Functional description | EN

TF5200 | TwinCAT 3 CNC

Block search
Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT technology is patent protected, in particular by the following applications and patents:


with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilisation of this document as well as the communication of its contents to others without express authorisation are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
## General and safety instructions

### Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

#### Icons in explanatory text

1. Indicates an action.

   ![Icon](image)

   Indicates an action statement.

<table>
<thead>
<tr>
<th>DANGER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acute danger to life!</strong></td>
</tr>
<tr>
<td>If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Personal injury and damage to machines!</strong></td>
</tr>
<tr>
<td>If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Restriction or error</strong></td>
</tr>
<tr>
<td>This icon describes restrictions or warns of errors.</td>
</tr>
</tbody>
</table>

| ![Icon](image) |
| **Tips and other notes** |
| This icon indicates information to assist in general understanding or to provide additional information. |

### General example

Example that clarifies the text.

### NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

| ![Icon](image) |
| **Specific version information** |
| Optional or restricted function. The availability of this function depends on the configuration and the scope of the version. |
Table of contents

Notes on the documentation ........................................................................................................... 3

General and safety instructions ....................................................................................................... 4

1 Overview ...................................................................................................................................... 8

2 Description .................................................................................................................................... 9

3 Block search types ...................................................................................................................... 10
  3.1 Type 1: Continuation position defined by file offset .............................................................. 10
  3.2 Type 3: Continuation position defined by block counter ....................................................... 12
  3.3 Type 4: Continuation position defined by block number ....................................................... 13
  3.4 Type 5: Block search at program end ..................................................................................... 15
  3.5 All types: Block search with additional breakpoint .............................................................. 16
    3.5.1 Type 6: Breakpoint without block search ......................................................................... 17
  3.6 All types: Block search as of a specific program position (file offset) ................................... 18

4 General parameters ..................................................................................................................... 19
  4.1 Continuation position within a block .................................................................................... 19
    4.1.1 Covered distance in current block in per thousand.......................................................... 19
    4.1.2 Covered distance from program start ........................................................................... 21
  4.2 Restart to the contour after block search ............................................................................. 27
    4.2.1 Manual restart during block search .............................................................................. 30
    4.2.2 Track the C axis with block search (#CAX TRACK) .................................................. 32
    4.2.3 Path-dependent angle compensation at block search (#VECTOR OFFSET) .................. 34
  4.3 Backward motion after block search ..................................................................................... 34

5 Block search interface and parameters ...................................................................................... 36
  5.1 General block search parameters ......................................................................................... 36
    5.1.1 Covered distance .......................................................................................................... 37
  5.2 Block search type 1 parameters: File start/end position ..................................................... 39
    5.2.1 Start position ............................................................................................................... 39
    5.2.2 End position ................................................................................................................. 41
  5.3 Block search type 3 parameters: Block counter .................................................................. 43
  5.4 Block search type 4 parameters: Block number ................................................................. 43
  5.5 Status data: Access via HLI .................................................................................................. 44
  5.6 Status data: Access via CNC objects: .................................................................................. 48

6 Locking program areas for block search .................................................................................... 49

7 On/off handshake with PLC ...................................................................................................... 51

8 Known restrictions ....................................................................................................................... 53

9 Examples ...................................................................................................................................... 55
  9.1 Block search type 4 ............................................................................................................... 55
    9.1.1 Specify block number and pass counter ........................................................................ 55
    9.1.2 Specify the block number and covered distance in the block ...................................... 56
  9.2 Block search type 3 ............................................................................................................... 57
    9.2.1 Specify block counter ..................................................................................................... 57

10 Exceptions and errors ............................................................................................................... 58
11 Support and Service........................................................................................................................................... 59

Index ........................................................................................................................................................................... 60
List of figures

Figure 1  Interactions and interfaces with block search ................................................................. 8
Figure 2  Block search position by start and end ........................................................................... 11
Figure 3  Continuation position by block counter ........................................................................ 12
Figure 4  Continuation position by block number ........................................................................ 14
Figure 5  Continuation position by block number and pass counter .......................................... 14
Figure 6  Continuation position at program end ............................................................................ 15
Figure 7  Block search and additional breakpoint ......................................................................... 16
Figure 8  Additional breakpoint ..................................................................................................... 17
Figure 9  Jump point with block search ......................................................................................... 18
Figure 10 Continuation position with current block split by per thousand .................................. 19
Figure 11 Per thousand display with two inserted polynomial blocks .......................................... 20
Figure 12 Per thousand display with one inserted polynomial block ........................................... 20
Figure 13 Distance from program start ......................................................................................... 21
Figure 14 Influence of starting movement is prevented by NC commands .................................. 22
Figure 15 Continuation position with current block split by distance from program start .......... 23
Figure 16 Search for continuation position by distance from program start over several blocks ... 24
Figure 17 G74, G100, #FLUSH WAIT during search for continuation position ....................... 25
Figure 18 Continuation position by distance from program start before current block ............... 26
Figure 19 Use of different tool radii ............................................................................................... 27
Figure 20 Use of block search to restore the program context ...................................................... 28
Figure 21 Manual restart to the contour ......................................................................................... 29
Figure 22 Restart to the contour ................................................................................................... 30
Figure 23 Sequence of block search and completion of manual restart ...................................... 31
Figure 24 Example of continuous alignment of the C axis to the contour ................................... 32
Figure 25 Backward motion after block search ............................................................................ 35
Figure 26 Block search states ....................................................................................................... 45
Figure 27 Interaction between BOOLEAN-LC control unit and PLC ........................................ 52

TF5200 | TwinCAT 3 CNC
Block search
Version: 1.01
7
1 Overview

Task

The operator can start machining at what is called the continuation position at any point in the program. After a program is interrupted (e.g. tool breakage), this is a quick method to reactivate machining at the point of interruption.

The continuation position can be defined using a number of different block search types (file offset, block counter, block number, etc.).

It is imperative to restore the entire program context at the starting point specified here (program parameters, axis positions, etc.). This is ensured by processing the program up to this continuation position without any axis motion (simulation). Technology functions are signalled to the PLC, even during the simulation. All the vital machine functions for the machining process are then activated at the continuation position (e.g. coolant, velocity).

When the program reaches the continuation position, the axes can be moved to their current positions at this program position either manually or automatically.

The operator can then start the continued execution of the program.

Figure 1: Interactions and interfaces with block search

- Block search type 2 is no longer available.
2 Description

Program start in block search mode

In block search, a specific program point, which is called the continuation position, is controlled without real axis motions. After this program point is reached, the axes are positioned at the contour either manually or automatically and the machining process is continued.

Continuation position

The continuation position is the position at which processing the NC program ends in block search and real machining starts on the workpiece. Normally this is the interrupt position of a previous machining process. The reason for an interrupt may be to measure a workpiece or a tool breakage.

Continue in a motion block

The continuation position may also be located within an NC block. For this reason, one option with all block search types is to define the covered distance in the NC block as of which actual machining should continue.

Block search must be selected on the control panel (HMI) or via the PLC before the main program starts.

Block search selection is modal, i.e. it remains active after the NC program ends. Block search must then be deselected explicitly, e.g. via the HMI.

Simulation and axis motions

No physical axis motion takes place in block search mode up to the continuation position. However, the NC program is completely decoded to produce the required program context at the continuation position. Since no real interpolation takes place, the continuation position is usually reached very much faster than with normal program processing.

Simulation and technology functions

In block search mode, the PLC receives all technology functions as in normal mode so that machine functions can be switched correctly. These functions must also be acknowledged.

The PLC synchronises the selection/deselection of block search mode. This may result in the specific handling of certain technology commands if this is supported by the PLC (group handling, activating certain functions before block search is deselected).

This extension is available as of Builds V2.11.2018.09, V2.11.2804.10 and V3.1.3030.2.

Move to continuation position

When the continuation position is reached in block search mode, ensure that the axes were moved from their current actual positions to the restored command positions before actual machining starts (Restart to the contour)

At the same time, the spindles must reach their last commanded speeds.

