 3 is broken down as follows:

@ 3 x y Three-digit @ code

Special function

0: Machine data
1: Setting data
2: Tool offsets
3: Zero offsets
4: Programmed setpoints
5: External setpoints (available soon)
6: Actual values
7: Program data
e: System cells

} transfer to R parameters

} read to R parameters

All @ commands in this main group feature <Var> as their first notation. Consequently, an R parameter is defined, directly or via a pointer, to which the contents of the system memory cell addressed are to be transferred.

Main group 3/subgroup 0: Transfer machine data to R parameters

@300 <Var> <Value 1> Machine data NC
The address of an NC machine data is defined under <Value 1>.
Address range: 0 to 4999.

Example:
@300 R50 K2240 LF
In parameter R50, the value of the 1st software limit switch is in plus direction for the first axis.

@301 <Var> <Value 1> Machine data NC byte
The byte address of an NC machine data is defined under <Value 1>.
Address range: 5000 to 6999.
@302 <Var> <Value 1> <Value 2> Machine data NC bit
The byte address of an NC machine data bit is defined under <Value 1>. Address range: 5000 to 6999. The bit address (0 to 7) is under <Value 2>.

@306 <Var> <Value 1> Machine data PLC
The byte address of a PLC machine data is defined under <Value 1>. Address range: 0 to 1999.

@307 <Var> <Value 1> Machine data PLC byte
The byte address of a PLC machine data is defined under <Value 1>. Address range: 2000 to 3999.

@308 <Var> <Value 1> <Value 2> Machine data PLC bit
The byte address of a PLC machine data bit is defined under <Value 1>. Address range: 2000 to 3999. The bit address (0 to 7) is under <Value 2>.

Main group 3/subgroup 1: Transfer setting data to R parameters

@310 <Var> <Value 1> Setting data NC
The address of a setting data is defined under <Value 1>. Address range: 0 to 4999.

@311 <Var> <Value 1> Setting data NC byte
The byte address of a setting data is defined under <Value 1>. Address range: 5000 to 9999.

@312 <Var> <Value 1> <Value 2> Setting data NC bit
The byte address of a setting data bit is defined under <Value 1>. Address range: 5000 to 9999. The bit address (0 to 7) is under <Value 2>.

Main group 3/subgroup 2: Transfer tool offsets to R parameters

@320 <Var> <Value 1> <Value 2> <Value 3> Tool offset
With this command the single compensation values from the tool offset memory can be read to the parameter under the notation <Var>.

The notations <Value 1> to <Value 3> should be as follows:
<Value 1> TO range
Range: 0
<Value 2> Tool offset number (D number)
Range: 1 to 99
<Value 3> Number of tool offset memory (P number)
Range: 0 to 9

Example:
@320 R67 K0 K14 K2 Lp
Reading the offset value P2 (geometry, length) of the tool offset no. D14 for TO range 0 into parameter R67.
Main group 3/subgroup 3: Transfer zero offsets to R parameters

@330 <Var> <Value 1> <Value 2> <Value 3> Settable zero offset
The notations <Value 1> to <Value 3> should be as follows:
- <Value 1> Group of settable zero offsets (G54=1 to G57=4)
- <Value 2> Axis number
- <Value 3> Coarse or fine value (0 or 1)

Example:
@330 R81 K1 K2 K0 L F
The coarse value of the first settable zero offset (G54) of the second axis is read into parameter R81.

@331 <Var> <Value 1> <Value 2> Programmable zero offset
Significance:
- <Value 1> Group of programmable additive zero offsets (G58=1 to G59=2)
- <Value 2> Axis number

@332 <Var> <Value 2> External zero offset from PLC
<Value 2> defines the number of the axis of the external zero offset entered from the PLC.

@333 <Var> <Value 2> DRF offset
<Value 2> defines the number of the axis of the PRESET offset.

@334 <Var> <Value 2> Preset offset
<Value 2> defines the number of the axis of the DRF offset.

@336 <Var> <Value 2> Total offset
<Value 2> defines the number of the axis of the total offset. The total offset comprises:
- the selected settable zero offset
- the programmable additive zero offset
- the external zero offset
- the selected tool offset
PRESET and DRF offsets are not taken into account.

@337 <Var> <Value 1> <Value 2> <Value 3> Settable coordinate rotation
The angle of rotation of the settable coordinate rotation can be entered into the <Var> parameter.
The notations <Value 1> to <Value 3> should be as follows:
- <Value 1> Number of channel (0 = own channel)
- <Value 2> Group of settable coordinate rotations (G54=1 and G59=2)
- <Value 3> Number of angle (currently = 1)
@338 <Var> <Value 1> <Value 2> <Value 3> Programmable coordinate rotation
The angle of rotation of the programmable coordinate rotation can be entered into the <Var> parameter.

Significance:
<Value 1> Number of channel (1 = own channel)
<Value 2> Group of programmable coordinate rotations (G58=1 and G59=2)
<Value 3> Number of angle (currently = 1).

Main group 3/subgroup 4: Read programmed setpoints into R parameter

@342 <Var> <Value 1> <Value 3> Programmed spindle speed
The programmed spindle speed from one channel is read into the parameter <Var>. The notations shall be as follows:
<Value 1> Number of channel (0 = own channel)
<Value 3> Spindle no. (0 = number of leadscrew)

@345 <Var> <Value 1> <Value 2> Programmed cutting rate
By this instruction the cutting speed programmed under G96 may be transferred into an R parameter. The channel no. will be entered in <Value 1> (own channel = 0). <Value 2> will be "0".

Main group 3/subgroup 6: Read actual values to R parameters

Prior to instructions @360 to @36b, a separate block must be programmed with @714. To be able to read values from a channel other than your own, NC start must be given in that channel as well.

@360 <Var> <Value 2> Actual axis position, workpiece-related
<Value 2> defines the axis whose workpiece-related actual value is to be transferred to the R parameter.

Example:
@360 R54 K2 L_P
Read actual position value of the 2nd axis related to workpiece - zero and enter it into register R54.

@361 <Var> <Value 2> Actual axis position, machine-related
<Value 2> defines the axis whose machine-related actual value is to be transferred to the R parameter.
In simulation @360 and @361 provide the same values.

@363 <Var> <Value 2>  Actual spindle position
<Value 2> defines the spindle whose actual position is to be transferred to the R parameter.

@364 <Var> <Value 2>  Actual spindle speed
<Value 2> defines the spindle whose actual speed is to be transferred to the R parameter.

@367 <Var> <Value 1>  Axis number of the current plane/leadscrew number
<Var> defines the R parameter beginning with which the axis numbers of the plane (4 values) selected via G16 are entered. The number of the leadscrew is stored in the 5th R parameter. <Value 1> defines the channel no. (0 = own channel).

The following R parameters are loaded:
- Rn: Number of horizontal axis
- Rn+1: Number of vertical axis
- Rn+2: Number of the axis vertical to the plane
- Rn+3: Number of the axis in which length 2 is effective (tool type 30 on the M version)
- Rn+4: Number of leadscrew

If a tool length compensation in negative direction is programmed via G16 XYZ, the value 128 is added for the Z axis by @367 in order to be able to recognize the minus sign.

**Example:**
```
@367 R50 K1 LF
```
The data of the current plane and spindle number are read in channel 1 and stored from R50 on

@36a <Var> <Value 1>  Actual D function
This command is used to transfer the number of the selected tool offset (D number) to the R parameter. The channel (own channel = 0) must be entered under <Value 1>. "0" must be entered in the case of the SINUMERIK 810.

Comment: "a" in the @ code stands for hexadecimal "A" (=10).

@36b <Var> <Value 1> <Value 3>  Actual G function
The G function of the part program block being processed is read from the working memory into the parameter <Var>. <Value 1> defines the channel number. If zero is defaulted, the own channel is read. <Value 3> defines the internal G group to which the current G function belongs. Section 12 contains a table with the internal G group division (overview for @36b).

**Example:**
```
@36b R50 K0 K0 LF
```
In the own channel the actual G function of the first internal G group (Group 0) is read into parameter R50.

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
Main group 3/subgroup 7: Read program data to $R$ parameters

@371 <Var> <Value1> <Value3>

**Special bits**

This command is used to read out special bits for detecting different active signals.

**Channel-dependent bits:**
- Bit 0 = Block search active
- Bit 1 = Dry run feed active
- Bit 2 = Simulation active

The channel number (own channel = 0) should be entered in <Value1>. <Value3> contains the bit number.

**Channel-independent bits:**
- Bit 0 = Measuring input 1 active
- Bit 1 = Measuring input 2 active

99 must be entered in <Value1>. <Value3> contains the bit number.

**Example:**

@371 R81 K0 K1 L F

The state of the special bits for "dry run feed" in the own channel is read into parameter R81.

Main group 3/subgroup 8: Read PLC signal bits to $R$ parameters

@380 <Var> <Value1> <Value2> <Value3>

The state of an input bit in the PLC is read into the parameter defined with <Var>. The PLC number is defined by <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.

**Example:**

@380 R50 K1 K2 K0 L F

The state of the defined input bit is read into parameter R50.

@381 <Var> <Value1> <Value2> <Value3>

The state of an output bit in the PLC is read into the parameter defined with <Var>. The PLC number is defined by <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.

@382 <Var> <Value1> <Value2> <Value3>

The state of a flag bit in the PLC is read into the parameter defined with <Var>. The PLC number is defined by the <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 must be entered for the PLC number.
@383 <Var> <Value1> <Value2> <Value3> <Value4>
The state of a PLC data word bit is read into the parameter defined with <Var>. The PLC number is defined by <Value1>, the number of the DB or DX by <Value2>, the data word number by <Value3> and the bit address by <Value4>. K1 must be entered for the PLC number.

Example:
@383 R51 K1 K2 K4 K2
The state of the defined PLC data word bit is read into R51.

Note:
In the 3rd channel (simulation) @80 to @83 are skipped. This can cause the program run to be changed.

Main group 3 / subgroup e: Read system cells to R parameters

@3e4 <Var> <Value1> Read active gear stage
Depending on the spindle number <Value1> the active gear stage is read to the parameter <Var>. If the spindle number is 0, the number of the leadscrew is used as spindle number.

11.7 Data transfer, R parameters to system memory

Main group 4 is broken down as follows:

@ 4 x y Three-digit @ code

<table>
<thead>
<tr>
<th>Special function</th>
<th>transfer</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: R parameters to machine data</td>
<td>1: R parameters to setting data</td>
<td></td>
</tr>
<tr>
<td>2: R parameters in tool offsets</td>
<td>3: R parameters in zero offsets</td>
<td></td>
</tr>
<tr>
<td>4: R parameters in programmed setpoints</td>
<td>e: R parameters to system cells</td>
<td></td>
</tr>
<tr>
<td>4: Main group 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
All @ commands in this main group feature <Value> as their first notation. Consequently, the numerical value to be transferred is defined either directly using a constant or indirectly via an R parameter or a pointer.

### Main group 4/subgroup 0: Transfer R parameters to machine data

1. **@400 <Value 1> <Value>**
   - **Machine data NC**
   - The address of an NC machine data is defined under <Value 1>.
   - Address range: 0 to 4999.
   - **Example:**
     - @400 K2241 R90 Lp

2. **@401 <Value 1> <Value>**
   - **Machine data NC byte**
   - The byte address of an NC machine data bit is defined under <Value 1>.
   - Address range: 5000 to 6999.

3. **@402 <Value 1> <Value> <Value>**
   - **Machine data NC bit**
   - The byte address of an NC machine data bit is defined under <Value 1>.
   - Address range: 5000 to 6999. The bit address (0 to 7) is under <Value 2>.

