3. Program configuration

3. Program configuration

3.1 Program

Each program begins with a program number with a maximum of 7 digits (DOS - PC) and/or 9 digits. The program consists of individual blocks.

3.2 Block

Each block is identified by a block number with a maximum of 10digits. A block is made up of various words, which contain all instructions for an operation. The block length is variable (maximum 200 signs).
Blocks are identified in the program with ascending block numbers. This determines the sequence when the program is running and means that identical block numbers are not acceptable.

3.3 Word

A word consists of an address and a data section. The word length is variable. All words which can be contained in a block are included in the following table.

3.4 Mathematicals expression

A number can be also replaced by a mathematical expression in round brackets.

For example: ((110 + P20) / 3)

3.5 Block functions

Word	Address	No. of words per block	Data digits	Dimensional unit	Effect	Description in section
Block number	N	1	10 #		block by block	
Feed	F	1	x ~	mm/min	modal	3.
Spindle speed	S	8	x ~	1/min	modal	3.
Traverse conditions	G	8 *	10 #		modal/bl.-by-bl.	4.
Circle centre point	I / J / K	1	x ~		modal	4.
Circle radius	R	1	x ~		modal	4.
Cycle	G	8 *	10 #		block-by-block	5.
Additonal functions	M	8	3 #		modal/bl.-by-bl.	6.
Tool	T	1	10 #		modal	7.
Parameters	P / q	x	x		modal/bl.-by-bl.	8.

Key: ~ Floating point
Integer

* Traverse condition and cycle together 8 per block

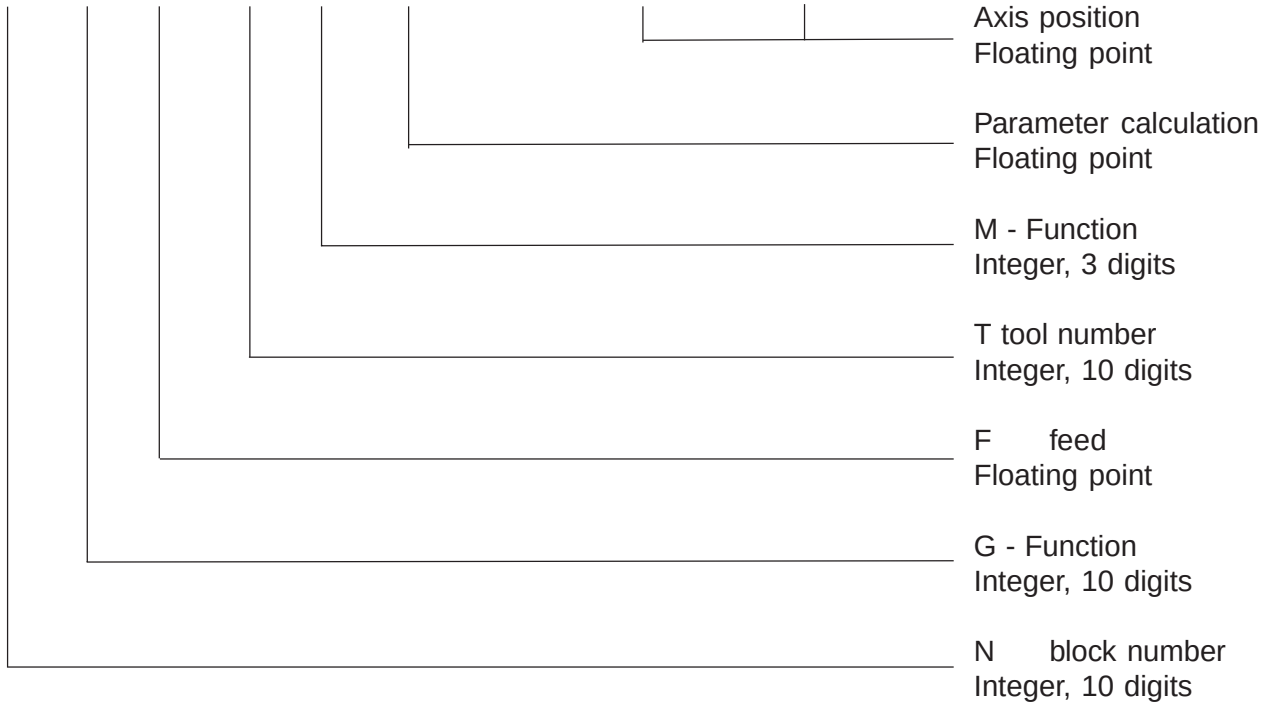
All other letters can be used for axis terms.