Real mode

After switching over to real machining, the NC program continues as if it had been started without block search mode. A number of methods are available for restart to the contour (see Restart to the contour after block search [27]).
3 Block search types

3.1 Type 1: Continuation position defined by file offset

Block search type 1:
Continuation position and end position by file offset

This block search type defines a processing range by specifying a continuation/end position (start/end mark) using file offset. Processing starts at the continuation point and ends after the end position is reached. The NC program is then ended immediately. If no end position is specified, processing continues until NC program end M30.

File offset defines the continuation/end position as the distance to file start of the NC program.

It is the operator’s responsibility to define the file offset. File offset is always determined in relation to the start of the NC row of each NC program (main program (HP), global subroutine (UP)).

File offset and other data to determine the continuation/end position are transferred as parameters when block search is selected.

Continuation position

Parameter:
• file offset,
• file name,
• identifier whether file is in the HP-(0) or UP path (1),
• pass counter (optional),
• covered distance within the block (optional)

end position (optional)

Parameter:
• file offset,
• file name,
• identifier whether file is in the HP-(0) or UP path (1),
• pass counter (optional)
Start in the main program in 1st pass at file offset 100 at block start, end at file offset 239

Figure 2: Block search position by start and end
3.2 Type 3: Continuation position defined by block counter

Block search type 3:

Continuation position by block counter

At program start the block counter (controller-internal) is incremented by 1 for each NC row decoded. The block counter is also incremented continuously for comment lines, empty lines, in loops and during subroutine calls. The block counter is displayed to the PLC or the HMI during normal program execution. To determine the continuation position, the block counter is transferred as one of the parameters when block search is selected.

Parameter:
- block counter,
- covered distance within the block (optional)

With block counter

Start at block counter 12

![Diagram showing block counter and examples of code.](image)

Figure 3: Continuation position by block counter
3.3 Type 4: Continuation position defined by block number

Block search type 4:

Continuation position by block number and program name

The block number is the number (N word) of an NC row in the NC program. To generate the NC program, the operator or the system is responsible for providing every row which is relevant later, even only once, with a unique block number. However, block number ambiguities may occur due to local and global subroutines. Therefore, to determine the exact continuation position, the program name (%...) can also be specified as an option to conduct a search for the block number.

Parameter:

- block number,
- program name (%...) (optional),
- covered distance within the block (optional)

*The optional specification of a program name is available as of Build V3.01.3000.00.*

Continuation position by block number and pass counter

Since the block number is passed several times, e.g. in loops, it is not always unique on its own. In this case, machining start may be optionally triggered by the additional specification of a pass counter. The program name (%...) can also be specified as an additional option here.

Parameter:

- block number,
- pass counter,
- program name (%...) (optional),
- covered distance within the block (optional)

*It is the user’s responsibility to supply the pass counter.*

With block number

Start at block number 60
**Block search types**

### TF5200 | TwinCAT 3 CNC

**Block search**

14

Version: 1.01

---

**Figure 4:** Continuation position by block number

**With block number and pass counter**

Start at block number 60 in 5th pass

- Start of execution
  - N10
  - N20
  - N30
  - N40
  - N50
  - N60
  - N70
  - N80
  - N90
  - N100
  - N110
  - N120
  - N130
  - ...

- Axis movement

- No axis movement

---

**Figure 5:** Continuation position by block number and pass counter

- Start of execution if P1 = 5
  - N10 P1 = 0
  - N20
  - N30
  - N40
  - N50
- $WHILE$ P1 < 8
- N60 G91 X10
- N70
- N80
- N90
- $ENDWHILE$
- N100
- N110
- N120
- N130
- ...

- Axis movement

- No axis movement

---

![Diagram](image-url)
3.4 Type 5: Block search at program end

Block search type 5:
Continuation position at program end

This special block search type is used in particular in job planning on a simulation system for a rapid test of NC programs. The continuation position is placed implicitly at program end (M17, M30). The NC program is only decoded but no axis positions are interpolated. This permits long NC program to run through quickly.

When program end is reached, the complete NC program is terminated in block search mode.

Distinction from “Dry Run”

In Dry Run [FCT-C17] mode the processing speed is identical to execution on the machine. Axis positions are interpolated but axis motions are not executed. This operation mode is only useful if it is executed directly on the machine controller.

Block search at program end M30

```
N10
N20
N30
N40
N50 L...
N60
N70
N80
N90 LL..
N100
N110
N120
N130
...
M30
```

Figure 6: Continuation position at program end
3.5 All types: Block search with additional breakpoint

Set a breakpoint with block search

An automatic stop is inserted by specifying a breakpoint at the distance from program start (cf. explicitly programmed M0). This permits the automatic instrumentation of an NC program with an M0.

The breakpoint can be specified in addition to the continuation position of the block search. The breakpoint must be placed after the continuation position.

During block search, the NC program is processed up to the specified continuation position without axis motion (green section). The axes are then moved for real (grey and white section).

![Diagram of block search and additional breakpoint]

**Definition of breakpoint**

The breakpoint is defined in a similar way to the continuation position of block search before the start of an NC program by the distance from program start by

```plaintext
mc_cmd_bs_breakpoint_position_w
```

(see section Additional breakpoint).

**Interaction with PLC**

The stop caused by the breakpoint is displayed at the breakpoint (see section HLI: Stop conditions). The block search state then signals “Wait for continue motion” (see section HLI: Block search state). This is displayed until the PLC requests release for continuation of machining (see [HLI/Continue motion]).

The breakpoint is only evaluated the first time it is reached in forward direction. If the contour is then moved in backward/forward direction, no stop is executed at the breakpoint.
3.5.1 Type 6: Breakpoint without block search

Set a breakpoint without block search

If a breakpoint is to be set without a previous block search, this can be done by specifying the block search type SIMULATION = 6.

In this case the program is processed normally with an additional breakpoint (grey and white section).

Figure 8: Additional breakpoint
3.6 All types: Block search as of a specific program position (file offset)

Set a jump point with block search

The NC program can be started with a file offset in combination with any block search type. The file offset defines a jump to a known position in the NC program.

The program part before the jump point is ignored. Evaluation starts at the jump point as for a program shortened by file offset.

The jump point can be specified in addition to the continuation position of the block search. The jump point must be **before** the continuation position.

The full technology scope must be reproduced at the jump point so that machining can continue. The NC program then runs as usual up to a continuation position in the commanded block search type.

This method saves time in the block search with large NC programs.

During block search, the NC program is processed up to the specified continuation position (green section) without axis motion after the jump point (grey section). The axes are then moved for real (white section).

**Definition of jump point**

The jump point is defined in a similar way to the continuation position of block search before the start of an NC program by

```
mc_command_file_offset_w
```

(see section Program start at file offset).

File offset to define a jump point can also be used without block search. Processing then starts directly at the jump point as for a program shortened by file offset.
4  General parameters

4.1  Continuation position within a block

Position within motion block

Normally block search is disabled between two NC rows. However, if this is a motion block, it may also be required to specify the continuation position within the motion block with greater precision. Therefore, there is an option for motion blocks to define a path distance within the block in addition to the block specification. This path distance can be specified in two ways:

Specify the covered distance in the block in per thousand referred to the current distance length of the block (see section Covered distance in current block in per thousand[11]).

Specify the covered distance from program start or from the last #DISTANCE PROG START CLEAR (see section Covered distance from program start).

4.1.1  Covered distance in current block in per thousand

Per thousand

The position within a motion block is defined in per thousand.

The per thousand display can be read as a display datum on the PLC interface (see Chapter HLI: Covered distance in the block – per thousand) when the current motion is interrupted.

Alternatively, the per thousand display can be specified as a purely required value without being previously read.

! If the restart position is defined by the relative display in relation to the motion block (per thousand), this position changes – within the range of resolution accuracy – even for restart when tool radius compensation is active and another tool radius is not.
This means that, if tool radius compensation is active, a tool with a different radius can be replaced in block search.

Value range of per thousand display

The per thousand display of a block is normally within the range [0, 1000].