4. **@406 <Value 1> <Value>**
   - **Machine data PLC**
   - The address of a PLC machine data is defined under <Value 1>.
   - Address range: 0 to 1999.

5. **@407 <Value 1> <Value>**
   - **Machine data PLC byte**
   - The byte address of a PLC machine data is defined under <Value 1>.
   - Address range: 2000 to 3999.

6. **@408 <Value 1> <Value> <Value>**
   - **Machine data PLC bit**
   - The byte address of a PLC machine data bit is defined under <Value 1>.
   - Address range: 2000 to 3999. The bit address (0 to 7) is under <Value 2>.

### Main group 4/subgroup 1: Transfer R parameters to setting data

1. **@410 <Value 1> <Value>**
   - **Setting data NC**
   - The address of a setting data is defined under <Value 1>.
   - Address range: 0 to 4999.

2. **@411 <Value 1> <Value>**
   - **Setting data NC byte**
   - The byte address of a setting data is defined under <Value 1>.
   - Address range: 5000 to 9999.

3. **@412 <Value 1> <Value> <Value>**
   - **Setting data NC bit**
   - The byte address of a setting data bit is defined under <Value 1>.
   - Address range: 5000 to 9999. The bit address (0 to 7) is under <Value 2>.
Main group 4/subgroup 2: Write R parameters in tool offsets

@420 <Value 1> <Value 2> <Value 3> <Value> Tool offset
   The numerical value is entered in the tool offset memory. The contents of the memory are overwritten.

   The notations <Value 1> to <Value 3> should be as follows:
   <Value 1>  TO range
                Range: 0
   <Value 2>  Tool offset number (D number)
                Range: 1 to 99
   <Value 3>  Number of tool offset value (P number)
                Range: 0 to 9.

   Example:
   @420 K0 K2 K3 R80 Lf  In the TO range 0, offset memory D2 under tool offset value P3 is loaded with the contents of R80.

@423 <Value 1> <Value 2> <Value 3> <Value> Tool offset, additive
   The numerical value is added to the value in the tool offset memory.
   The notations <Value 1> to <Value 3> should be as follows:
   <Value 1>  TO range; range: 0
   <Value 2>  Tool offset number (D number)
                Range: 1 to 99
   <Value 3>  Number of tool offset value (P number)
                Range: 0 to 9.

Main group 4/subgroup 3: Write R parameters in zero offsets

@430 <Value 1> <Value 2> <Value 3> <Value> Settable zero offset
   The numerical value is entered in the zero offset memory. The contents of the memory are overwritten.

   The notations <Value 1> to <Value 3> should be as follows:
   <Value 1>  Group of settable zero offsets (G54 = 1 to G 57 = 4)
   <Value 2>  Axis number
   <Value 3>  Coarse or fine value (0 or 1)

   Example:
   @430 K1 K2 K0 K500 Lg  The coarse value of the first settable zero offset is loaded with the constant 500 for the second axis.
@431 <Value 1> <Value 2> <Value 3> <Value > Settable zero offset, additive
The numerical value is added to the value in the zero offset memory.

The notations <Value 1> to <Value 3> should be as follows:
- <Value 1> Group of settable zero offsets (G54=1 to G57=4)
- <Value 2> Axis number
- <Value 3> Coarse or fine value (0 or 1).

@432 <Value 1> <Value 2> <Value> Programmable zero offset
Significance:
- <Value 1> Group of programmable additive zero offsets (G58=1 to G59=2)
- <Value 2> Axis number.

@434 <Value 2> <Value> DRF offset
<Value 2> defines the number of the axis of the DRF offset.

@435 <Value 2> <Value> PRESET offset
<Value 2> defines the number of the axis of the PRESET offset. If the @435 is used to describe the PRESET offset in the AUTOMATIC or MDA mode, the offset is only active after M2/M30/PRESET. If “TRANSMIT” function is active, command @435 also applies to fictitious axes.

@435 is chiefly used to deliberately reset the PRESET offset at the end of program.

@437 <Value 1> <Value 2> <Value 3> <Value> Settable coordinate rotation
Write coordinate rotation.
The notations <Value 1> to <Value 3> should be as follows:
- <Value 1> Number of channel (0 = own channel)
- <Value 2> Group of settable coordinate rotations (G54=1 and G57=4)
- <Value 3> Number of angle (currently =1)

@438 <Value 1> <Value 2> <Value 3> <Value> Settable coordinate rotation, additive
Write additive, settable coordinate rotation (G5 to G57).
Significance:
- <Value 1> Number of channel (0 = own channel)
- <Value 2> Groups 1 to 4 (G54 to G57)
- <Value 3> Number of angle (currently =1).
@439 <Value 1> <Value 2> <Value 3> <Value>  Programmable coordinate rotation

  Write programmable coordinate rotation.
  Significance:
  <Value 1> Number of channel (0 = own channel)
  <Value 2> Group of programmable additive coordinate rotations
    (G58=1 and G59=2)
  <Value 3> Number of angle (currently = 1).

@43a <Value 1> <Value 2> <Value 3> <Value>  Programmable coordinate rotation, additive

  Write additive programmable coordinate rotation (G58 and G59).
  Significance:
  <Value 1> Number of channel (0 = own channel)
  <Value 2> Group 1 to 2 (G58 and G59)
  <Value 3> Number of angle (currently = 1).

Main group 4/subgroup 4:  Write R parameters in programmed setpoints

@440 <Value 3> <Value>  Programmed axis position

  This command makes it possible to program axes independently of axis names.
  The number of the axis to be traversed is entered under <Value 1> and the position to be approached or the distance to be traversed is entered under <Value>.
  If function “Axis duplication” is active, the value of the programmed axis position is assigned to both the leading and the following axis.

  Example:
  @440 K2 K100  Value 100 is assigned to the traversing path of the second axis via a constant.

@442 <Value 3> <Value>  Programmed spindle speed

  This command makes it possible to program the spindle speed.
  The number of the spindle is entered under <Value 3> and the spindle speed under <Value>.

@446 <Value>  Programmed radius

  This command makes it possible to program the radius independently of the address specified in the machine datum. The numerical value is defined under <Value>.

@447 <Value>  Programmed angle

  This command makes it possible to program the angle independently of the address specified in the machine datum. The numerical value is defined under <Value>.
@482 <Value1><Value2><Value3><Value>

The state of a flag bit in the PLC is loaded via a parameter, a pointer or a constant.

The PLC number is defined by <Value1>, the byte address by <Value2> and the bit address by <Value3>. K1 should be entered for the PLC number.

Example:
@482 K1 K2 K0 K0

The state of the defined PLC flag bit is loaded with 0

@483 <Value1><Value2><Value3><Value4><Value>

The state of a data word bit in the PLC is loaded via a parameter, a pointer or a constant.

The PLC number is defined by <Value1>, the DB number by <Value2>, the data word number by <Value3> and the bit address by <Value4>. K1 must be entered for the PLC number.

Example:
R60=1 R62=2 R63=4
@483 R60 R62 R63 K2 K1

The state of the defined data word bit is loaded with 1

Note:
In the 3rd channel (simulation) @480 to @483 are overread. This can cause the program run to be changed.

Main group 4/subgroup e: Write R parameters to system cells

@4e1 <Value 1> <Value 2> <Value>

Program spindle acceleration time constant

This command makes it possible to program the spindle acceleration time constant. The spindle acceleration time constant indicated under <Value> does not overwrite the values entered in the machine data but an internal data location. The machine data become active again after cancellation of thread cutting (e. g. G00...), after a change in machine data, after RESET and after POWER ON. The spindle number is defined by <Value 1> and the gear stage by <Value 2>.
11.8 Mathematical functions

Main group 6 is broken down as follows

@ 6  x  y

Three-digit @ code

- Special function
  - 0: Value assignments with arithmetic operations
  - 1: Arithmetic functions
  - 2: Arithmetic procedures
  - 3: Trigonometric functions
  - 4: Logarithmic functions
  - 5: Logic functions
  - 6: Logic procedures
  - 7: Boolean assignments

6: Main group 6

Main group 6/subgroup 0: Value assignments with arithmetic operations

In this subgroup @ is not required. An incremental calculation with several notations on the right-hand side of the equation is allowed.

<Var> = <Value 1> + <Value 2> Addition
<Var> = <Value 1> - <Value 2> Subtraction
<Var> = <Value 1> * <Value 2> Multiplication
<Var> = <Value 1> / <Value 2> Division

Main group 6/subgroup 1: Arithmetic functions

@610 <Var> <Value> Absolute value generation
The absolute value part of the numerical value defined under <Value> is stored in <Var>.

Example:
R12=-34 L_F
@610 R76 R12 L_F The R parameter R76 contains the absolute value (= 34) from R12.

@613 <Var> <Value> Square root
The square root is formed from the numerical value defined under <Value> and is stored after <Var>.

Example:
@613 R13 K64 L_F The square root is formed from the constant (= 64) and the result (= 8) is entered/stored in R13.
11.90 Mathematical functions

@614 <Var> <Value 1> <Value 2> Root from sum of squares
The sum of the squares is formed from the numerical value defined under <Value 1> and <Value 2> and the square root of this is stored in <Var>.

Example:
R25=15 R26=20 L F
@614 R77 R25 R26 L F
The square root is formed from the sum of the squares of the R parameters R25 (=225) and R26 (=400). The result (= 25) is entered in R77.

Main group 6/Subgroup 2: Arithmetic procedures

@620 <Var> Incrementing
The contents of the R parameter defined under <Var> are incremented.

Example:
R70=1 L F
@620 R70 L F
The content of parameter R70 is incremented; the new content is 2.

@621 <Var> Decrementing
The contents of the R parameter defined under <Var> are decremented.

Example:
R70=1 L F
@621 R70 L F
The content of parameter R70 is decremented; the new content is 0.

@622 <Var> Integer component
The integer component is formed from the numerical value defined by means of an R parameter or a pointer. The result is then in the same R parameter or pointer.

Example:
R60=2.9 L F
@622 R60 L F
The integer content of R60 is formed; the new content is 2.
Main group 6/Subgroup 3: Trigonometric functions

@630 <Var> <Value>
Sine
The sine is formed from the angle value defined under <Value> and is stored in <Var>.

Example:
R27=30 L_F
@630 R15 R27 L_F
The sine of the content of R27 (= 0.5) is entered/stored in R15.

@631 <Var> <Value>
Cosine
The cosine is formed from the angle value defined under <Value> and is stored in <Var>.

@632 <Var> <Value>
Tangent
The tangent is formed from the angle value defined under <Value> and is stored in <Var>.

@634 <Var> <Value>
Arc sine
The arc sine is formed from the numerical value defined under <Value> and is stored as an angle value in <Var>.

Example:
R35=0.70710678 L_F
@634 R17 R35 L_F
The arc sine is formed from the content of R35 and the result (= 45) is entered/stored in R17.

@637 <Var> <Value 1> <Value 2>
Angle between two vector components
The numerical values defined under <Value 1> and <Value 2> are viewed as vectors. The result is the angle between the component under <Value 2> and the sum vector.

Example:
R35=20 R36=30 L_F
@637 R17 R35 R36 L_F
The angle is formed from the vector components (contents of R parameters R35 and R36) and the result (= 146.30993) is entered/stored in R17.
Main group 6/subgroup 4: Logarithmic functions

@640 <Var> <Value> Natural logarithm
The natural logarithm is formed from the numerical value defined under <Value> and is stored in <Var>.

Example:
@640 R80 K10 Lf
The natural logarithm is formed from the constant 10. The result (=2.3025846) is entered/stored in R80.