3.6 Syntax

Fundamental construction of a NC block:

Example :

N10 G01 F100 T03 M100 P30:(P20+35)*3 X200.0000 Y400.550 { comment }



Block length = 200 signs

3.6 Syntax (continued)

General functions

/ Block skip

\ Chain blocks, i.e. several blocks are joined to one NC block.

() Bracket functions, mathematical expressions or comment

{} Bracket comment

: Allocation

$ Signal for hexadecimal numbers
Hexadezimale expressions must be completed with the separating sign(;)!
e.g.: N10 G01 F1000 P500 : $1AF; X: P500

Axes

Axes can be marked:

- with a letter — X, Y, Z, U, V, W . . .
  X100, Z33

- with a letter and index 1 to 8 — X1, X2, X3 . . .
  X2:100, Z1:33 . . .

Clear parameter content

P500: - - clears the content of P500

3.6 syntax (continued)

Comparison operators

		Example :
=	Equal	P500=110.50 (Skip to block 50, if the content of P500 = 110)
>	Larger	
<	Smaller	
<>	Unequal	
>= =>	larger equal	
<= >=	smaller equal	

If the skip - condition is fulfilled, it is skipped to the indicated block number.

Calculation operators

		Example .:
*	Multiplication	P500:P200*5
+	Addition	P500:P200+P201
-	Subtraction	P500:P200-1
/ %	Division	P500:P200 / 2
| mod	Modulus	P500:P200|10
		The trigonometrical functions use degree !
sin cos	tan	P500:sin(90)
asin acos	atan	P500:acos(P10)
or and	not	Bit operations P500:P500 or $100;
sqr	Square root	
int	Integer value	
intr	Rounding on integer	
abs	Amount	
ln	Logarithm with basis e	
lay	Logarithm with basis 10	
exp	Exponent	
del	test if cleared	P500:del(510) Feedback value : 0 parameter not cleared 1 parameter cleared

Functions are always written in lower-case letters!

3.6 syntax (continued)

Definition of a tool radius t

If no tool administration is existing in the system, a temporary correciton radius can be defined with identification ‘t’. With this correction radius, the subsequently activated radius correction (G41/ 42) is working.

Example:

```
N10..
N20 t:0.5
N30 G1 X100 Y100 G42
N40..
```

When programming ‘t’
- the tool radius is written and
- the tool radius correction is deleted
  in the active data block (P8150 ...)

The quadrant of the ‘t’-correction is always 0!

3.6 Syntax (continued)

Syntax of symbolic variables

Symbolic variables always do start with the sign '_' (e.g.: _abs, _test5, ...), they may have a maximum length of 30 signs. Capitel and small letters are allowed, but the meaning of _karl may not be the same as _Karl! (case-sensitive).

Binding a symbol to a parameter

_wegx ::= 500;

According to this definition, _wegx is in place of P500, i.e. both expressions are equivalent according to the above mentioned example (_wegx <—> P500).

P500 : 10
_wegx:=10

Indexing is allowed at parameter variables.
Example

_wegx(0) := 10; (d.h. P500:10)
_wegx(1) := 11; (d.h. P501:11)

Internal variables

_wegxy := 500;

If a value is assigned to a variable, which is not 'bound' to a parameter, this variable is allocated as internal variable. That means that values can be stored without using a parameter.
Variables that are not initialized have the value 0.
Internal variable do only exist as long as the NC program is active.
Internal variables can not be displayed directly (e.g. at a program test).

Example

n10 _test1::=12 (Binding to P12)
N20 _test2:=10 (internal variable)
N30 G00 X:_test1 Y:_test2
N40 ...

3.6 Syntax (continued)

Text output of NC programs

Arbitrary texts from NC programs can be displayed in the message line. When switching back to MANUAL, texts that are eventually standing in line, are deleted.

Delete syntax messages

N10 !
N10 M33 P1:23 !

The identification '!' may also be programmed with other NC block elements in the same NC block. However, '!' has to be the last sign of the block!