Figure 10: Continuation position with current block split by per thousand
General parameters

Figure 11: Per thousand display with two inserted polynomial blocks

If only one block is inserted by the CNC between two original blocks (e.g. when #HSC [OPMODE = 1]), its distance may be between [0, 2000]

Figure 12: Per thousand display with one inserted polynomial block
4.1.2 Covered distance from program start

Distance from program start

Every motion block can be identified by the covered distance from program start. The distance is displayed on the PLC interface during the process (see section HLI: Covered distance - path increments, section Covered distance - path increments). The distance is formed as the sum of the main axis motions of all previous motion blocks. If a motion block contains no main axis motion, the distance of the tracking axis that moves at its dynamic limit is added.

\[
\text{distance}_{\text{ProgStart}} = 0
\]

\[
\text{distance}_{N10} = \text{distance}_{\text{ProgStart}} + \text{Anfahrweg}
\]

\[
\text{distance}_{N20} = \text{distance}_{N10} + 10
\]

\[
\text{distance}_{N30} = \text{distance}_{N20} + 24,14
\]

\[
\text{distance}_{N40} = \text{distance}_{N30} + 114,14
\]

Figure 13: Distance from program start

To approach the identical position in the block search, the original contour may not be changed in the definition of position by distance. This means that, if tool radius compensation is active, an identical tool must be replaced in block search.

However, if starting is executed with different tool geometries, the “Distance from program start” changes.

Distance display, NC commands

To render this independent of the starting axis position, the distance display can be controlled by the following commands in the NC program.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DISTANCE PROG START ON</td>
<td>(modal)</td>
</tr>
<tr>
<td>#DISTANCE PROG START OFF</td>
<td>(modal)</td>
</tr>
<tr>
<td>#DISTANCE PROG START CLEAR</td>
<td>(non-modal)</td>
</tr>
</tbody>
</table>

ON Distance of following motion blocks is evaluated for display (default after program start).

OFF Distance of following motion blocks is not evaluated for display.

CLEAR The current distance is set to 0 (default at program start).
Figure 14: Influence of starting movement is prevented by NC commands

**#DISTANCE PROG START**

%example
N10 #DISTANCE PROG START OFF
N20 G01 G90 X0 Y0 Z0 F1000

N100 G92 X33 Y55 ; Offset -> Lage des Teils
N110 X0 Y0 Z0 ; Starting movement of geometry
N120 #DISTANCE PROG START ON
N130 X100 ; distance = [ 0, 100]
N140 Y100 ; distance = [ 100, 200]
N150 X0 ; distance = [ 200, 300]
N160 Y0 ; distance = [ 300, 400]

N170 #DISTANCE PROG START OFF
...
N200 G92 X600 Y700 ; Offset -> position of part
N210 X0 Y0 Z0 ; Starting movement of geometry
N220 #DISTANCE PROG START ON
N230 X100 ; distance = [ 400, 500]
N240 Y100 ; distance = [ 500, 600]
N250 X0 ; distance = [ 600, 700]
N260 Y0 ; distance = [ 700, 800]
N270 #DISTANCE PROG START OFF
M30

**Position in block by distance**

If a block is stopped or interrupted, the current distance from program start can be read on the PLC interface (see section HLI: Covered distance - path increments) and the continuation position can be specified more exactly by this distance display.

**Continuation position with current block split by distance from program start**

<table>
<thead>
<tr>
<th>Block search type</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block number</td>
<td>30</td>
</tr>
<tr>
<td>Distance since program start</td>
<td>16 mm</td>
</tr>
</tbody>
</table>

Continuation position is within block N30
Figure 15: Continuation position with current block split by distance from program start
Search for continuation position by distance from program start over several blocks

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search type</td>
<td>4</td>
</tr>
<tr>
<td>Block number</td>
<td>30</td>
</tr>
<tr>
<td>Covered distance</td>
<td>234.79 mm</td>
</tr>
<tr>
<td>Continuation position is</td>
<td>within block N50</td>
</tr>
</tbody>
</table>

Figure 16: Search for continuation position by distance from program start over several blocks

NOTE

A homing run G74 or a measurement run G100 are movements which are ended by an external signal. The signal is simulated during block search up to the specified block limit (Phase 1).

If block search is additionally extended by specifying the covered distance (Phase 2), no commands such as G74 or G100 may occur in this section of the NC program since they cannot be simulated there. However, if this type of command occurs, an error message is output.
G74, G100, #FLUSH WAIT during search for continuation position

Block search type: 4
Block number: 30
Covered distance: 495.12 mm

Continuation position is within block N100

The continuation position can only be moved by tracking, i.e. towards program end, beyond the specified block limit (solid green section).

Moving the continuation position to a position already skipped in block search is not possible. This is prevented and a warning is output.
Continuation position by distance from program start before current block

<table>
<thead>
<tr>
<th>Block search type</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block number</td>
<td>30</td>
</tr>
<tr>
<td>Covered distance</td>
<td>2 mm</td>
</tr>
<tr>
<td>Continuation position is within block N20</td>
<td></td>
</tr>
<tr>
<td>Warning is output</td>
<td></td>
</tr>
</tbody>
</table>

Figure 18: Continuation position by distance from program start before current block
4.2 Restart to the contour after block search

**Move back to command position**

Before normal machining can be resumed, the axes must be located back at the command positions of the NC program (restart to the contour).

This starting movement is executed either

- by an NC program specified by the operator or
- manually before start of block search or
- automatically in a straight line at the end of block search.

![Diagram showing tool change, move to tool change position, interrupt position at contour, return to contour, and restart position.](image-url)

**Figure 19: Use of different tool radii**

As the above figure shows, the covered distance in the current block must be specified in per thousand relative to the entire block length since the absolute block lengths are different for circular blocks with different tool radii after block search.

**Automatic restart to the contour**

The automatically generated motion is moved in rapid traverse (G00).
• Operator starts normal program at position 1.
• Program interrupt at position 2 in block N20.
• Axis is moved to position 3 and the tool may be replaced (tool radius may change).
• The program context is restored at the end of block search to movement to position 4. Automatic restart occurs in a straight line.
• The program can be continued with the new tool radius in block N20.

Figure 20: Use of block search to restore the program context

Parameterisation
Automatic restart = TRUE
mc_cmd_bs_auto_return_w

Manual restart to the contour
If restart to the contour is executed manually and the axes are not repositioned exactly on the contour, an offset occurs between the command positions of the NC program and the actual positions. Here, the operator can specify the maximum permitted three-dimensional offset.
**Parameterisation**

Maximum permitted offset after block search between command position and actual position

`mc_cmd_bs_deviation_max_w`

- Backward motion is not possible with an offset > 0.

If an offset is active, please note the following in connection with software limit switch monitoring or collision monitoring:

The offset between the command positions and the actual position may result in SLS errors or collision errors although the actual positions are still within the valid range and no collision has occurred. Similarly, an SLS overshoot can not be detected in path preparation, only in the position controller. In this case, a collision cannot be detected in path preparation.
4.2.1 Manual restart during block search

**JogOfPath**

Manual restart to the contour (see Phase (4) “Pre-return to contour”) by means of an NC program can also be executed by a JogOfPath channel. Switchover between restart channel (JogOfPath) and block search channel always takes place when the axes are at standstill (see FCT-C15 Jog of path).

Block search may take longer if the program is rather large. Manual restart using a JogOfPath channel has the advantage that it can still be executed after block search has already started. In this case, it must be completed by the PLC before automatic restart to the contour is enabled (see Phase (6)).

Restart using any NC program can be used to start the contour via any strategy. By contrast, automatic restart only takes places by direct linear motion.

![Diagram of Restart to the contour](image)

Figure 22: Restart to the contour

**Sequence**

1. Normal program start
2. Program interrupt and channel reset
3. Move to any position
4. Switchover to JogOfPath and start program in block search channel
5. Start manual start program in JogOfPath channel
6. Switchover to block search channel and enable restart by PLC
7. Automatic restart of residual path difference

Remark: As opposed to starting on a JogOfPath channel, manual restart in a block search channel can only take place before the block search starts.

Channel block search can be started even though the channel axes are still released to the JogOfPath channel (PLC sets HLI.channel.SuspendAxisOutput).

The flow diagram below shows a manual restart to the contour executed by a JogOfPath channel before the PLC enables START. Before the block search channel continues, channel output is switched back to the block search channel (HLI.BlockSearchChannel.SuspendOutput = FALSE, HLI.JogOfPathChannel.SuspendOutput=TRUE).