@641 <Var> <Value> Exponential function
The exponential function \( e^x \) is formed from the numerical value defined under <Value> and is stored in <Var>.

Example:
@641 R80 K2.5 Lf
The exponential function is formed for the exponent defined by the constant. The result (=12.182496) is entered in R80.

Main group 6/Subgroup 5: Logical functions

@650 <Var> <Var 1> <Value> OR
The bit patterns under <Var 1> and <Value> undergo a logical OR operation. The result is stored in <Var>.

Example:
R50=00101100 Lf
R51=10110011 Lf
@650 R52 R50 R51 Lf
The pattern variables R50 and R51 undergo a logical OR operation and the result is stored in R52.
The content of R52 is 10111111.
@651 <Var> <Var 1> <Value>  EXCLUSIVE OR
The bit patterns under <Var 1> and <Value> undergo a logical exclusive OR operation. The result is stored in <Var>.

@652 <Var> <Var 1> <Value>  AND
The bit patterns under <Var 1> and <Value> undergo a logical AND operation. The result is stored in <Var>.

@653 <Var> <Var 1> <Value>  NAND
The bit patterns under <Var 1> and <Value> undergo a logical AND operation. The result is negated and stored in <Var>.

@654 <Var> <Value>  NOT
The bit pattern under <Value> is logically negated. The result is stored in <Var>.

Example:
R50=00101100 LF
@654 R52 R50 LF
The contents of the pattern variable R50 is negated and the result is stored in R52. The content of R52 is 11010011.

@655 <Var> <Var 1> <Value>  OR bit
The bits under <Var 1> and <Value> undergo a logical OR operation. The result is stored in <Var>.

@656 <Var> <Var 1> <Value>  EXCLUSIVE OR bit
The bits under <Var 1> and <Value> undergo a logical OR operation. The result is stored in <Var>.

@657 <Var> <Var 1> <Value>  AND bit
The bits under <Var 1> and <Value> undergo a logical AND operation. The result is stored in <Var>.

Example:
R50=1 LF
R51=0 LF
@657 R52 R50 R51 LF
The Boolean variables R50 and R51 undergo a logical AND operation and the result is stored in R52. The content of R52 is 0.

@658 <Var> <Var 1> <Value>  NAND bit
The bits under <Var 1> and <Value> undergo a logical AND operation. The result is negated and stored after <Var>.

Example:
R50=1 LF
R51=0 LF
@658 R52 R50 R51 LF
The Boolean variables R50 and R51 undergo a logical AND operation. The result is negated and stored in R52. The content of R52 is 1.

@659 <Var> <Value>  NOT bit
The bit under <Value> is logically negated. The result is stored in <Var>.
Main group 6/subgroup 6: Logic procedures

@660 <Var> <Const> Delete bit in PATTERN
The constant <Const> is used to define a bit (0 to 7) which is to be deleted in the bit pattern specified by means of <Var>.

Example:
\[ R60=01100111 L_F \]
\[ @660 R60 K6 L_F \]
Bit no. 6 of the pattern variable is deleted. The content of R60 is 00100111.

@661 <Var> <Const> Set bit
The constant <Const> is used to define a bit (0 to 7) which is to be set to “1” in the bit pattern specified by means of <Var>.

Example:
\[ R70=00000000 L_F \]
\[ @661 R70 K2 L_F \]
Bit no. 2 of the pattern variable is set. The content of R70 is 00000100.

Main group 6/subgroup 7: Boolean assignments

@671 <Var 1> <Var 2> <Value> Equal to
If the numerical values defined under <Var 2> and <Value> are equal, the Boolean variable <Var 1> is set to “1”.

Example:
\[ R50=11001100 L_F \]
\[ @671 R51 R50 K11001100 L_F \]
As R50 equals the bit pattern of constant K, R51 is set to “1”.

@672 <Var 1> <Var 2> <Value> Not equal to
If the numerical values defined under <Var 2> and <Value> are not equal, the Boolean variable <Var 1> is set to “1”.

@673 <Var 1> <Var 2> <Value> “>”
If the numerical value defined under <Var 2> is greater than the value under <Value>, the Boolean variable <Var 1> is set to “1”.

@674 <Var 1> <Var 2> <Value> “>=”
If the numerical value defined under <Var 2> is greater than or equal to the value under <Value>, the Boolean variable <Var 1> is set to “1”.

@675 <Var 1> <Var 2> <Value> “<”
If the numerical value defined under <Var 2> is less than the value under <Value>, the Boolean variable <Var 1> is set to “1”.

@676 <Var 1> <Var 2> <Value> “<=”
If the numerical value defined under <Var 2> is less than or equal to the value under <Value>, the Boolean variable <Var 1> is set to “1”.
11.9 NC-specific functions

Main group 7 is broken down as follows:

@  7  x  y  Three-digit @ code

Special functions

- 0: Modification to program and machine reference points
- 1: Individual functions
- 2: Measurement functions

7: Main group 7

---

Main group 7/subgroup 0: Modification to program and machine reference points

@706  Position in the block in relation to the actual-value system of the machine

With command @706, a position is specified with reference to machine zero and approached by the specified axes. The command acts only block by block.

As many axes can be specified as the NC can traverse simultaneously. The positions to be approached by the axes are either programmed in DIN code or by means of the command @440 ...

The command @706 suppresses all zero offsets (settable, settable additive, programmable and external) as well as the PRESETS and DRF offsets.

In order to approach a position in the machine actual value system, the tool offsets must also be cancelled.

If machine data 5007.1 is set, G53 acts in the same way as @706.

If this machine data is not set, the PRESETS and DRF offsets are not suppressed with G53.

Example:

@706 X1000 Z500 L F   The programmed traversing paths in X and Z are approached in relation to machine zero.
Main group 7/subgroup 1 : Individual functions

@710 <Var 1> <Var 2>  
Reference processing
This command is required for reference processing in the turning cycle L 95 "stock removal". It is used to split up a contour programmed in a subroutine into single blocks. The data are stored in R parameters.

The contour element is stored in a total of eight R parameters, beginning with the R parameter <Var1> (Rn). Reference processing requires a total of four R parameters as input data.

The first of these R parameters is defined by <Var2> (Rm). Before the first call of @710 the input control parameter (Rm + 3) has to be set to "1". The first contour element of the subroutine is stored in R parameters, beginning at the starting point of the contour (Rm +1, Rm +2).

This point is not programmed in the subroutine. The @710 command sets the control parameter to "0" so that on every subsequent call the values of the next contour element are loaded. If the command recognizes "end of subroutine" (M17), the output control parameter (Rn+7) is automatically set to "1" or "2".

Prerequisite

1. Contour in subroutine
2. Starting points
3. Control parameter (Rm + 3) = 1

Parameters

- Rm : Subroutine number
- Rm +1 : Starting point Y
- Rm +2 : Starting point X
- Rm +3 : Control parameter

@710 loads to:

- Rn : Starting point Y
- Rn +1 : Starting point X
- Rn +2 : End point Y
- Rn +3 : End point X
- Rn +4 : Interpolation parameter J
- Rn +5 : Interpolation parameter I
- Rn +6 : G function
- Rn +7 : Control parameter

Rn +7 = 0: Block without M17  
Rn +7 = 1: Block with M17  
Rn +7 = 2: M17 alone in the block
@711 <Var 1> <Var 2> <Var 3> \hspace{1cm} \textbf{Intersection calculation}

This command is required for calculating points of intersection in the L95 “stock removal” turning cycle. The @711 command is used to calculate the point of intersection of a path sought with a contour element. The calculation of the point of intersection requires a total of 8 parameters as input data for the 1st contour element.

The first parameter is determined by <Var 2> (Rn). The second contour starting with <Var 3> (Rr) is presently not implemented, i.e. this notation is currently not required; yet you have to enter some R parameter when programming.

The output data of the intersection calculation are stored in a total of three R parameters starting with the R parameter <Var 1 > (Rm). The search direction is programmed behind the command as a normal traversing movement. Both axis values are required for calculating the point of intersection. The output data from reference processing are used as input data for the intersection calculation.

\begin{itemize}
  \item \textbf{Path sought} \hspace{1cm} \textbf{Point of intersection}
  \item \textbf{Register contents}
\end{itemize}

<table>
<thead>
<tr>
<th>Rn</th>
<th>Block start X</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rn+1</td>
<td>Block start Z</td>
<td>80</td>
</tr>
<tr>
<td>Rn+2</td>
<td>End of block X</td>
<td>50</td>
</tr>
<tr>
<td>Rn+3</td>
<td>End of block Z</td>
<td>100</td>
</tr>
<tr>
<td>Rn+4</td>
<td>Interpolation parameter I</td>
<td>-20</td>
</tr>
<tr>
<td>Rn+5</td>
<td>Interpolation parameter K</td>
<td>0</td>
</tr>
<tr>
<td>Rn+6</td>
<td>G function</td>
<td>2</td>
</tr>
</tbody>
</table>

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\textbf{Rm} & \textbf{Rn} & \textbf{Rr} & \textbf{G01} & \textbf{G90} & \textbf{X0} & \textbf{Z0} & \textbf{LF} \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Rm} & Result 1*) \\
\hline
\textbf{Rm+1} & Point of intersection X 67.320 \\
\hline
\textbf{Rm+2} & Point of intersection Z 90.000 \\
\hline
\end{tabular}
\end{center}

*) 1 = found, 0 = not found
Evaluation of intersection calculation with @711:

1) Point of intersection not found
2) Point of intersection found
3) Arc goes outside quadrant limits (!): Point of intersection not found
4) Path sought was too short: Point of intersection nevertheless found
5) Wrong search direction: Point of intersection nevertheless found

The search direction defines a linear path along which the point of intersection is expected. Whenever this linear path or its extension meets the contour stored in \( R_n \) to \( R_n+6 \), a point of intersection is found. Thus a point of intersection is found even when the path programmed behind @711 is in the wrong direction or is too short.

The @711 function now enters the value "0" into the \( R_m \) register when no point of intersection has been found, or the value "1" when the search has been successful. If a point of intersection has been found, the \( R_{m+1} \) and \( R_{m+2} \) parameters contain the intersection coordinates, otherwise the value "0".

If the values from \( R_{m+1} \) and \( R_{m+2} \), are intended to be used for programming a traversing movement, it must previously be scanned in a loop, whether a point of intersection has been found or not.
Example: Intersection point calculation with @711

The point of intersection with the programmed contour at Z=120 is searched for.

Subroutine of contour:

L20 LF
N5 X50 Z200 LF
N10 G03 X100 Z150 I0 K-50 LF
N15 G03 X50 Z100 I-50 K0 LF
N20 X50 Z50 LF
N25 X150 Z0 LF
N30 M17 LF

Main program:

%30 LF
G0 X150 Z120 LF
R50=20 R51=0 R52=250 R53=1 R70=0 LF
N300 @131 R70 K0 K305 LF
@710 R54 R50 LF
@711 R70 R54 R62 G01 G90 X0 Z120 LF
@100 K-300 LF
N305 G00 X=R71 Z=R72 LF
.
.
M30 LF

R50 to R54, input data; R70: flag for intersection calculation
While loop, as long as R70=0
Reference processing
Intersection calculation
End of loop
Approach the point of intersection in rapid traverse
The program loop is run through until the point of intersection has been found.