Displaying messages with predefined colours

N10 ! this is a message text white letters on blue bottom
N20 !0, this is a message text black letters on grey bottom
N30 !1, this is a message text white letters on blue bottom

Displaying messages with colour selection that can be defined freely.

N40 !S8E, this is a message text (yellow on black)

N40 !code, this is a message text code = HF + VF

HF	(background colour)
80	black
90	blue
A0	green
B0	turquoise
C0	red
D0	magenta
E0	brown
F0	light grey

VF	(Forefront colour)
0	black
1	blue
2	green
3	turquoise
4	red
5	magenta
6	brown
7	light grey
8	dark grey
9	light blue
A	light green
B	light turquoise
C	light red
D	light magenta
E	yellow
F	white

3.6 Syntax (continued)

Working sequence of the block interpreter

1. Parameter calculations, Parameter allocations
   are executed in the sequence programmed in the NC block.

2. Parameter skips
   are executed in the sequence programmed in the NC block.

3. M - Function - Skips
   are executed in the sequence programmed in the NC block.

4. Sub-program call M28

The sequence of the block elements when dispatching at PLC (real time)

1. Block number

2. Parameter (real time parameter)

3. S-value

4. T-value

5. M-function before traverses / after traverses

3.6 Syntax (continued)

Enlarged syntax

The NC interpreter offers with System-Calls (sc) further possibilities, to shift the interpreter -mode or to trigger functions.

Syntax sc: n

Function numbers

0 Activating of interpreter-mode 0, standard mode.
- Is always preset at NC program start.

- Each block produces a block end. When switching on the interpreter to the next NC block a block change command results, with which the NC block informations are transmitted to the following modules.

1 Activating of interpreter-mode 1, supervision mode.
- In mode 1 the block change command is suppressed. When switching the interpreters to the next NC block no block change command results.
  Because the analyzed NC block elements are only transmitted to the following modules with block change, they remain now for the time being in the block interpreter.
  (This does not count for the additonal functions.)

- When switching back to mode 0 a block change command is given.

2 Activating the interpreter mode 2
- supervision mode at M26

- as mode 1, however, when switching the mode, the system waits until it gets a feedback from the previous NC block. E.g. if you want to make some calculations or supervising loops while an axis is moving, the calculation and/or test loop will start at the beginning of this movement (and the pipeline of the NC control is previously deleted).

- A block change command is sent when switching back to mode 0.

3.6 Syntax (continued)

Example of a supervision loop

While N10 is processed, supervision functions can be perceived in the loop (N30 . .. N50).

```
N10     G01  F100  X1000
N20     sc :1                 changeover in mode 1
N30     P500>P501.140         skip, if supervision loop should be left.
N40     . ..
N50     M23.30                skip to the beginning of the loop
N140    . ..
N150    sc :0                 end  of the supervision loop
N200    G00  X200
```

100 Triggering of a block change command
- In interpreter mode 1 a block change can be forced herewith.

3.7 Feed

The feed with the address letter F is programmed in mm or inch according to the set unit of measurement.

G94, G95 and G99 determine the feed modification.

G94	Feed in mm/min
G95	Feed in mm/r
G99	Block end feedrate in mm/min

A programmed feed is effective modally and can only by overwritten with another feed.