Figure 23: Sequence of block search and completion of manual restart
4.2.2 Track the C axis with block search (#CAX TRACK)

#CAX TRACK

If automatic tracking of the C axis (#CAX TRACK) is already active at the restart position of the NC program, the C axis position can be restored before automatic restart.

Figure 24: Example of continuous alignment of the C axis to the contour

The following sequence runs at restart:

1. Align the C axis according to the tangent at the restart position
2. Restart according to the block search position
3. Reactivate automatic C axis tracking
4. Wait for continue from operator/HMI/PLC
Block search to block N40 at C axis = 0°

In the example below, the C axis is first aligned in block 40 according to the tangent to C = 0 at restart.

```plaintext
%block-search-cax-track
N10 G00 G90 X0 Y0 Z0 C0
N20 X5 Y5 C45 ;straight line 45° to X axis, tracking axis ;C aligned parallel to the contour
N20 #CAXTRACK ON [ANGLIMIT 3, OFFSET 0] ;Activate axis ;tracking, limit angle 3°, ;Angular offset 0°
N30 X10 Y10 ;Primary motion block, C axis is ;already aligned
N40 X20 ;Angle to previous block: -45° > ;Limit angle -> Block is inserted: ;End position of C = 0
N60 X40 ;C axis angle 0°
N70 X30 ;C axis angle 180°
N80 Y0 ;C axis angle -90°
N90 #CAXTRACK OFF ;Deactivate axis tracking
M30
```

The C axis is only restarted if automatic restart to the contour is selected (see also mc_cmd_bs_auto_return_w).
4.2.3  Path-dependent angle compensation at block search
(#VECTOR OFFSET)

#VECTOR OFFSET

If path-dependent angle compensation (#VECTOR OFFSET) is active at the restart position of the NC
program, the compensation is produced separately at automatic restart (see also #CAX TRACK).

The following sequence runs at restart:
- (Align the C axis according to the tangent at the restart position)
- Align the angle vertically and tangentially to the contour (ANC, DRC)
- Restart according to the block search position
- Reactivate (automatic C axis tracking and) path-dependent angle compensation
- Wait for continue from operator/HMI/PLC

Block search at N40 at ANC/DRC angle offset
In the example below, the vector offset at restart to block N40 is first set according to the active setting of
block N30.

```plaintext
%block-search-vector-offset
N10 #VECTOR OFFSET ON [ DS=20 DE=15 ANC=2 DRC=3
          ANCE=0.5 DRCE=0.2 ]
N20 G01 G90 X100 Y0 F200
N30 #VECTOR OFFSET ON [ DS=1 DE=1 ANC=0.5 DRC=0.2 ]
N40 G03 X110 Y10 J10 F75
N50 #VECTOR OFFSET ON [ DS=15 DE=20 ANC=2 DRC=3
          ANCE=0.1 DRCE=0.1 ]
N60 G01 Y115
N70 G01 Y0
...
N100 #VECTOR OFFSET OFF ALL
M30
```

The vector offset is only restarted if automatic restart to the contour is selected (see also
mc_cmd_bs_auto_return_w).

4.3  Backward motion after block search

If a backward motion is executed after block search is used, the actually programmed NC program is
interpolated in backward direction as of the start position from the block search.
Figure 25: Backward motion after block search
5  Block search interface and parameters

Validity of block search parameters

The block search parameters (block search type, continuation position display) are defined before the start of the NC program. At program start, the current data is adopted in the start parameters. Any further change in block search parameters after program start has no more influence on the running program.

Availability on the HMI

Assigning parameter before program start avoids time-critical situations. The parameters are available on the HMI and access is made via the PLC by read/write requests with CNC objects.

5.1  General block search parameters

<table>
<thead>
<tr>
<th>Block search type</th>
<th>Description</th>
<th>Type</th>
<th>Value range</th>
<th>HMI elements</th>
<th>IndexOffset</th>
</tr>
</thead>
</table>
|                  | Defines the block search type. | UNS16 | 0 – none (default), 1 – file offset, 3 – block counter, 4 – block number, 5 – program end | mc_cmd_bs_type_w (write)  
mc_cmd_bs_type_r (read) | 0x49 (write)  
0x61 (read)  
(IndexGroup = 0x000201<ii> where <ii> = channel) |

Automatic restart

| Description | Defines that restart to the contour should be automatic. | BOOLEAN | TRUE, FALSE (default) | mc_cmd_bs_auto_return_w (write)  
mc_cmd_bs_auto_return_r (read) | 0x4B (write)  
0x63 (read)  
(IndexGroup = 0x000201<ii> where <ii> = channel) |

No stop restart

| Description | Specifies whether restart should occur directly after restart to the contour without any operator input. | BOOLEAN | TRUE, FALSE (default) | mc_cmd_bs_no_hold_at_restart_w (write)  
mc_cmd_bs_no_hold_at_restart_r (read) | 0x79 (write)  
0x7A (read)  
(IndexGroup = 0x000201<ii> where <ii> = channel) |

Maximum path deviation
### Description
Describes maximum deviation of the axes between actual position and continuation position when processing is resumed after block search. If restart to the contour is automatic, the maximum path deviation is not considered since reaching the exact continuation position is ensured.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value range</th>
<th>HMI elements</th>
<th>IndexOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uns32</td>
<td>([0, \text{MAX}_\text{UNS32}]) in 0.1 µm, default 0</td>
<td><code>mc_cmd_bs_deviation_max_w</code> (write) <code>mc_cmd_bs_deviation_max_r</code> (read)</td>
<td>0x4C (write) 0x64 (read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>((\text{IndexGroup} = 0x000201&lt;\text{ii}&gt; \text{ where } &lt;\text{ii}&gt; = \text{channel}))</td>
</tr>
</tbody>
</table>

### Additional breakpoint
Specifies an automatic breakpoint by specifying the distance to program start.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value range</th>
<th>HMI elements</th>
<th>IndexOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL64</td>
<td>([0.0, \text{MAX}_\text{REAL64}]) in 0.1 µm</td>
<td><code>mc_cmd_bs_breakpoint_position_w</code> (write) <code>mc_cmd_bs_breakpoint_position_r</code> (read)</td>
<td>0x7B (write) 0x7C (read)</td>
</tr>
<tr>
<td></td>
<td>Default 0.0, breakpoint inactive</td>
<td></td>
<td>((\text{IndexGroup} = 0x000201&lt;\text{ii}&gt; \text{ where } &lt;\text{ii}&gt; = \text{channel}))</td>
</tr>
</tbody>
</table>

### Program start as of file offset
The file offset defines a jump to a known position in the NC program. The program part before the jump point is not evaluated. Processing starts at the jump point as for a program shortened by file offset.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value range</th>
<th>HMI elements</th>
<th>IndexOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uns32</td>
<td>([0, \text{MAX}_\text{UNS32}]), default 0</td>
<td><code>mc_command_file_offset_w</code> (write)</td>
<td>0x11 (write)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>((\text{IndexGroup} = 0x2010&lt;\text{ii}&gt; \text{ where } &lt;\text{ii}&gt; = \text{channel}))</td>
</tr>
</tbody>
</table>

### 5.1.1 Covered distance

#### Covered distance in motion block in per thousand
Defines the covered distance in the NC block in *per thousand* at which point actual processing is to continue. The first part of the block in the block search is then executed without motion and only the remaining part is executed with moved axes.

<table>
<thead>
<tr>
<th>Type</th>
<th>Value range</th>
<th>HMI elements</th>
<th>IndexOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL64</td>
<td>([0.0, 1000.0]) , default 0.0</td>
<td><code>mc_cmd_bs_covered_distance_w</code> (write) <code>mc_cmd_bs_covered_distance_r</code> (read)</td>
<td>0x4A (write) 0x62 (read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>((\text{IndexGroup} = 0x000201&lt;\text{ii}&gt; \text{ where } &lt;\text{ii}&gt; = \text{channel}))</td>
</tr>
</tbody>
</table>
### Covered distance from program start - path increments

<table>
<thead>
<tr>
<th>Description</th>
<th>The current distance from program start or #DISTANCE PROG START CLEAR can be commanded over the following interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>REAL64</td>
</tr>
<tr>
<td>Value range</td>
<td>[ 0.0, MAX_MAX_REAL64 ] in 0.1 µm, default 0.0 = deactivated</td>
</tr>
<tr>
<td>HMI elements</td>
<td>mc_cmd_bs_distance_prog_start_w (write)</td>
</tr>
<tr>
<td></td>
<td>mc_cmd_bs_distance_prog_start_r (read)</td>
</tr>
<tr>
<td>IndexOffset</td>
<td>0x44 (write)</td>
</tr>
<tr>
<td></td>
<td>0x45 (read)</td>
</tr>
<tr>
<td>(IndexGroup = 0x000201&lt;ii&gt; where &lt;ii&gt; = channel)</td>
<td></td>
</tr>
</tbody>
</table>
5.2 Block search type 1 parameters: File start/end position

5.2.1 Start position

The following parameters can be accessed via CNC objects to define the start position.