<table>
<thead>
<tr>
<th>Call</th>
<th>( R_n )</th>
<th>( R_{n+1} )</th>
<th>( R_{n+2} )</th>
<th>( R_{n+3} )</th>
<th>( R_{n+4} )</th>
<th>( R_{n+5} )</th>
<th>( R_{n+6} )</th>
<th>( R_{n+7} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st call</td>
<td>@710</td>
<td>50</td>
<td>250</td>
<td>50</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2nd call</td>
<td>@710</td>
<td>50</td>
<td>200</td>
<td>100</td>
<td>150</td>
<td>0</td>
<td>-50</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd call</td>
<td>@710</td>
<td>100</td>
<td>150</td>
<td>50</td>
<td>100</td>
<td>-50</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the above example the point of intersection is found after the third call at \( X=90 \) \((R71)\) and \( Z=120 \) \((R72)\).

@713 <Var>

**Start preparation for cycles**

This command is used to transfer the numerical value corresponding to the clearance of 1 mm in the current input format (with \( G70 = 0.03937 \) and with \( G71 = 1 \)) to the \( R \) parameter defined with <Var>.

The numerical value “1” in the case of radius programming and “2” in the case of diameter programming is transferred to the next \( R \) parameter. The allowable parameter range for <Var> is \( R0 \) to \( R98 \) and \( R900 \) to \( R998 \).

@714

**Decoding stop; until buffer empty**

This “STOP-DEC” command prevents block preparation (decoding) until the buffer is empty.

When the program is being processed, several program blocks are decoded in advance and loaded into the buffer of the NC.

This speeds up program processing, but in conjunction with certain NC commands (read actual value, measuring, data transfer NC-PLC) it may cause a faulty program run.

The STOP-DEC command (stop decoding) stops the decoding of NC blocks in advance which stand after this command until the block (@714) has been processed with the STOP-DEC command. This ensures that the buffer is empty and information needed in the following NC blocks is provided.

The STOP-DEC command must be programmed for the following types of information from the PLC, provided they are required for the subsequent NC blocks.

- Machine data
- Setting data
- Tool offsets
- Zero offsets
- \( R \) parameters
- “Mirroring” signal

The STOP-DEC command has to be programmed prior to every reading of actual values in the own channel and after every measuring.

This command has also to be programmed before zero offsets and tool offsets are written if the new values are to be effective only from this block onward.
The STOP-DEC command @714 must always be in an NC block of its own.

Example:
M94 LFR
@714 LFR
@123 R60 K100 K5 LFR
The part number is transferred from the PLC to R60.
Stop decoding in order to enable the current part number to be processed in the next NC block.
Branching according to part number.

@715 Decoding stop; until buffers empty (coordinate rotation)
If the angle of rotation of the settable or programmable coordinate rotation is loaded by means of the CL 800 language, this angle value is immediately used in the calculation. As a result of this, that angle might get included in the calculation of earlier traversing blocks.
If the angle is not to be valid until the following block, all previous blocks must be processed (empty buffers).
By programming STOP DEC 1 this can also be achieved while TNRC is selected.
Decoding is then only reactivated when the buffer is empty before coordinate rotation.

Example:
@715 LFR
@437 K0 K1 K30 LFR
Coordinate rotation G54 with angle of rotation 30° in channel 0 (own channel)

The @715 command must always be in the target code in an NC block of its own.
The @714 command may be used instead of the @715 command; but in this case TNRC must be cancelled first.
Main group 7/subgroup 2: Measurement functions

@720 <Var> <Value>  

Inprocess measurement related to machine zero

This command is used in measuring cycles. The measurement function is used to determine the actual values of the moving axes at the moment of an input signal from the probe. The actual values are acquired directly by the control's measuring circuit when the leading edge of the signal from the probe is recognized.

Beginning at the <Var> parameter, the actual values are stored after the <Var> parameter with axis numbers in ascending order and they are related to machine zero. Then the control generates a "delete distances to go", i.e. the distances to go of all axes are deleted.

The control specifies the setpoint 0 as jump function. The deceleration distances of the axes are traversed, i.e. the following errors are eliminated. As a result of this the following traversing blocks have to be programmed in absolute dimensions (G90).

The acquired actual values of the moving axes at the moment of measuring are stored after the parameter defined under <Var> with axis numbers in ascending order.

The number of the measurement input (1 or 2) is specified using <Value>.

The traversing paths of the axes (setpoints) are programmed via DIN code or the @440 command in the same NC block. The setpoints of the axes are related to machine zero. When the function "axis duplication" is active, command @440 applies both to the leading, programmed axes and to the duplicated axes.

Example:

. .
@720 R93 K1
N35 @440 K1 R70 Lp
. .

The actual value of the first axis is measured and loaded to R93.
The value from R70 is assigned to the traversing path of the first axis via a constant.

If the function “axis duplication” is active, the programmed path is duplicated on the following axis only the actual value of the leading axis is measured. Both axes stop when the measuring probe is triggered.
11.10 I/O functions

The main a group (I/O functions) is structured as follows:

![Diagram](https://example.com/diagram.png)

---

@ a x y Three-digit @ code

Special function

1: I/O functions NC
2: I/O functions general

a: Main group a

---

Main group a/subgroup 1: I/O functions NC

@a15 <Value4> <Value5> (Select menu from NC program)

This command selects menus (displays and menu texts) in the user area and the standard area from the NC program.

- `<Value 4>` = 0: User area
- `<Value 4>` = 1: Standard area
- `<Value 5>` = 1 to 254: Menu number

Example:

```
N.. @a15 K1 K20 L F
```

Calls the standard display for data input/output.

Application:

- Calling configured displays which provide the user with information on the current machining cycle.
- User screenforms that must be acknowledged by the user.

@a1b (Return to the initial menu)

After selecting any menu with @a15, with this command you can return to the menu you had selected before the first @a15 call.

Example:

At present the PLC status display is called.

```
N.. @a15 K1 K20 L F
```

Calling the I/O display

```
N.. @a1b
```

Returning to the PLC status display

Application:

A display selected and overwritten by the user can be called again.
Main group a/subgroup 2: I/O functions in general

@a20 <Value> Select RS232C (V.24) interface
This command selects the interface through which data input/output is to be handled.

<Value> = Number of interface (1 or 2).

Example:

N.. @a20 K2 L_P The second interface is selected.

Note: The interface (1 or 2) selected by the operator is not affected by this command.

Before data is input or output with the commands @a25, @a26, @a27, @a28 and @a29, the interface must be selected with @a20 in the program.

@a25 <Value 1> Output of zero offsets via RS232C (V.24)
With this command, zero offsets and channel-specific angles of rotation can be output via RS232C (V.24).

<Value 1> = 0 Zero offsets G54-G57
1 Angle of rotation for coordinate rotation channel 1
2 " " channel 2
3 " " channel 3.

Example:

N.. @a20 K2 L_P Settable zero offset
N.. @a25 K0 L_P G54-G57 are output via the second RS232C (V.24) interface.
Note:

• Selection of the interface with @a20 must be programmed before calling @a25.

• A machine data bit can be used to select whether data is to be output simultaneously with the running program or whether the block change should be disabled for the time of transmission.

@a26 <Value 2> <Value 3> <Value 4> Output of data via RS232C (V.24)

This command permits particular data to be output via RS232C (V.24). Parameters must be assigned in <Value 2> to designate the data.

<Value 2> = 1: Main program
2: Subroutines
5: NC machine data
6: Tool offset
8: Setting data
9: PLC machine data.

The start address is defined with <Value 3>, the end address of the data block with <Value 4>.

Example:

N..  @a20 K1  L F

The available part programs with program identifiers %1.

N..  @a26  K1  K1  K10  L F

%1 to %10 are output via the first RS232C (V.24) interface.

Note:

• Before data output, the interface must be selected with command @a2.

• A machine data bit can be used to select whether data is to be output simultaneously with the running program or whether the block change should be disabled for the time of transmission.

• If the data to be output are changed in the subsequent program part, a @714 (STOP DEC) must be programmed after the command @a26.

@a27 <Value 1> <Value 3> <Value 4> Parameter output via RS232C (V.24)

With this block individual R parameter blocks can be output via the RS232C (V.24). The channel number is defined via <Value 1>. If a zero is specified, the R parameters are always output from the own channel. For central variables zero is always specified. The start address of the block is defined with <Value 3>, the end address with <Value 4>.

With machine data 5147.1 = “1”, the block change can be disabled in the current program at the same time as R parameters are output.

<Value 1> = Channel number
0: Central R parameter or own channel
1: R parameters channel 1
2: R parameters channel 2
3: R parameters channel 3.

<Value 3> = Start address
000 to 699 for global parameters
700 to 999 for central parameters.

<Value 4> = End address
000 to 699 for global parameters
700 to 999 for central parameters.
Example:

N.. @a20 K1 L_F
N.. @a27 K1 K0 K999 L_F
N.. @a27 K0 K700 K999 L_F
N.. R1=20 R11=40 L_F
N.. @a27 K1 R1 R11 L_F

Select first RS232 C interface
All global parameters of channel 1 are output
All central R parameters are output
Global R parameters 20 to 40 from channel 1 are output.

Application:

- Output of measured values to a printer
- Output of parameters to a linked computer
- Output of data to another NC control.

Note:

Selection of the interface with @a20 must be programmed before calling @a27. If the R parameters to be output are changed in the subsequent program part, a @714 (STOP DEC) must be programmed after the command.

A machine data bit can be used to select whether output is to occur simultaneously with the running program or whether the block change should be disabled for the time of transmission.

@a28 <Value 2> Reading in data via RS232C (V.24)

With this command reading in data via the RS232C (V.24) interface can be started. <Value 2> specifies what type of data may be read in.

<Value 2> = 0: The data type is not checked
1: Main programs
2: Subroutines
3: Clear programs
5: NC machine data
6: Tool offsets
7: Zero offsets
8: Setting data
9: PLC machine data
10: R parameters.

Example:

N.. @a20 K1 L_F
N.. @a28 K10 L_F

Selection of first RS232 C (V.24) interface.
The control expects R parameters in the input. If a different type of data is received, an error message is output.

Note:

Selection of the interface with @a20 must be programmed before calling @a28.

A machine data bit can be used to select whether the data are to be read in simultaneously with the running program or whether the block change should be disabled for the time of data input.

If in the subsequent program part, processing is to involve the R parameters read in, the command @714 (STOP DEC) must be programmed after transmission. Commands @a27 and @a28 are a simple way of transferring data between two SINUMERIK controls.
Prerequisites for NC-NC link:

Interface-specific setting data:

- No header or trailer when reading out.

If the same interface is also required for linking with other devices, the respective setting data can be converted before transmission and subsequently reset.

**Example:**

| N.. | @311 | R700 | K5016 | LF | SD 5016 is saved into R700 |
| N.. | @411 | K5016 | K0000010 | LF | Convert SD 5016 |
| N.. | @a20 | K1 | LF | | Select 1st RS232C (V.24) |
| N.. | @a27 | K1 | K50 | K50 | R50 is output |
| N.. | @714 | IY | | | STOP DEC |
| N.. | @411 | K5016 | R700 | LF | Reset SD 5016. |

**@a29 Output of ETX via RS232C (V.24)**

With this command, the end of transmission character defined in the setting data can be output via the RS232C (V.24) interface.

**Example:**

| N.. | @a20 | K1 | LF | Via the first RS232C (V.24) interface, R parameters |
| N.. | @a27 | K1 | K0 | K100 | R0 to R100 are output from channel 1 followed by the end of transmission character. |
| N.. | @a29 | IY | | |

**Application:**

- After a sequence of outputs without end of transmission character the receiver is informed of the end of the data package with @a29.
11.11 Operator guidance macro (OGM)

**Main group c/subgroup c: Back translation**

@ccc          (Identifier for the beginning of the OGM data block)

Parameterization of machining cycles can be assisted graphically by means of configured input displays. If the user supplies one of these input displays with new machining parameters, these values are first stored in machine input buffers (MIB). For further processing of the input values press a softkey to start an operator guidance macro (OGM).