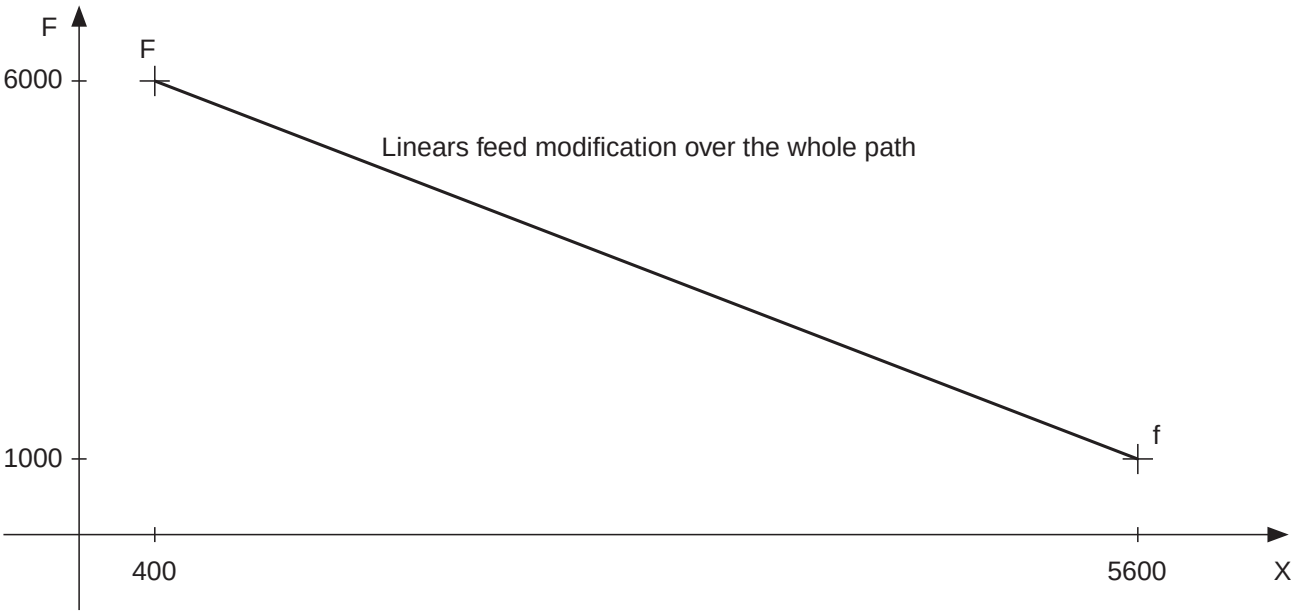
Through positioning in rapid traverse (G00) the previously stored effective feed is not cleared, but becomes effective again with G01, G02 and G03.

With the feed-override-potentiometer the programmed feed can be changed in the area of 0 to 120%.

F	path feed
f	final feed, i.e. path feed, which are achieved at the end of block.

Example:

N50	X400				
N60	G1	F6000	f1000	X5600	



3.8 Spindle speed

The spindle speed is programmed with the address letter S. With index 1 to 8 several spindles can be programmed.

Example: S1000

S1:1000
:
S8:8000

G96 and G97 determine the speed modifications.

G96 constant cutting speed in mm/min
G97 number of revolutions in 1/min

The spindle speed is effective modally and can only be overwritten through another spindle speed.

With the spindle-override-potentiometer the programmed spindle speed can be changed in the area of 0 to 120%.

3.9 Input / output (I/O)

DOS data format

Structure of a NC program file

Blank line (CR, LF)

Identification P/Z with program number (program number with max. 9 digits

NC block beginning with N or /N
...
...
...
NC block

Program end sign (#)

EOF-sign (default : character 04)

Blank line (CR, LF)

Example:
File name : P123456

P123456
N10 G0 X0 Y0 Z0
N20 F100 G1 X100
N30 M30
#

3.9 Input / output

Structure of a parameter file

Blank line	(CR, LF)	
Identification D	(at identification D:	parameter status is not overtaken exception: If mantissa programs,
	(at identification D+:	parameter status is overtaken Exception: If mantissa programs, however in the parameter status the loading bit (byte 1, bit 1) is not set, than the bit ‘ parameter loaded ‘ is set! example: D+ K1 P1: 123 S:$32000100 in this case the status becomes to S:$32000101!

q parameter number : parameter content [S: parameter status] * * [ ] optional

...
...
...

Program end sign	(#)
EOF-sign	(default : character 04)
Blank line	(CR, LF)

3.9 Input / output

Structure of a parameter file

Example:

Filename : D123

```
D
q     0:  —                 S:$00000000
q     1:          8         S:$00000001
q     2:          2
q     3:      30000
q     4:  —
q     5:  —                 S:$00000000
#
```

or

```
D
K1:P  10:         1         S:$00000009
K1:P  11:       100         S:$0000000D
K1:P  12:       200         S:$00000001
K1:P  13:         5
K1:P  14:   —
K1:P  15:         2
#
```

Extensions

starting from version 080 :

With identifier D+ knows the parameter status with the function ‘ SET ‘, or with which old parameter status with the function ‘ OR ‘ is set.

Example: Parameter status with function ‘ SET ‘:
K1 P1: 123 S:$32000101

Parameter status with function ‘ OR ‘:
K1 P1: 123 S|$32000101