- File offset
- File name
- Program path ID
- Pass counter

They are described below.

<table>
<thead>
<tr>
<th>File offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Value range</td>
</tr>
</tbody>
</table>
| HMI elements | mc_cmd_bs_pos_start_offset_w (write)  
 mc_cmd_bs_pos_start_offset_r (read) |
| IndexOffset | 0x50 (write)  
 0x68 (read)  
(IndexGroup = 0x000201<ii> where <ii> = channel) |

<table>
<thead>
<tr>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Value range</td>
</tr>
</tbody>
</table>
| HMI elements | mc_cmd_bs_pos_start_name_w (write)  
 mc_cmd_bs_pos_start_name_r (read) |
| IndexOffset | 0x4F (write)  
 0x67 (read)  
(IndexGroup = 0x000201<ii> where <ii> = channel) |

**NOTE**

The specified block search filename must be identical with the file started in automatic mode. It also contains any additional specified path. If an explicit path is specified at program start, the path must also be included in the block search when the filename is specified.

<table>
<thead>
<tr>
<th>Program path ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Value range</td>
</tr>
</tbody>
</table>
| HMI elements | mc_cmd_bs_pos_start_type_w (write)  
 mc_cmd_bs_pos_start_type_r (read) |
| IndexOffset  | 0x4E (write)  
 0x66 (read)  
(IndexGroup = 0x000201<ii> where <ii> = channel) |
### Pass counter start position

<table>
<thead>
<tr>
<th>Description</th>
<th>The pass counter determines the number of times the start position should be passed up to the final start, e.g. in loops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SGN16</td>
</tr>
<tr>
<td>Value range</td>
<td>[ 1; MAX_SGN16 ], default 1</td>
</tr>
<tr>
<td>HMI elements</td>
<td>[mc_cmd_bs_pos_start_count_w (write), mc_cmd_bs_pos_start_count_r (read)]</td>
</tr>
<tr>
<td>IndexOffset</td>
<td>0x51 (write)</td>
</tr>
<tr>
<td></td>
<td>0x69 (read)</td>
</tr>
<tr>
<td>(IndexGroup = 0x000201&lt;ii&gt; where &lt;ii&gt; = channel)</td>
<td></td>
</tr>
</tbody>
</table>
5.2.2 End position

The following parameters can be accessed via CNC objects to define the end position.

- File offset
- File name
- Program path ID
- Pass counter

They are described below.

### File offset

<table>
<thead>
<tr>
<th>Description</th>
<th>Defines the file offset at which point processing is to end.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SGN32</td>
</tr>
<tr>
<td>Value range</td>
<td>[0, MAX_SGN32]</td>
</tr>
<tr>
<td>HMI elements</td>
<td>mc_cmd_bs_pos_end_offset_w (write)</td>
</tr>
<tr>
<td></td>
<td>mc_cmd_bs_pos_end_offset_r (read)</td>
</tr>
<tr>
<td>IndexOffset</td>
<td>0x54 (write)</td>
</tr>
<tr>
<td></td>
<td>0x6C (read)</td>
</tr>
<tr>
<td></td>
<td>(IndexGroup = 0x000201&lt;ii&gt; where &lt;ii&gt; = channel)</td>
</tr>
</tbody>
</table>

### File name

<table>
<thead>
<tr>
<th>Description</th>
<th>Specifies the file which contains the end position specified by file offset. This means that actual processing end at this point.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>STRING</td>
</tr>
<tr>
<td>Value range</td>
<td>[0, 83] – maximum 84 characters</td>
</tr>
<tr>
<td>HMI elements</td>
<td>mc_cmd_bs_pos_end_name_w (write)</td>
</tr>
<tr>
<td></td>
<td>mc_cmd_bs_pos_end_name_r (read)</td>
</tr>
<tr>
<td>IndexOffset</td>
<td>0x53 (write)</td>
</tr>
<tr>
<td></td>
<td>0x6B (read)</td>
</tr>
<tr>
<td></td>
<td>(IndexGroup = 0x000201&lt;ii&gt; where &lt;ii&gt; = channel)</td>
</tr>
</tbody>
</table>

**NOTE**

The specified block search filename must be identical with the file started in automatic mode. It also contains any additional specified path.

If an explicit path is specified at program start, the path must also be included in the block search when the filename is specified.

### Program path ID

<table>
<thead>
<tr>
<th>Description</th>
<th>The identifier indicates whether the NC program defined by the filename is located on the main program path (HP) or the subroutine path (UP).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>UNS16</td>
</tr>
<tr>
<td>Value range</td>
<td>0 - HP (default); 1 - UP</td>
</tr>
<tr>
<td>HMI elements</td>
<td>mc_cmd_bs_pos_end_type_w (write)</td>
</tr>
<tr>
<td></td>
<td>mc_cmd_bs_pos_end_type_r (read)</td>
</tr>
<tr>
<td>IndexOffset</td>
<td>0x52 (write)</td>
</tr>
<tr>
<td></td>
<td>0x6A (read)</td>
</tr>
<tr>
<td></td>
<td>(IndexGroup = 0x000201&lt;ii&gt; where &lt;ii&gt; = channel)</td>
</tr>
</tbody>
</table>

### Pass counter end position
<table>
<thead>
<tr>
<th>Description</th>
<th>The pass counter determines the number of times the end position should be passed up to the end of machining, e.g. in loops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SGN16</td>
</tr>
<tr>
<td>Value range</td>
<td>[1; MAX_SGN16], default 1</td>
</tr>
<tr>
<td>HMI elements</td>
<td>mc_cmd_bs_pos_end_count_w (write) mc_cmd_bs_pos_end_count_r (read)</td>
</tr>
<tr>
<td>IndexOffset</td>
<td>0x55 (write) 0x6D (read) (IndexGroup = 0x000201&lt;ii&gt; where &lt;ii&gt; = channel)</td>
</tr>
</tbody>
</table>
### 5.3 Block search type 3 parameters: Block counter

<table>
<thead>
<tr>
<th>Block counter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td><strong>Value range</strong></td>
</tr>
<tr>
<td><strong>HMI elements</strong></td>
</tr>
<tr>
<td><strong>IndexOffset</strong></td>
</tr>
</tbody>
</table>

### 5.4 Block search type 4 parameters: Block number

<table>
<thead>
<tr>
<th>Block number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td><strong>Value range</strong></td>
</tr>
<tr>
<td><strong>HMI elements</strong></td>
</tr>
<tr>
<td><strong>IndexOffset</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pass counter block number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td><strong>Value range</strong></td>
</tr>
<tr>
<td><strong>HMI elements</strong></td>
</tr>
<tr>
<td><strong>IndexOffset</strong></td>
</tr>
</tbody>
</table>
### 5.5 Status data: Access via HLI

The PLC can access the data listed below via the HLI.