This OGM assigns the values from the MIB to specified addresses such as G, N, R, X, Y, Z, L, U, I, K, etc. By combining the addresses and the machining input buffer in an operator guidance macro the following data block can be generated:

```
G01 G90  R10 = 50  R11 = 20  R12 = 15  L901 (OGM: GROOVING) L F
```

The OGM enters this data block in a part program as program block. For this purpose, the program must be selected in editing mode beforehand and the insert position must be marked with the cursor.

Back translation means that a program block generated with the aid of an operator guidance macro is put back in its input display. This makes graphically supported editing possible.

A data block prepared for back translation can be recognized by two additional elements:

1. @ccc (x:y) identifies the beginning of the data block. The axis names specify the plane for which the data block has been programmed.

2. (OGM: GROOVING) identifies the end of the data block. "GROOVING" is the name of the OGM.

When such a data block is marked with the cursor in the part program (editing mode), the associated input display is called after pressing the "INPUT DISPLAY" softkey. The information contained in the data block is copied in the MIB and displayed. The complete data block is then:

```
@ccc (x:y) G01 G90  R10 = 50  R11 = 20  R12 = 15  L901 (OGM: GROOVING) L F
```

Back translation is no longer possible if the order of the parameters is changed. Message: "OGM cannot be back translated".

**Configuration of the back translation function**

Every OGM which is to generate a re-editable data block, contains an unambiguous name in the first block. The name is programmed like a comment and must have the following format:

```
(OGM: GROOVING1)   "GROOVING1" is the name.
```

This name is entered in an assignment list. OGM 999 has the function of the assignment list. It assigns the OGM number and the menu block number (number of the configured input display) to the name.
The name may have a length of 12 characters. Blanks are not permitted. If the name contains more than 12 characters or if it is not entered in the assignment list the message "OGM not found" is output.

**Example:** Structure of the assignment list in OGM 999

```plaintext
% OGM 999
GROOVING1 = 7, 11
GROOVING2 = 9, 12
MACHINING = 12, 13
TAPER = 4, 25
```

Menu block number
OGM number

The initial @ccc identifier is generated if after the OGM number the OGM name is configured in the form: (OGM:<name>). A data block configured like this can be edited later graphically by means of the "Back translation" function.

**Example:** Structure of OGM 7

Configured data block

```plaintext
% OGM 7
(OGM: GROOVING1)
N~150  G~100  G~101  R~=103  R12=104  R13=105 LF
R14=106  R15=107  L901  LF
```

This OGM 7 would insert the following data block in the part program at the position marked with the cursor:

```plaintext
@ccc (x:z:y)  N2  G01  G90  R10=50  R11=20  R12=15  R13=25  LF
R14=27  R15=39  L901 (OGM:GROOVING)  LF
```

**Note:**
- The maximum block length of 120 characters and the maximum OGM name length of 12 characters should be observed.
- The first instruction should be a block number ("N..."). Thus the data block can be found quicker when editing.
- In the part program the OGM may not generate a block number or a comment at the beginning of the block.
- To delete the data block, position the cursor in the data block to be deleted, enter "N+" and press the "CANCEL" key or start the procedure with the configured softkey "Delete block" after having positioned the cursor.
### 11.12 @ code table