#### Block search is active

<table>
<thead>
<tr>
<th>Description</th>
<th>The interpolator works in block search mode. No axis motion occurs. As long as the block search process in the interpolator remains in the HLI_BS_ACTIVE or HLI_BS_WAIT_FOR_PLC_OFF state, this value indicates TRUE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal flow</td>
<td>CNC → PLC</td>
</tr>
</tbody>
</table>
| ST path     | **CNC Build >= V2.11.2800**  
              gpCh[channel_idx].bahn_state.block_search_active_r  
              where channel_idx = [0, HLI_SYS_CH_MAXIDX]  

              **CNC Build < V2.11.2800**  
              pMC[channel_idx].addr.StateBahn_Data.X_BlockSearchActive  
              where channel_idx = [1, HLI_SYS_CHNMAX] |
| Data type   | BOOL |
| Value range | [TRUE = active - Interpolator works in block search mode., FALSE] |
| Access      | PLC reads |

#### Block search state

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates the current state of the block search mode in the interpolator.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal flow</td>
<td>CNC → PLC</td>
</tr>
</tbody>
</table>
| ST path     | **CNC Build >= V2.11.2800**  
              gpCh[channel_idx].bahn_state.block_search_state_r  
              where channel_idx = [0, HLI_SYS_CH_MAXIDX]  

              **CNC Build >= V2.11.2800**  
              pMC[channel_idx].addr.StateBahn_Data.W_BlockSearchState  
              where channel_idx = [1, HLI_SYS_CHNMAX] |
| Data type   | INT |
| Value range | Identifier | Value |
|             | HLI_BS_INACTIVE | 0 |
|             | HLI_BS_WAIT_FOR_PLC_ON | 1 |
|             | HLI_BS_ACTIVE | 2 |
|             | HLI_BS_WAIT_FOR_PLC_OFF | 3 |
|             | HLI_BS_WAIT_RETURN_TO_CONTOUR | 4 |
|             | HLI_BS_RETURNING_TO_CONTOUR | 5 |
|             | HLI_BS_WAIT_FOR_CONTINUE_CONTOUR | 6 |
| Access      | PLC reads |
Currently covered path in the block - per thousand (PCS)

<table>
<thead>
<tr>
<th>Description</th>
<th>Reads the actually covered path in the current motion block in the PLC (e.g. after feedhold, emergency stop, program abort, etc.).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information flow</td>
<td>CNC → PLC</td>
</tr>
<tr>
<td>Unit</td>
<td>0.1%</td>
</tr>
</tbody>
</table>
| ST path | **CNC Build >= V2.11.2800**  
  \[\text{gpCh[channel_idx]^.bahn_state.covered_distance_r}\]  
  where \text{channel_idx} = [0, \text{HLI_SYS_CH_MAXIDX}]  
  **CNC Build >= V2.11.2800**  
  \[\text{pMC[channel_idx]^.addr^.StateBahn_Data.D_CoveredDistance}\]  
  where \text{channel_idx} = [1, \text{HLI_SYS_CHNMAX}] |
| Data type | DINT |
| Access | PLC reads |
| Characteristics | The read value can be used to command the block search to define the covered path in the NC block from where actual machining should effectively continue. |

Currently covered distance - path increments (PCS)

<table>
<thead>
<tr>
<th>Description</th>
<th>Reads the covered distance in the PLC from program start or from the last NC command #DISTANCE PROG START CLEAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information flow</td>
<td>CNC → PLC</td>
</tr>
<tr>
<td>Unit</td>
<td>0.1 µm</td>
</tr>
</tbody>
</table>
Block search interface and parameters

<table>
<thead>
<tr>
<th>Block search interface and parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Block search interface and parameters</strong></td>
</tr>
<tr>
<td><strong>ST path</strong></td>
</tr>
<tr>
<td>CNC Build &gt;= V2.11.2800</td>
</tr>
<tr>
<td>gpCh[[channel_idx]].bahn_state.dist_prog_start_high</td>
</tr>
<tr>
<td>gpCh[[channel_idx]].bahn_state.dist_prog_start_low</td>
</tr>
<tr>
<td>where [channel_idx] = [0, HLI_SYS_CH_MAXIDX]</td>
</tr>
<tr>
<td>CNC Build &gt;= V2.11.2800</td>
</tr>
<tr>
<td>pMC[[channel_idx]].addr.StateBahn_Data.D_DistProgStartHigh</td>
</tr>
<tr>
<td>pMC[[channel_idx]].addr.StateBahn_Data.D_DistProgStartLow</td>
</tr>
<tr>
<td>where [channel_idx] = [1, HLI_SYS_CHNMAX]</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
</tr>
<tr>
<td><strong>Access</strong></td>
</tr>
<tr>
<td><strong>Characteristics</strong></td>
</tr>
</tbody>
</table>

### Block counter

<table>
<thead>
<tr>
<th>Description</th>
<th>Defines the block counter at which point actual machining is to continue.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information flow</strong></td>
<td>CNC → PLC</td>
</tr>
<tr>
<td><strong>ST path</strong></td>
<td></td>
</tr>
<tr>
<td>CNC Build &gt;= V2.11.2800</td>
<td></td>
</tr>
<tr>
<td>gpCh[[channel_idx]].bahn_state.block_count_r</td>
<td></td>
</tr>
<tr>
<td>where [channel_idx] = [0, HLI_SYS_CH_MAXIDX]</td>
<td></td>
</tr>
<tr>
<td>CNC Build &gt;= V2.11.2800</td>
<td></td>
</tr>
<tr>
<td>pMC[[channel_idx]].addr.StateBahn_Data.D_BlockCount</td>
<td></td>
</tr>
<tr>
<td>where [channel_idx] = [1, HLI_SYS_CHNMAX]</td>
<td></td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>UDINT</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>PLC reads</td>
</tr>
</tbody>
</table>

### Distance from continuation position

<table>
<thead>
<tr>
<th>Description</th>
<th>If an NC program is started in block search, the NC program is processed in simulative mode (without axis motion) until the continuation position is reached. Block search is then in the HLI_BS_WAIT_FOR_PLC_OFF state and calculates the distance between the actual positions of the axis and the continuation position. If block search is in the HLI_BS_RETURNING_TO_CONTOUR state, this value is refreshed cyclically.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal flow</strong></td>
<td>CNC → PLC</td>
</tr>
<tr>
<td><strong>ST path</strong></td>
<td></td>
</tr>
<tr>
<td>CNC Build &gt;= V2.11.2800</td>
<td></td>
</tr>
<tr>
<td>gpCh[[channel_idx]].bahn_state.block_search_path_deviation_r</td>
<td></td>
</tr>
<tr>
<td>where [channel_idx] = [0, HLI_SYS_CH_MAXIDX]</td>
<td></td>
</tr>
<tr>
<td>CNC Build &gt;= V2.11.2800</td>
<td></td>
</tr>
<tr>
<td>pMC[[channel_idx]].addr.StateBahn_Data.D_BlockSearchPathDeviation</td>
<td></td>
</tr>
<tr>
<td>where [channel_idx] = [1, HLI_SYS_CHNMAX]</td>
<td></td>
</tr>
<tr>
<td><strong>Unit</strong></td>
<td>0.1 µm</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>UDINT</td>
</tr>
<tr>
<td><strong>Value range</strong></td>
<td>0..MAX_SGN32</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>PLC reads</td>
</tr>
</tbody>
</table>

### Stop condition

<table>
<thead>
<tr>
<th>Description</th>
<th>Displays the condition why the current motion was stopped.</th>
</tr>
</thead>
</table>
### Signal flow

<table>
<thead>
<tr>
<th>ST path</th>
<th>Description</th>
</tr>
</thead>
</table>
| **CNC Build >= V2.11.2800** | gpCh[channel_idx].bahn_state.stop_conditions_r  
where channel_idx = [0, HLI_SYS_CH_MAXIDX] |
| **CNC Build < V2.11.2800** | pMC[channel_idx].addr.StateBahn_Data.D_StopConditions  
where channel_idx = [1, HLI_SYS_CHNMAX] |

### Data type

DINT

### Value range

See table below with explanations.