**Key:**

<table>
<thead>
<tr>
<th>y</th>
<th>Relational operator rop</th>
<th>1) Not at CL800 level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>. . . . No condition</td>
<td>2) &quot;Condition&quot;</td>
</tr>
<tr>
<td>1:=</td>
<td>. . . . Equal to</td>
<td>a) Var =Boolean variable</td>
</tr>
<tr>
<td>2:</td>
<td>. . . . Not equal to</td>
<td>b) Var . Const =Bit from pattern</td>
</tr>
<tr>
<td>3:</td>
<td>. . . . Greater than</td>
<td>c) Var &quot;rop&quot; Value</td>
</tr>
<tr>
<td>4:</td>
<td>= . . . Greater than or equal to</td>
<td>d) Extended condition</td>
</tr>
<tr>
<td>5:</td>
<td>. . . . Less than</td>
<td>3) No pointers possible,</td>
</tr>
<tr>
<td>6:</td>
<td>= . . . Less than or equal to</td>
<td>only Const definable on CL 800 level</td>
</tr>
<tr>
<td>7:</td>
<td>. . . True</td>
<td></td>
</tr>
<tr>
<td>8:</td>
<td>. . . not</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>@ Code</th>
<th>CL 800 statement</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>@00f</td>
<td></td>
<td>Enable for softkey start</td>
</tr>
<tr>
<td>@041</td>
<td>$\text{R Par 1 R Par 2}$</td>
<td>Saving of a group of R parameters to stack</td>
</tr>
<tr>
<td>@042</td>
<td>Const $\text{R Par n . . .}$ $\text{R Par 1}$</td>
<td>(Pop)(^1) Fetching saved R parameters from stack</td>
</tr>
<tr>
<td>@043</td>
<td>$\text{R Par 1 R Par 2}$</td>
<td>(Pop Block)(^3) Fetching group of saved R parameters from stack</td>
</tr>
<tr>
<td>@100</td>
<td>Const $\text{R-Par}$</td>
<td>GOTO Label ; Absolute jump to NC block</td>
</tr>
<tr>
<td>@100</td>
<td>$\text{R-Par}$</td>
<td>3)</td>
</tr>
<tr>
<td>@111</td>
<td>$\text{Var Value1 Const 1}$ $\text{Value2 Const 2}$</td>
<td>CASE Var = Value 1 : Statement 1 ; CASE branching</td>
</tr>
<tr>
<td></td>
<td>. . .</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>. . .</td>
<td>= Value n :</td>
</tr>
<tr>
<td></td>
<td>. . .</td>
<td>Statement n ;</td>
</tr>
<tr>
<td>@12y</td>
<td>$\text{Var Value Const}$</td>
<td>IF condition (^2) THEN Statement 1 ; IF-THEN-ELSE statement y relational operator rop Var R parameter or pointer</td>
</tr>
<tr>
<td>@13y</td>
<td>$\text{Var Value Const}$</td>
<td>WHILE condition (^2) DO Statement ; Repeat statement with sampling of repeat condition at start. y relational operator rop</td>
</tr>
<tr>
<td>@14y</td>
<td>$\text{Var Value Const}$</td>
<td>REPEAT Statement ; UNTIL condition; (^2) Repeat statement with sampling of repeat condition at end. y relational operator rop</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@151 Var Value 2 Const</td>
<td>FOR Var = Value 1 TO Value 2 DO Statement ;</td>
<td>Repeat statement with repetitions until Var has incrementally reached Value 2</td>
</tr>
<tr>
<td>@161 Var Value 2 Const</td>
<td>FOR Var = Value 1 DOWN TO Value 2 DO Statement ;</td>
<td>Repeat statement with repetitions until Var has decrementally reached Value 2</td>
</tr>
<tr>
<td>@200 Var</td>
<td>CLEAR( Var ) ;</td>
<td>Delete variable</td>
</tr>
<tr>
<td>@201 Var Value</td>
<td>Var = Value</td>
<td>Load variable with value</td>
</tr>
<tr>
<td>@202 Var 1 Var 2</td>
<td>XCHG ( Var 1 , Var 2 );</td>
<td>Swapping of variable contents</td>
</tr>
<tr>
<td>@203 Var 1 Var 2 Const</td>
<td>Reading of a bit from bit pattern</td>
<td></td>
</tr>
<tr>
<td>@210 Value 3 Value 4</td>
<td>CLEAR MIB ( &lt;Value 3&gt;, &lt;Value 4&gt; );</td>
<td>Clear machine input buffer Value 3: MIB start address 0 . . . 499 Value 4: MIB end address 0 . . . 499</td>
</tr>
<tr>
<td>@211 Var Value 1</td>
<td>Var = MIB ( Value );</td>
<td>Load numerical variable Var with the contents of MIB (machine input buffer) cell Value 1 Value 1: MIB No. 0 . . . 499</td>
</tr>
<tr>
<td>@212 Value 1 Value</td>
<td>MIB ( Value1 )= Value ;</td>
<td>Load MIB (machine input buffer) cell Value 1 with the numerical number Value Value 1: MIB No. 0 . . . 499</td>
</tr>
<tr>
<td>@300 Var Value 1</td>
<td>Var =MDN ( Value 1 );</td>
<td>Machine data NC Value 1: Addr. 0 . . . 4999</td>
</tr>
<tr>
<td>@301 Var Value 1</td>
<td>Var =MDNBY ( Value 1 );</td>
<td>Machine data NC byte Value 1: Byte addresses 5000 . . . 6999</td>
</tr>
<tr>
<td>@302 Var Value 1 Value 2</td>
<td>Var =MDNBI ( Value 1 , Value 2 );</td>
<td>Machine data NC bit Value 1: Byte addresses 5000 . . . 6999 Value 2: Bit addr. 0 . . . 7</td>
</tr>
<tr>
<td>@306 Var Value 1</td>
<td>Var =MDP ( Value 1 );</td>
<td>Machine data PLC Value 1: Addr. 0 . . . 1999</td>
</tr>
<tr>
<td>@307 Var Value 1</td>
<td>Var =MDPBY ( Value 1 );</td>
<td>Machine data PLC bytes Value 1: Byte addresses 2000 . . . 3999</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@308</td>
<td>Var Value 1 Value 2</td>
<td>Var = MDPBI (Value 1, Value 2); Machine data PLC bit Value 1: Byte addresses 2000 . . . 3999 Value 2: Bit addr. 0 . . . 7</td>
</tr>
<tr>
<td>@310</td>
<td>Var Value 1</td>
<td>Var = SEN (Value 1); Setting data NC Value 1: Addr. 0 . . . 4999</td>
</tr>
<tr>
<td>@311</td>
<td>Var Value 1</td>
<td>Var = SENBY, (Value 1); Setting data NC byte Value 1: Byte addresses 5000 . . . 9999</td>
</tr>
<tr>
<td>@312</td>
<td>Var Value 1 Value 2</td>
<td>Var = SENBI (Value 1, Value 2); Setting data NC bit Value 1: Byte 5000 . . . 9999 Value 2: Bit addr. 0 . . . 7</td>
</tr>
<tr>
<td>@320</td>
<td>Var Value 1 Value 2 Value 3</td>
<td>Var = TOS (Value 1, Value 2, Value 3); Tool offset Value 1: 0 Value 2: D no. 0 . . 99 Value 3: P no. 0 . . (9)</td>
</tr>
<tr>
<td>@330</td>
<td>Var Value 1 Value 2 Value 3</td>
<td>Var = ZOA (Value 1, Value 2, Value 3); Settable zero offset (G54-G57) Value 1: Group 1 . . . 4 (G54-G57) Value 2: Axis no. 1 . . 4 Value 3: Coarse/fine (0/1)</td>
</tr>
<tr>
<td>@331</td>
<td>Var Value 1 Value 2</td>
<td>Var = ZOPR (Value 1, Value 2); Programmable zero offset (G58,G59) Value 1: Group 1 or 2 (G58 or G59) Value 2: Axis no. 1, 2</td>
</tr>
<tr>
<td>@332</td>
<td>Var Value 2</td>
<td>Var = ZOE (Value 2); External zero offset from PLC Value 2: Axis no. 1, 2</td>
</tr>
<tr>
<td>@333</td>
<td>Var Value 2</td>
<td>Var = ZOD (Value 2); DRF offset Value 2: Axis no. 1, 2</td>
</tr>
<tr>
<td>@334</td>
<td>Var Value 2</td>
<td>Var = ZOPS (Value 2); PRESET offset Value 2: Axis no. 1, 2</td>
</tr>
<tr>
<td>@336</td>
<td>Var Value 2</td>
<td>Var = ZOS (Value 2); Total offset Value 2: Axis no. 1, 2</td>
</tr>
<tr>
<td>@337</td>
<td>Var Value 1 Value 2 Value 3</td>
<td>Var = ZOA DW (Value 1, Value 2, Value 3); Settable coordinate rotation (G54-G57) Value 1: Channel no. 0 . . 2 Value 2: Group 1 . . . 4 (G54-G57) Value 3: Angle no. (=1)</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@338</td>
<td>Var =ZOPRDW ( Value 1 , Value 2 , Value 3 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Programmable coordinate rotation (G58-G59)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 1: Channel no. 0 . . .2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 2: Group 1 or 2 (G58 or G59)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 3: Angle no. (= 1)</td>
</tr>
<tr>
<td>@342</td>
<td>Var =PRSS ( Value 1 , Value 3 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read programmed spindle speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 1: Chan. no. 0 ... 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 3: Spindle no.0 ... 6</td>
</tr>
<tr>
<td>@345</td>
<td>Var =PRVC ( Value 1 , Value 2 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Programmed cutting speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 1: Channel no. 0 . . .2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 3: 0=G96</td>
</tr>
<tr>
<td>@360</td>
<td>Var =ACPW ( Value 2 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual axis position workpiece-related</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 2: Axis no. 1, 2 . . .</td>
</tr>
<tr>
<td>@361</td>
<td>Var =ACPM ( Value 2 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual axis position machine-related</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 2: Axis no. 1, 2 . . .</td>
</tr>
<tr>
<td>@363</td>
<td>Var =ACSP ( Value 2 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual spindle position</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 2: Spindle no. 0 . . .6</td>
</tr>
<tr>
<td>@364</td>
<td>Var =ACSS ( Value 2 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual spindle speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 2: Spindle no. 0 . . .6</td>
</tr>
<tr>
<td>@367</td>
<td>Var =ACAS ( Value 1 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read axis number of current plane/leading spindle no. into R parameter Var :</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Var+0: no. of horizontal axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Var+1: no. of vertical axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Var+2: no. of the axis perpendicular to the plane</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Var+3: no. of the axis in which length 2 is effective (tool type 30)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Var+4: no. leading spindle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 1: Channel no. 0 . . .2</td>
</tr>
<tr>
<td>@36a</td>
<td>Var =ACD ( Value 1 );</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual D function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 1=0</td>
</tr>
<tr>
<td>Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@36b</td>
<td><code>Var =ACG (Value 1, Value 3);</code></td>
<td>Reading the G function of the current out of the working memory. Value 1: Chan. no. 0 . . . 2 Value 3: Internal G Group to which the G function belongs 0 . . . 15</td>
</tr>
<tr>
<td>@371</td>
<td><code>Var =SOB (Value 1, Value 3);</code></td>
<td>Special bits Value 1: Chan. no. 0 . . . 2 = channel dependent 99=channel-independent Value 3: Bit no. 0 . . . 7</td>
</tr>
<tr>
<td>@380</td>
<td><code>Var =PLCI (Value 1, Value 2, Value 3);</code></td>
<td>PLC input bit Value 1: PLC No. 1 Value 2: Byte addr. 0...255 Value 3: Bit No. 0...7</td>
</tr>
<tr>
<td>@381</td>
<td><code>Var =PLCQ (Value 1, Value 2, Value 3);</code></td>
<td>PLC output bit Value 1: PLC No. 1 Value 2: Byte addr. 0...255 Value 3: Bit No. 0...7</td>
</tr>
<tr>
<td>@382</td>
<td><code>Var =PLCF (Value 1, Value 2, Value 3);</code></td>
<td>PLC flag bit Value 1: PLC No. 1 Value 2: Byte addr. 0...255 Value 3: Bit No. 0...7</td>
</tr>
<tr>
<td>@383</td>
<td><code>Var =PLCF (Value 1, Value 2, Value 3, Value 4);</code></td>
<td>PLC data word bit Value 1: PLC No. 1 Value 2: DB No. 0...255 Value 3: DW No. 0...255 Value 4: Bit No. 0...15</td>
</tr>
<tr>
<td>@3e4</td>
<td><code>Var =AGS (Value 1);</code></td>
<td>Read active gear speed Value 1: Spindle nos. 0 to 6</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@400</td>
<td>MDN Value 1 = Value;</td>
<td>Machine data NC</td>
</tr>
<tr>
<td></td>
<td>Value 1: Addr. 0 . . 4999</td>
<td></td>
</tr>
<tr>
<td>@401</td>
<td>MDNBY Value 1 = Value;</td>
<td>Machine data NC byte</td>
</tr>
<tr>
<td></td>
<td>Value 1: Byte addresses 5000 . . 6999</td>
<td></td>
</tr>
<tr>
<td>@402</td>
<td>MDNBI Value 1 , Value 2 = Value;</td>
<td>Machine data NC bit</td>
</tr>
<tr>
<td></td>
<td>Value 1: Byte addresses 5000 . . 6999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 2: Bit addr. 0 . . 7</td>
<td></td>
</tr>
<tr>
<td>@406</td>
<td>MDP Value 1 = Value;</td>
<td>Machine data PLC</td>
</tr>
<tr>
<td></td>
<td>Value 1: Addr. 0 . . 1999</td>
<td></td>
</tr>
<tr>
<td>@407</td>
<td>MDPBY Value 1 = Value;</td>
<td>Machine data PLC byte</td>
</tr>
<tr>
<td></td>
<td>Value 1: Byte addresses 2000 . . 3999</td>
<td></td>
</tr>
<tr>
<td>@408</td>
<td>MDNBI Value 1 , Value 2 = Value;</td>
<td>Machine data PLC bit</td>
</tr>
<tr>
<td></td>
<td>Value 1: Byte addresses 2000 . . 3999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 2: Bit addr. 0 . . 7</td>
<td></td>
</tr>
<tr>
<td>@410</td>
<td>SEN Value 1 = Value;</td>
<td>Setting data NC</td>
</tr>
<tr>
<td></td>
<td>Value 1: Addr. 0 . . 4999</td>
<td></td>
</tr>
<tr>
<td>@411</td>
<td>SENBY Value 1 = Value;</td>
<td>Setting data NC byte</td>
</tr>
<tr>
<td></td>
<td>Value 1: Byte addresses 5000 . . 9999</td>
<td></td>
</tr>
<tr>
<td>@412</td>
<td>SENBI Value 1 , Value 2 = Value</td>
<td>Setting data NC bit</td>
</tr>
<tr>
<td></td>
<td>Value 1: Byte addresses 5000 . . 9999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 2: Bit addr. 0 . . 7</td>
<td></td>
</tr>
<tr>
<td>@420</td>
<td>TOS ( Value 1 , Value 2 , Value 3 )= Value ;</td>
<td>Tool offset</td>
</tr>
<tr>
<td></td>
<td>Value 1: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 2: D no. 1 . . 99</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 3: P no. 0 . . 7 (9)</td>
<td></td>
</tr>
<tr>
<td>@423</td>
<td>TOAD ( Value 1 , Value 2 , Value 3 )= Value ;</td>
<td>Tool offset additive</td>
</tr>
<tr>
<td></td>
<td>Value 1: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 2: D No. 1 . . 99</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 3: P No. 0 . . 7 (9)</td>
<td></td>
</tr>
<tr>
<td>@430</td>
<td>ZOA ( Value 1 , Value 2 , Value 3 )= Value ;</td>
<td>Settable zero offset (G54-G57)</td>
</tr>
<tr>
<td></td>
<td>Value 1: Group 1 . . . 4 (G54-G57)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 2: Axis no. 1, 2 . . .</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value 3: coarse/fine (0/1)</td>
<td></td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@431</td>
<td>ZOFA (Value 1, Value 2, Value 3) = Value;</td>
<td>Settable zero offset additive Value 1: Group 1...4 (G54-G57) Value 2: Axis no. 1, 2... Value 3: coarse/fine (0/1)</td>
</tr>
<tr>
<td>@432</td>
<td>ZOPR (Value 1, Value 2) = Value;</td>
<td>Programmable zero offset (G58,G59) Value 1: Group 1 or 2 (G58 or G59) Value 2: Axis no. 1, 2...</td>
</tr>
<tr>
<td>@434</td>
<td>ZOD (Value 2) = Value;</td>
<td>DRF offset Value 2: Axis no. 1, 2...</td>
</tr>
<tr>
<td>@435</td>
<td>ZOPS (Value 2) = Value;</td>
<td>PRESET offset Value 2: Axis no. 1, 2...</td>
</tr>
<tr>
<td>@437</td>
<td>ZOADW (Value 1, Value 2, Value 3) = Value;</td>
<td>Settable coordinate rotation absolute Value 1: Channel no. 0...2 Value 2: Group 1...4 (G54-G57) Value 3: Angle no. (= 1)</td>
</tr>
<tr>
<td>@438</td>
<td>ZOFADW (Value 1, Value 2, Value 3) = Value;</td>
<td>Settable coordinate rotation additive Value 1: Channel no. 0...2 Value 2: Group 1...4 Value 3: Angle no. (= 1)</td>
</tr>
<tr>
<td>@439</td>
<td>ZOPRDW (Value 1, Value 2, Value 3) = Value;</td>
<td>Programmable coordinate rotation Value 1: Channel no. 0...2 Value 2: Group 1 or 2 (G58-G59) Value 3: Angle no. (= 1)</td>
</tr>
<tr>
<td>@43a</td>
<td>ZOFPRDW (Value 1, Value 2, Value 3) = Value;</td>
<td>Programmable coordinate rotation additive Value 1: Channel no. 0...2 Value 2: Group 1 or 2 (G58 or G59) Value 3: Angle no. (= 1)</td>
</tr>
<tr>
<td>@440</td>
<td>PRAP (Value 3) = Value;</td>
<td>Programmed axis position Value 3: Axis no. 1, 2...</td>
</tr>
<tr>
<td>@442</td>
<td>PRSS (Value 3) = Value;</td>
<td>Programmed spindle speed Value 3: Spindle no. 0...6</td>
</tr>
<tr>
<td>@446</td>
<td>PRAD = Value;</td>
<td>Programmed radius</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@447 Value</td>
<td>PANG = Value ;</td>
<td>Programmed angle</td>
</tr>
<tr>
<td>@448 Value 3 Value</td>
<td>PRIP ( Value 3 ) = Value ;</td>
<td>Programmed interpolation parameter for circle and thread Value 3: Axis no. 1,2 ...</td>
</tr>
<tr>
<td>@482 Value 1 Value 2 Value 3 Value</td>
<td>PLCF ( Value 1 , Value 2 , Value 3 ) = Value ;</td>
<td>PLC flag bit Value 1: PLC No. 1 Value 2: Byte addr. 100...115, 136...199, 224...255 Value 3: Bit No. 0...7</td>
</tr>
<tr>
<td>@483 Value 1 Value 2 Value 3 Value 4 Value</td>
<td>PLCW ( Value 1 , Value 2 , Value 3 ) = Value ;</td>
<td>PLC data word bit Value 1: PLC No. 1 Value 2: DB No. 1...255 Value 3: DW No. 0...255 Value 4: Bit No. 0...15</td>
</tr>
<tr>
<td>@4e1 Value 1 Value 2 Value</td>
<td>SATC ( Value 1 , ( Value 2 ) = Value ;</td>
<td>Spindle acceleration time constant Value 1: Spindle no. 0 to 6 Value 2: Gear speeds 1 to 8 Value 3: Spindle acceleration time constant 0 to 16000</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{Var} & = \text{Value 1} + \text{Value 2} ; \\
\text{Var} & = \text{Value 1} - \text{Value 2} ; \\
\text{Var} & = \text{Value 1} \times \text{Value 2} ; \\
\text{Var} & = \text{Value 1} / \text{Value 2} ; \\
\end{align*}
\]