### Access

PLC reads

### Constant in PLC | Value | Explanation
---|---|---
HLI_SC_FEEDHOLD | 0x0001 | Path feed stop
HLI_SC_VFG | 0x0002 | No axis-specific feed enable.
HLI_SC_SINGLE_BLOCK | 0x0004 | Single step mode active.
HLI_SC_M00_OR_M01 | 0x0010 | M00 (programmed stop), M01 (optional stop) is active.
HLI_SC_PLC_ACKNOWLEDGE | 0x0020 | Stop occurs due to waiting for an acknowledgement from the PLC. This may occur as a result of the output of M or H technology functions but is not restricted to this alone.
HLI_SC_OVERRIDE_ZERO | 0x0040 | Override = 0.
HLI_SC_DELAY_TIME | 0x0200 | Dwell time.
HLI_SC_CHANNEL_SYNC | 0x0800 | Channel synchronisation is active.
HLI_SC_IPO_INPUT_EMPTY | 0x1000 | Input FIFO of the interpolation is empty.
HLI_SC_IPO_INPUT_DISABLED | 0x2000 | Read function blocks (e.g. motion blocks, etc.) disabled.
HLI_SC_WAIT_FOR_AXES | 0x8000 | Stop occurs due to waiting until a commanded axis swap is completed.
HLI_SC_CHANNEL_ERROR | 0x00010000 | An error occurred in the channel.
HLI_SC_WAIT_TECHNO_ACK | 0x00020000 | Waiting for acknowledgement of M/H/ST technology functions.
HLI_SC_W_C_AFTER_COLLISION | 0x00040000 | After a detected collision, waiting for motion resumption.
HLI_SC_SLOPE_SUPPLY_PROBLEM | 0x00080000 | Block supply problem (only occurs in conjunction with HSC slope).
HLI_SC_BACK_INTERPOLATION | 0x00100000 | Back interpolation after tracking mode is active.
HLI_SC_STOP_REVERSIBLE | 0x00200000 | Stop since M00 (programmed stop) is active. However, the NC program can be processed backwards despite M00 (available as of V3.1.3039.01).
HLI_SC_BREAKPOINT_STOP | 0x00400000 | Stop because a breakpoint (stop point) is reached; available as of V2.11.2024.03, V2.11.2807.01, V3.1.3039.01.

Value range of stop conditions
### 5.6 Status data: Access via CNC objects:

<table>
<thead>
<tr>
<th>Covered distance - path increments</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Distance from program start or from the last NC command #DISTANCE PROG START CLEAR.</td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>REAL64</td>
<td></td>
</tr>
<tr>
<td><strong>Value range</strong></td>
<td>[ MIN_REAL64; MAX_REAL64 ]</td>
<td></td>
</tr>
<tr>
<td><strong>HMI elements</strong></td>
<td>distance_from_program_start (read)</td>
<td></td>
</tr>
<tr>
<td><strong>IndexOffset</strong></td>
<td>0x52 (read)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(IndexGroup = 0x000213&lt;ii&gt; where &lt;ii&gt; = channel)</td>
<td></td>
</tr>
</tbody>
</table>
6 Locking program areas for block search

In the NC program, any program areas can be locked for block search by the command

`#BLOCKSEARCH LOCKED/RELEASED`

any program areas can be locked for the block search. If the continuation position of the block search then lies in one of these locked areas, the error message P-ERR-21399 is output.

The block search lock also includes all local and global subroutine called in the corresponding area.

If locked areas are nested, the block search lock includes the area starting at the first activation up to the first deactivation (see example 2).

| #BLOCKSEARCH LOCKED | RELEASED | (modal) |

Lock program areas for block search

Example 1:

No start position may be selected for block search in the range of NC blocks N40–N100 including the subroutines called in it.

```plaintext
%BLOCKSEARCH
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #BLOCKSEARCH LOCKED
N50 X20
N60 Y20
N65 L GSP.nc
N70 Z20
N80 X30
N90 Z30
N100 #BLOCKSEARCH RELEASED
N110 Y30
N120 X40
N130 Z40
N999 M30
```
Example 2:

The area of the block search lock when nested covers N40-N75.

```
%BLOCKSEARCH
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #BLOCKSEARCH LOCKED
N50 X20
N55 #BLOCKSEARCH LOCKED
N60 Y20
N65 L GSP.nc
N70 Z20
N75 #BLOCKSEARCH RELEASED
N80 X30
N90 Z30
N100 #BLOCKSEARCH RELEASED
N110 Y30
N120 X40
N130 Z40
N999 M30
```
## 7 On/off handshake with PLC

While block search is active, the PLC is notified of every technology M function (e.g. torch on/off). As opposed to normal mode, however, this is not executed. Instead this is handled in the PLC by a special module. The PLC is then notified of every change in status of block search mode (on/off). The PLC then acknowledges the change and sends it to the NC: The NC waits for the acknowledgement from the PLC in the same way as for synchronised M functions.

If block search is active during an NC reset, block search without handshake is deselected (the PLC should then be notified of the reset by the reset specific control unit).

### Block search on/off to PLC

<table>
<thead>
<tr>
<th>Description</th>
<th>At every block search on/off request, the CNC initiates a handshake with the PLC; As long as the request element [51] has the value TRUE, the PLC is notified of every change in block search mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>If an NC program starts with block search mode activated, the CNC sets the CNC signal [51] to TRUE and waits for the PLC signal [51], indicating that the PLC is ready for block search. When the PLC has executed the necessary actions to prepare for block search, it notifies this to the CNC by setting the PLC signal [51] to TRUE. After this signal, the NC program can be processed in block search mode. This takes place either by the &quot;Continue motion&quot; or &quot;Program start&quot; commands. If the restart position is reached while the NC program is processed, the CNC signals this by setting the CNC signal [51] to FALSE. The PLC detects this, completes its preparations for operation with real axis motions and then sets the PLC signal [51] to FALSE.</td>
</tr>
</tbody>
</table>

### ST path, ST element for CNC Build >= 2800

<table>
<thead>
<tr>
<th>ST path</th>
<th>gpCh[channel_idx]^^.bahn lc_control.block_search where channel_idx = [0, HLI_SYS_CH_MAXIDX]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>LC_CONTROL_BOOL_UNIT</td>
</tr>
<tr>
<td>CNC signal</td>
<td>.command_r</td>
</tr>
<tr>
<td>Data type</td>
<td>BOOL</td>
</tr>
<tr>
<td>Value range</td>
<td>[TRUE = NC program was started in block search mode, FALSE = Block search is disabled]</td>
</tr>
<tr>
<td>PLC signal</td>
<td>.state_r</td>
</tr>
<tr>
<td>Data type</td>
<td>BOOL</td>
</tr>
<tr>
<td>Value range</td>
<td>[TRUE = PLC acknowledged the notification about block search activation, FALSE = PLC acknowledged the notification of block search deactivation]</td>
</tr>
<tr>
<td>Request</td>
<td>.enable_w</td>
</tr>
<tr>
<td>Data type</td>
<td>BOOL</td>
</tr>
<tr>
<td>Value range</td>
<td>[TRUE = PLC wants to be notified about activation of block search, FALSE]</td>
</tr>
</tbody>
</table>

### ST path, ST element for CNC Build > 2800

<table>
<thead>
<tr>
<th>ST path</th>
<th>pMC[channel_idx]^^.addr.LCControlBahn_Data.LCControlBoolUnit_BlockSearch where channel_idx = [1, HLI_SYS_CHNMAX]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>LCControlBoolUnit</td>
</tr>
<tr>
<td>CNC signal</td>
<td>.X_Command</td>
</tr>
</tbody>
</table>

If block search is active during an NC reset, block search without handshake is deselected (the PLC should then be notified of the reset by the reset specific control unit).
On/off handshake with PLC

<table>
<thead>
<tr>
<th>PLC signal</th>
<th>ST element</th>
<th>Request</th>
<th>ST element</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.X_State</td>
<td></td>
<td>.X_Enable</td>
</tr>
</tbody>
</table>

**wait for block search on / off**

![Diagram](image)

Figure 27: Interaction between BOOLEAN-LC control unit and PLC

When the CNC is reset, the CNC resets the CNC signal and the PLC signal.
8 Known restrictions

Offset
If the program continues with an offset after block search since it was not fully restarted on the contour, this offset remains valid up to program end and reset. When the program is restarted, the motion continues without offset.

Motion and technology in starting block
If an NC block intended for restart receives motion and technology commands (e.g. N100), only the remaining motion commands are actually executed. All technology commands are only simulated in block N100 (block search mode).

Motion and technology in starting block

<table>
<thead>
<tr>
<th>N90</th>
<th>X90</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100</td>
<td>X100 S1000 M3 M111</td>
</tr>
<tr>
<td>N110</td>
<td></td>
</tr>
</tbody>
</table>

If technology commands also require to be executed in real motion, the operator may only continue to move in block search up to the end of block N90 with a covered distance of 100%. This refers to the above example.

Automatic restart
Please note the points below for automatic restart to the contour:

1. Axes are moved backward along a straight line. The starting movement is executed with rapid traverse values (G00). If obstacles occur on the straight line, the operator must carry out a manual repositioning before automatic restart to the contour.
2. If soft gantry is active, the starting movement is executed decoupled for slave axes, i.e. the axes only move in coupled mode after the starting block.
3. An axis swap is then permitted in block search and continues to permit return to the contour if the axes are requested by a request for positions by the interpolator (default setting, not with #CALL AX FAST).
4. Axes which are moved during block search and which are released before switch-over to read mode may not start automatically at the last known position.

In the example below, the B axis is not moved back to position 45 if automatic restart to the contour is selected. The Z axis is not moved either. The C axis is moved correctly to the contour.

Automatic restart

<table>
<thead>
<tr>
<th>N00</th>
<th>X10 Y20</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>#CALL AX [B, 4, 3]</td>
</tr>
<tr>
<td>N20</td>
<td>B45     (B axis is not considered)</td>
</tr>
<tr>
<td>N30</td>
<td>#PUT AX [B]</td>
</tr>
<tr>
<td>N40</td>
<td>Z100</td>
</tr>
<tr>
<td>N50</td>
<td>#PUT AX [Z] (Z axis released, cannot be)</td>
</tr>
<tr>
<td></td>
<td>(restored)</td>
</tr>
<tr>
<td>N60</td>
<td>#PUT AX [X,Y]</td>
</tr>
<tr>
<td>N70</td>
<td>#CALL AX [X, 1, 1] [Y, 2, 0] (exchange X/Y) (considered)</td>
</tr>
<tr>
<td></td>
<td>(-Continuation position-)</td>
</tr>
<tr>
<td>N100</td>
<td>X100 Y200 (Continue normal processing) (after block search)</td>
</tr>
<tr>
<td>N30</td>
<td></td>
</tr>
</tbody>
</table>
Known restrictions

If program processing is to continue to the continuation position after an interruption using block search, the parameters can be obtained directly from the status data in the PLC (see section Status data: Access via the HU) [44]. If the block search parameters are specified directly based on the NC program, it must be taken into consideration that contour influencing functions, e.g. tool radius compensation and polynomial smoothing, affect the original target positions and block limits.

**NOTE**

If a program only contains relative G91 positions in the block search section, different positions may be approached in the block search despite a restart of the same program.

It is recommended that at least one absolute position occurs in the block search section of the NC program.

**NOTE**

A homing run G74 is omitted during block search. This may result in different follow-on positions or axis offsets compared to real processing.

**NOTE**

A measurement block G100 {<axis><destination>} is replaced by a linear motion to the target position during block search.

G01 {<axis><destination>}

Therefore, the specified target position is approached as if the probe signal had not occurred during the measurement.

This may result in different follow-on positions of the axes compared to real processing.
9 Examples

9.1 Block search type 4

9.1.1 Specify block number and pass counter

Type 4: Block number and number of passes

Block search type = 4, block number = 100

Set number of passes to 1 and start program

Program stops at continuation position X = 10 Y = 12. After continuation, the residual part of the square is traversed.

Block search type 4

&t_sv_number.nc
N00 G00 G90 X2 Y2
P1 = 0
N80 G01 G91 X10 F500
N95 Y10
$FOR P1 = 1, 5, 1
N100 X-2
$ENDFOR
N120 Y-10
N130 M30

<table>
<thead>
<tr>
<th>Number of passes = 1</th>
<th>Number of passes = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search active</td>
<td>Block search active</td>
</tr>
<tr>
<td>Real axis motion</td>
<td>Real axis motion</td>
</tr>
<tr>
<td>Axis motion at program start position and continuation position is not marked</td>
<td>Axis motion at program start position and continuation position is not marked</td>
</tr>
</tbody>
</table>

```
N00 G00 G90 X2 Y2
P1 = 0
N80 G01 G91 X10 F500
N95 Y10
$FOR P1 = 1, 5, 1
N100 X-2
$ENDFOR
N120 Y-10
N130 M30
```
Specify the block number and covered distance in the block

**Distance covered in the block**

Block search type = 4, block number = 100

Set covered distance and start program.

After continuation, the residual part of the square is traversed.

**Block search type 4**

<table>
<thead>
<tr>
<th>Covered distance = 0 %</th>
<th>Covered distance = 300 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search active</td>
<td></td>
</tr>
<tr>
<td>Real axis motion</td>
<td></td>
</tr>
<tr>
<td>Axis motion at program start position and continuation position is not marked</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Covered distance = 800 %</th>
<th>Path covered = 1000 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search active</td>
<td></td>
</tr>
<tr>
<td>Real axis motion</td>
<td></td>
</tr>
<tr>
<td>Axis motion at program start position and continuation position is not marked</td>
<td></td>
</tr>
</tbody>
</table>
9.2 Block search type 3

9.2.1 Specify block counter

Block counter

Block search type = 3, block counter = 100

Start program

Program is executed up to continuation position. NBC waits for continue. After continuation, the residual part of the square is traversed.

```
%t_sv_count.nc
N00 G00 G90 X2 Y2
P1 = 0
N80 G01 G91 X10 F500
N095 Y10
$FOR P1 = 1, 5, 1
N100 X-2
$ENDFOR
N120 Y-10
N130 M30
```

<table>
<thead>
<tr>
<th>Block counter (BC) = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search active</td>
</tr>
<tr>
<td>Real axis motion</td>
</tr>
<tr>
<td>Axis motion at program start position and continuation position is not marked</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block counter (BC) = 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search active</td>
</tr>
<tr>
<td>Real axis motion</td>
</tr>
<tr>
<td>Axis motion at program start position and continuation position is not marked</td>
</tr>
</tbody>
</table>

```
Block counter = BC
%t_sv_count.nc
N00 G00 G90 X2 Y2
P1 = 0
N80 G01 G91 X10 F500
N095 Y10
$FOR P1 = 1, 5, 1
N100 X-2
$ENDFOR
N120 Y-10
N130 M30
```
10 Exceptions and errors

Block search to a block without motion

Start program and continue in N100 at 10% of covered distance.
Since N100 contains no motion, the block is not split at 10%.

Block search to a block without motion

%t_sv_p.nc
N907090 G01 X0 Y0 Z0 F1000
N070 Y10
N095 X2.1 Y2.2 Z2.3
N100 P100 = 1
N110 G00 G91 X10
N120 G90 X3.1 Y3.2 Z3.3
N907091 M30

Continuation position not found

Block search type = 4, start program

Set number of passes too high/too low. Set unknown block number.

If the number of passes is smaller than 2, the first occurrence of the block number is taken as the continuation position.

If the number of passes is greater than 5, the continuation position is not found and the message P-ERR-20704 is output.

If the block number is not found for continuation, a message (warning) is also output and the entire NC program is terminated in block search mode.

Block search type 4, start program

%t_sv_number.nc
N00 G00 G90 X0 Y0
P1 = 0
N08 G01 G91 X10 F500
N095 Y10
$FOR P1 = 1, 5, 1
N100 X-2
$ENDFOR
$ENDFOR
N120 Y-10
N130 M30
11 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff’s branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff’s branch offices and representatives round the world can be found on her internet pages: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963 157
Fax: +49 5246 963 9157
e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963 460
Fax: +49 5246 963 479
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963 0
Fax: +49 5246 963 198
e-mail: info@beckhoff.com
web: https://www.beckhoff.com
# Index

## B

<table>
<thead>
<tr>
<th>Bahn</th>
<th>Stopp:Grund</th>
<th>46</th>
</tr>
</thead>
</table>

## H

<table>
<thead>
<tr>
<th>Halt</th>
<th>Bahn:Bedingung</th>
<th>46</th>
</tr>
</thead>
</table>

## S

<table>
<thead>
<tr>
<th>Satzvorlauf</th>
<th>Interpolator:aktiv</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satzvorlauf an/aus</td>
<td>an PLC</td>
<td>51</td>
</tr>
<tr>
<td>Stopp</td>
<td>Bahn:Grund</td>
<td>46</td>
</tr>
</tbody>
</table>
More information:
www.beckhoff.com/TF5200