- Addition
- Subtraction
- Multiplication
- Division

<p>| @610 Var Value | Var = ABS ( Value );       | Absolute value generation                                               |
| @613 Var Value | Var = SQRT ( Value );      | Square root                                                             |
| @614 Var Value 1 Value 2 | Var = SQRTS ( Value 1 , Value 2 ); | Root from sum of squares                                                 |
| @620 Var       | INC ( Var );              | Incrementing of &quot;Var&quot; by 1                                              |
| @621 Var       | DEC ( Var );              | Decrementing of &quot;Var&quot; by 1                                              |
| @622 Var       | TRUNC ( Var );            | Integer                                                                  |
| @630 Var Value | Var = SIN ( Value );      | Sine                                                                     |
| @631 Var Value | Var = COS ( Value );      | Cosine                                                                   |
| @632 Var Value | Var = TAN ( Value );      | Tangent                                                                  |
| @634 Var Value | Var = ARCSIN ( Value );   | Arc sine                                                                 |
| @637 Var Value 1 Value 2 | Var = ANGLE ( Value 1 , Value 2 ); | Angle from two vector components                                         |</p>
<table>
<thead>
<tr>
<th>@ Code</th>
<th>CL 800 statement</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>@640 Var Value</td>
<td>Var = LN Value ;</td>
<td>Natural logarithm</td>
</tr>
<tr>
<td>@641 Var Value</td>
<td>Var = INV LN ( Value )</td>
<td>( e^x ) Exponential function</td>
</tr>
<tr>
<td>@650 Var Var 1 Value</td>
<td>Var = Var 1 OR Value ;</td>
<td>OR</td>
</tr>
<tr>
<td>@651 Var Var 1 Value</td>
<td>Var = Var 1 XOR Value ;</td>
<td>Exclusive OR</td>
</tr>
<tr>
<td>@652 Var Var 1 Value</td>
<td>Var = Var 1 AND Value ;</td>
<td>AND</td>
</tr>
<tr>
<td>@653 Var Var 1 Value</td>
<td>Var = Var 1 NAND Value ;</td>
<td>NAND</td>
</tr>
<tr>
<td>@654 Var Value</td>
<td>Var = NOT Value</td>
<td>NOT</td>
</tr>
<tr>
<td>@655 Var Var 1 Value</td>
<td>Var = Var 1 ORB Value ;</td>
<td>OR bit</td>
</tr>
<tr>
<td>@656 Var Var 1 Value</td>
<td>Var = Var 1 XORB Value ;</td>
<td>EXCLUSIVE OR bit</td>
</tr>
<tr>
<td>@657 Var Var 1 Value</td>
<td>Var = Var 1 ANDB Value ;</td>
<td>AND bit</td>
</tr>
<tr>
<td>@658 Var Var 1 Value</td>
<td>Var = Var 1 NANDB Value ;</td>
<td>NAND bit</td>
</tr>
<tr>
<td>@659 Var Value</td>
<td>Var = NOTB Value</td>
<td>NOT bit</td>
</tr>
<tr>
<td>@660 Var Const</td>
<td>CLEAR BIT ( Var . Const );</td>
<td>Delete bit in pattern Const=bit No. 0 ... 7</td>
</tr>
<tr>
<td>@661 Var Const</td>
<td>SET BIT ( Var . Const );</td>
<td>Set bit; Const=bit No. 0 ... 7</td>
</tr>
<tr>
<td>@67y Var 1 Var 2 Value</td>
<td>If the relation of Var 2 and Value is fulfilled, Boolean variable Var 1 is set to “1”</td>
<td></td>
</tr>
<tr>
<td>@706</td>
<td>POS MSYS</td>
<td>Specification of a position in relation to the actual-value system of the machine</td>
</tr>
<tr>
<td>@710 Var 1 Var 2</td>
<td>Var 1 = PREP REF ( Var2 );</td>
<td>Reference processing Var 1: Outp. data from Var 1 Var 2: Inp. data from Var 2</td>
</tr>
<tr>
<td>@711 Var 1 Var 2 Var 3</td>
<td>Var 1 = INT SEC ( Var 2 Var 3 );</td>
<td>Intersection calculation Var 1: Outp. data from Var 1 Var 2: 1st contour from Var 2 Var 3: Preset with 0</td>
</tr>
<tr>
<td>@713 Var</td>
<td>Var = PREP CYC;</td>
<td>Start preparation for cycles Var: Output data from Var</td>
</tr>
<tr>
<td>@714</td>
<td>STOP DEC;</td>
<td>Decoding stop; until buffer empty.</td>
</tr>
<tr>
<td>@715</td>
<td>STOP DEC 1;</td>
<td>Stop of decoding until buffer is empty (applies to coordinate rotation)</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@720 Var Value</td>
<td>Var = MEAS M Value ;</td>
<td>Inprocess measurement Var: Data stored from Var Value: No. of measurement input: 1 or 2</td>
</tr>
<tr>
<td>@a15 Value 4 Value 5</td>
<td>WRT PIC (Value 1, Value 2, Value 3);</td>
<td>Selection of displays and menu texts Value 4: 0=User area 1=Standard area Value 5: 1...254 Menu No.</td>
</tr>
<tr>
<td>@a1b</td>
<td>RECALL PIC</td>
<td>Return to the initial menu</td>
</tr>
<tr>
<td>@a20 Value</td>
<td>PORT Value</td>
<td>Selection of interface RS232C (V.24) Value: Number of interface 1 or 2</td>
</tr>
<tr>
<td>@a25 Value 1</td>
<td>OUTP ZOA (Value 1);</td>
<td>Output zero offsets via RS232C (V.24) Value 1: Chan. No. 0 . . . 3 0= ZO G54 - G57 1= Angle of rotation Channel 1 2= Angle of rotation Channel 2 3= Angle of rotation Channel 3</td>
</tr>
<tr>
<td>@a26 Value 2 Value 3 Value 4</td>
<td>OUTP DATA (Value 2 Value 3 Value 4);</td>
<td>Output data via RS232C (V.24) Value 2: Data type identifier 1= Main program 2= Subroutine 5= NC machine data 6= Tool offsets 8= Setting data 9= PLC machine data Value 3: Start address Value 4: End address</td>
</tr>
<tr>
<td>@ Code</td>
<td>CL 800 statement</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| @a27 Value 1 Value 3 Value 4 | OUTP PARA ( Value 1 Value 3 Value 4 ); | Parameter output via RS232C (V.24)  
Value 1: Channel no.  
Value 2: Start address  
Value 3: End address |
| @a28 Value 2 | INP ( Value 2 ); | Read-in via RS232C (V.24)  
Value 2: 0= No data type check  
. . .  
10=R parameter |
| @a29 | OUTP ETX | Output of the end-of-transmission character via RS232C (V.24) |
| @ccc | | Back translation to the input screenform.  
Identifier for the beginning of the "OGM data block" |
# 12 SINUMERIK 810T/820T Program Key

## 12.1 Internal G groups for @36b

<table>
<thead>
<tr>
<th>Group G intern</th>
<th>G Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hex Code</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>09</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>40 T</td>
</tr>
<tr>
<td>4</td>
<td>53</td>
</tr>
<tr>
<td>5</td>
<td>54 T</td>
</tr>
<tr>
<td>6</td>
<td>04</td>
</tr>
<tr>
<td>7</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>70</td>
</tr>
<tr>
<td>9</td>
<td>80 T</td>
</tr>
<tr>
<td>10</td>
<td>90 T</td>
</tr>
<tr>
<td>11</td>
<td>94</td>
</tr>
<tr>
<td>12</td>
<td>147</td>
</tr>
<tr>
<td>13</td>
<td>50 T</td>
</tr>
</tbody>
</table>

* T: Initial setting (overall reset) of the G group
## 12.2 Program key

<table>
<thead>
<tr>
<th>Group</th>
<th>EA</th>
<th>ISO</th>
<th>Value range/code</th>
<th>Function and significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOR</td>
<td>%</td>
<td>0</td>
<td>Program number</td>
<td></td>
</tr>
<tr>
<td>mpf ...</td>
<td>MPF</td>
<td>0</td>
<td>Main program</td>
<td></td>
</tr>
<tr>
<td>spf ...</td>
<td>SPF</td>
<td>1</td>
<td>Subroutine</td>
<td></td>
</tr>
<tr>
<td>... EOB</td>
<td>...LF</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>N</td>
<td>1</td>
<td>Main block</td>
<td></td>
</tr>
<tr>
<td>n/o</td>
<td>/N</td>
<td>.</td>
<td>Skippable main block</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>/N</td>
<td>.</td>
<td>Skippable subblock</td>
<td></td>
</tr>
</tbody>
</table>

### G0

g G 00 Rapid traverse, coarse exact positioning
  01 * Linear interpolation
  02 Circular interpolation, clockwise
  03 Circular interpolation, counter-clockwise
  06 Spline Interpolation
  10 Polar coordinate programming, rapid traverse
  11 Polar coordinate programming, linear interpolation
  12 Polar coordinate programming, circular interpolation clockwise
  13 Polar coordinate programming, circular interpolation counter-clockwise
  33 Thread cutting, constant lead
  34 Thread cutting, linearly progressive lead
  35 Thread cutting, linearly degressive lead
  36 Tapping without compensating chuck

### G1

g G 09 # Speed reduction, fine exact positioning

### G2

g G 16 Plane selection with free axis selection
  17 Plane selection X - Y (depending on machine data)
  18 * Plane selection Z - X (depending on machine data)
  19 Plane selection Y - Z (depending on machine data)

### G3

g G 40 * No tool nose radius compensation
  41 Tool nose radius compensation, counter-clockwise
  42 Tool nose radius compensation, clockwise

### G4

g G 53 # Suppression of zero offset

### G5

g G 54 * Zero offset 1
  55 Zero offset 2
  56 Zero offset 3
  57 Zero offset 4

### G6

g G 04 # 1) Dwell, duration predetermined in seconds under address X or F and in revolutions under address S. The time range is between:
  0.001 to 99999.999 s with X
  0.001 to 99.999 s with F
  0.1 to 99.9 with S
  25 # 1) Minimum working area limitation
  26 # 1) Maximum working area limitation
  58 # 1) Programmable additive zero offset 1
  59 # 1) Programmable additive zero offset 2
  74 # 1) Reference point approach via part program (PP)
  92 1) Limitation of set spindle speed under address S

### G7

g G 60 Speed reduction, fine exact positioning
  62 Continuous path operation, block transition with speed reduction
  63 Tapping without encoder, 100% feedrate override
  64 * Continuous path operation, block transition with speed reduction

### G8

g G 70 Inch input system
  71 Metric input system

© Siemens AG 1990 All Rights Reserved
SINUMERIK 810/820T, GA3 (BN)
<table>
<thead>
<tr>
<th>Group</th>
<th>EIA</th>
<th>ISO</th>
<th>Value range/code</th>
<th>Function and significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>G9</td>
<td>g</td>
<td>G</td>
<td>80 *</td>
<td>Delete G81 to G89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>81</td>
<td>Call cycle L81 - drilling, centering axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>82</td>
<td>Call cycle L82 - drilling, spot facing axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>83</td>
<td>Call cycle L83 - deep-hole drilling axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>84</td>
<td>Call cycle L84 - thread tapping with encoder axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>85</td>
<td>Call cycle L85 - boring 1 axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>86</td>
<td>Call cycle L86 - boring 2 axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>87</td>
<td>Call cycle L87 - boring 3 axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>88</td>
<td>Call cycle L88 - boring 4 axis switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>89</td>
<td>Call cycle L89 - boring 5 axis switching</td>
</tr>
<tr>
<td>G10</td>
<td>g</td>
<td>G</td>
<td>90 *</td>
<td>Absolute dimensioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>91</td>
<td>Incremental dimensioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>68</td>
<td>Absolute dimensioning on direct way (only with rotary axis)</td>
</tr>
<tr>
<td>G11</td>
<td>g</td>
<td>G</td>
<td>94 *</td>
<td>Feedrate under address F in mm/min or inches/min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td>Feedrate under address F in mm/rev or inches/rev</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>Feedrate under Address F in mm/rev or inch/rev and constant cutting speed under address S in m/min or f/min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>97</td>
<td>Cancel G96, store last set speed of G96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>98</td>
<td>Feedrate for rotary axis in rev/min</td>
</tr>
<tr>
<td>G12</td>
<td>g</td>
<td>G</td>
<td>110</td>
<td>Polar coordinate programming, take setpoint position reached as new centre</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td>Polar coordinate programming, centre programming with angle and radius</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>147 # 1)</td>
<td>Soft approach to contour with linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>247 # 1)</td>
<td>Soft approach to contour with quarter circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>347 # 1)</td>
<td>Soft approach to contour with semicircle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>148 # 1)</td>
<td>Soft leaving with linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>248 # 1)</td>
<td>Soft leaving with quarter circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>348 # 1)</td>
<td>Soft leaving with semicircle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>48 # 1)</td>
<td>Leave as approached</td>
</tr>
<tr>
<td>G13</td>
<td>g</td>
<td>G</td>
<td>50 *</td>
<td>Cancellation of scale modification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51</td>
<td>Scale modification</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>X</td>
<td>4)</td>
<td>± 0.001 to ±99999.999 Position data in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ±9999.999</td>
<td>Position data in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to ±99999.999</td>
<td>Dwell in sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.001* to ±99999.999*</td>
<td>Position data in degrees</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>Y</td>
<td>4)</td>
<td>± 0.001 to ±99999.999 Position data in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ±9999.999</td>
<td>Position data in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to ±99999.999</td>
<td>Dwell in sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.001* to ±99999.999*</td>
<td>Position data in degrees</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>Z</td>
<td>4)</td>
<td>± 0.001 to ±99999.999 Position data in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ±9999.999</td>
<td>Position data in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.001* to ±99999.999*</td>
<td>Position data in degrees</td>
</tr>
<tr>
<td></td>
<td>q</td>
<td>Q</td>
<td>4)</td>
<td>± 0.001 to ±99999.999 Auxiliary axes, position data in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ±9999.999</td>
<td>Auxiliary axes, position data in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.001* to ±99999.999*</td>
<td>Position data in degrees</td>
</tr>
</tbody>
</table>

1) No other functions may be written in this block
3) Other addresses selectable (A, B, C, E, U, V, W)
# Active block by block, all others modal
* Initial setting (after Reset, M02/M30, after powering up controller)
4) The range is set via machine data. Refer to the machine manufacturer for the valid range of values.
<table>
<thead>
<tr>
<th>Group</th>
<th>EIA</th>
<th>ISO</th>
<th>Value range</th>
<th>Function and significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>A</td>
<td>3)</td>
<td>0 to 359.99999*</td>
<td>Angle in degrees with contour definition</td>
</tr>
<tr>
<td>b</td>
<td>B</td>
<td>3)</td>
<td>± 0.001 to + 99999.999</td>
<td>Radius with contour definition in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to + 3999.9999</td>
<td>Radius with contour definition in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ 0 or – 0</td>
<td>Corner with contour definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 0.001 to – 99999.999</td>
<td>Chamfer with contour definition in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 0.0001 to – 3999.9999</td>
<td>Chamfer with contour definition in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ 0.001 to + 99999.999</td>
<td>Radius with circular interpolation in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ 0.0001 to + 3999.9999</td>
<td>Radius with circular interpolation in inches</td>
</tr>
<tr>
<td>c</td>
<td>C</td>
<td>4)</td>
<td>± 0.001 to + 99999.999</td>
<td>Auxiliary/rotary position information in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ± 3999.9999</td>
<td>Auxiliary/rotary position information in inch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.001° to ± 99999.999°</td>
<td>Position information in degrees</td>
</tr>
<tr>
<td>d</td>
<td>D</td>
<td></td>
<td>1 to 99</td>
<td>Tool offset selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Tool offset cancellation</td>
</tr>
<tr>
<td>f</td>
<td>F</td>
<td>4)</td>
<td>0.01 to 45000</td>
<td>Feedrate in mm/min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.1 to 1770.000</td>
<td>Feedrate in inches/min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to 50.000</td>
<td>Feedrate in mm/rev</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0001 to 2.0000</td>
<td>Feedrate in inches/rev</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to 16.000</td>
<td>Thread, lead increase or decrease in mm/rev</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0001 to 6.0000</td>
<td>Thread, lead increase or decrease in inches/rev</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to 99999.999</td>
<td>Dwell in sec.</td>
</tr>
<tr>
<td>h</td>
<td>H</td>
<td></td>
<td>1 to 999</td>
<td>Auxiliary functions</td>
</tr>
<tr>
<td>i</td>
<td>I</td>
<td>4)</td>
<td>± 0.001 to ± 99999.999</td>
<td>Interpolation parameters for X axis in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ± 3999.9999</td>
<td>Interpolation parameters for X axis in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to 400.000</td>
<td>Thread lead in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0001 to 16.000</td>
<td>Thread lead in inches</td>
</tr>
<tr>
<td>j</td>
<td>J</td>
<td>4)</td>
<td>± 0.001 to ± 99999.999</td>
<td>Interpolation parameters for Z axis in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ± 3999.9999</td>
<td>Interpolation parameters for Z axis in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to 400.000</td>
<td>Thread lead in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0001 to 16.000</td>
<td>Thread lead in inches</td>
</tr>
<tr>
<td>k</td>
<td>K</td>
<td>4)</td>
<td>± 0.001 to ± 99999.999</td>
<td>Interpolation parameters for Z axis in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>± 0.0001 to ± 3999.9999</td>
<td>Interpolation parameters for Z axis in inches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001 to 400.000</td>
<td>Thread lead in mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0001 to 16.000</td>
<td>Thread lead in inches</td>
</tr>
<tr>
<td>l</td>
<td>L</td>
<td></td>
<td>1 to 999</td>
<td>Subroutine number</td>
</tr>
<tr>
<td>p</td>
<td>P</td>
<td></td>
<td>1 to 99</td>
<td>Number of subroutine passes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ratio P = working diameter/unit diameter with cylindrical interpolation</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>R</td>
<td></td>
<td>0 to 49</td>
<td>Transfer parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50 to 99</td>
<td>Arithmetic parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>110 to 699</td>
<td>Channel-dependent declared parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>700 to 999</td>
<td>Channel-independent declared parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Central parameters</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>S</td>
<td></td>
<td>1 to 16000</td>
<td>Spindle speed in rev/min or 0.1 rev/min and constant cutting speed in m/min or 0.1 m/min or ft/ min or 0.1 ft/ min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 to 16000</td>
<td>Spindle speed limitation in rev/min or 0.1 rev/min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5 to 359.5</td>
<td>Spindle speed (M19), distance from zero mark of encoder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.1 to 99.9</td>
<td>Dwell in revolutions/min</td>
</tr>
<tr>
<td>t</td>
<td>T</td>
<td></td>
<td>1 to 9999</td>
<td>Tool number</td>
</tr>
</tbody>
</table>

1) No other functions may be written in this block
3) Other addresses selectable (A, B, C, E, U, V, W)
# Active block by block, all others modal
- Initial setting (after Reset, M02/M30, after powering up controller)
4) The range is set via machine data. Refer to the machine manufacturer for the valid range of values.
<table>
<thead>
<tr>
<th>Group</th>
<th>EIA</th>
<th>ISO</th>
<th>Code</th>
<th>Function and significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>m</td>
<td>M</td>
<td>00 #</td>
<td>Programmed stop, unconditional</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>Programmed stop, conditional</td>
</tr>
<tr>
<td>M2</td>
<td>m</td>
<td>M</td>
<td>02</td>
<td>Program end, in last program block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17</td>
<td>Subroutine end, in last subroutine block, without stop in repeat passes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>Program end, in last program block</td>
</tr>
<tr>
<td>M3</td>
<td>m</td>
<td>M</td>
<td>03</td>
<td>Direction of spindle rotation clockwise</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04</td>
<td>Direction of spindle rotation counter-clockwise</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>05+</td>
<td>Oriented spindle stop, angle in degrees under address S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>Spindle stop, non-oriented</td>
</tr>
<tr>
<td>M4</td>
<td>m</td>
<td>M</td>
<td>36</td>
<td>Feedrate programmed under F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>37</td>
<td>Feedrate in mm/min or mm/rev, 1:100 reduction ratio with G33</td>
</tr>
<tr>
<td>M5</td>
<td>m</td>
<td>M</td>
<td>0</td>
<td>Miscellaneous functions, freely assignable except for groups M1 to M4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>70...</td>
<td>Process-oriented (flying) measuring cycle</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>91/92</td>
<td>Retraction cycles for tool changes</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>93</td>
<td>Grooving cycle</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>95</td>
<td>Cutting cycle against contour, paravial</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>97</td>
<td>Thread cutting cycle with flank infeed</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>98</td>
<td>Deep drilling cycle</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>99</td>
<td>Chaining of threads (Four-point threading cycle)</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>L</td>
<td>999</td>
<td>Clear buffer memory</td>
<td></td>
</tr>
<tr>
<td>@</td>
<td>@</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>=</td>
<td></td>
<td></td>
<td>Separator, mandatory in address extensions, e.g. R35 = 123.5</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>Addition for parameters</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td>Subtraction for parameters</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>Multiplication for parameters</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td></td>
<td></td>
<td>Division for parameters</td>
</tr>
<tr>
<td>5-4-2</td>
<td>(</td>
<td></td>
<td></td>
<td>Start of comments</td>
</tr>
<tr>
<td>7-4-2</td>
<td>)</td>
<td></td>
<td></td>
<td>End of blocks</td>
</tr>
<tr>
<td>EOB</td>
<td>LF</td>
<td></td>
<td></td>
<td>Block end</td>
</tr>
</tbody>
</table>

2) **Punched tracks**

# Active block by block, all others modal

* Initial setting (after Reset, M02/M30, after powering up controller)
 Siemens AG  
AUT V250  
P.O. Box 3180  
D-91050 Erlangen  
Federal Republic of Germany

<table>
<thead>
<tr>
<th>From:</th>
<th>Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Should you come across any printing errors when reading this publication, please notify us on this sheet. Suggestions for improvement are also welcome.</td>
</tr>
<tr>
<td>Company/Dept.</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>Telephone</td>
<td>/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suggestions and/or corrections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrections</td>
</tr>
<tr>
<td>For Publication/Manual:</td>
</tr>
<tr>
<td>SINUMERIK 820T, GA3, SW3</td>
</tr>
<tr>
<td>Operating and Programming</td>
</tr>
<tr>
<td>User Documentation</td>
</tr>
<tr>
<td>User's Guide</td>
</tr>
<tr>
<td>Order No.: 6ZB5 410-0ER02-0BA2</td>
</tr>
<tr>
<td>Edition: January 1993</td>
</tr>
</tbody>
</table>