CNC Motion Mode

About this document

This document is mostly relevant for formal FW version number 1.3.0.1-165 (to become formal 1.4.0) and later.

About Motion Modes in general

The various motion modes determine what type of position reference will be generated for the relevant axis.

The position reference can be a single axis motion, independent of the other axes (such as: Point-to-Point motion) or a synchronized multi-axes motion (such as: Master/Slave gearing and CNC).

The reference can be internally calculated, as a function of parameters provided by the user (such as Point-to-Point motion using parameters as Acceleration, Speed and more) or can be a function of external signals, such as Pulse/Direction or Joystick mode (based on analog input).

The motion mode is determined using the MotionMode keyword and can't be modified while in motion. The motion itself starts using the Begin keyword. The motion ends when the motor arrives to its target (such as in Point-to-Point motion) or by the user (using for example the Stop keyword) or due to reaching motion limits, such as the Reverse Limit Switch.

Some of the motion parameters can be changed during motion (on-the-fly).

The actual position and velocity, as well as any other status keyword (such as Motion Status: MotionStat), can be inquired at any time.

Data Recording can be used to record the motion behavior and display it using the PC Suite software.

Most of the supported motion modes can be easily initiated and controlled using the relevant tool window at the PC Suite software.

Note:

The descriptions of controller keywords within this document are only to provide satisfactory understanding of their functionality. For full descriptions of each keyword please refer to the communication protocols and the keywords reference documents.

About Begin-On-Input

Typically, a motion starts when the controller receives the Begin keyword. However, the controller supports a delayed start of motion, where the motion itself will start only upon a rising edge of a user defined discrete input.

When the BeginDInOn parameter is set to 0, the motion will start immediately upon receiving the Begin message. When the BeginDInOn is set to 1, and a Begin message is received, the motion start will wait for the relevant discrete input rising edge.

Using this feature, a motion can be synchronized with an external event. By using the same signal as an input to multiple number of axes and/or controllers, multiple axes can be synchronized to each other.

About CNC motion

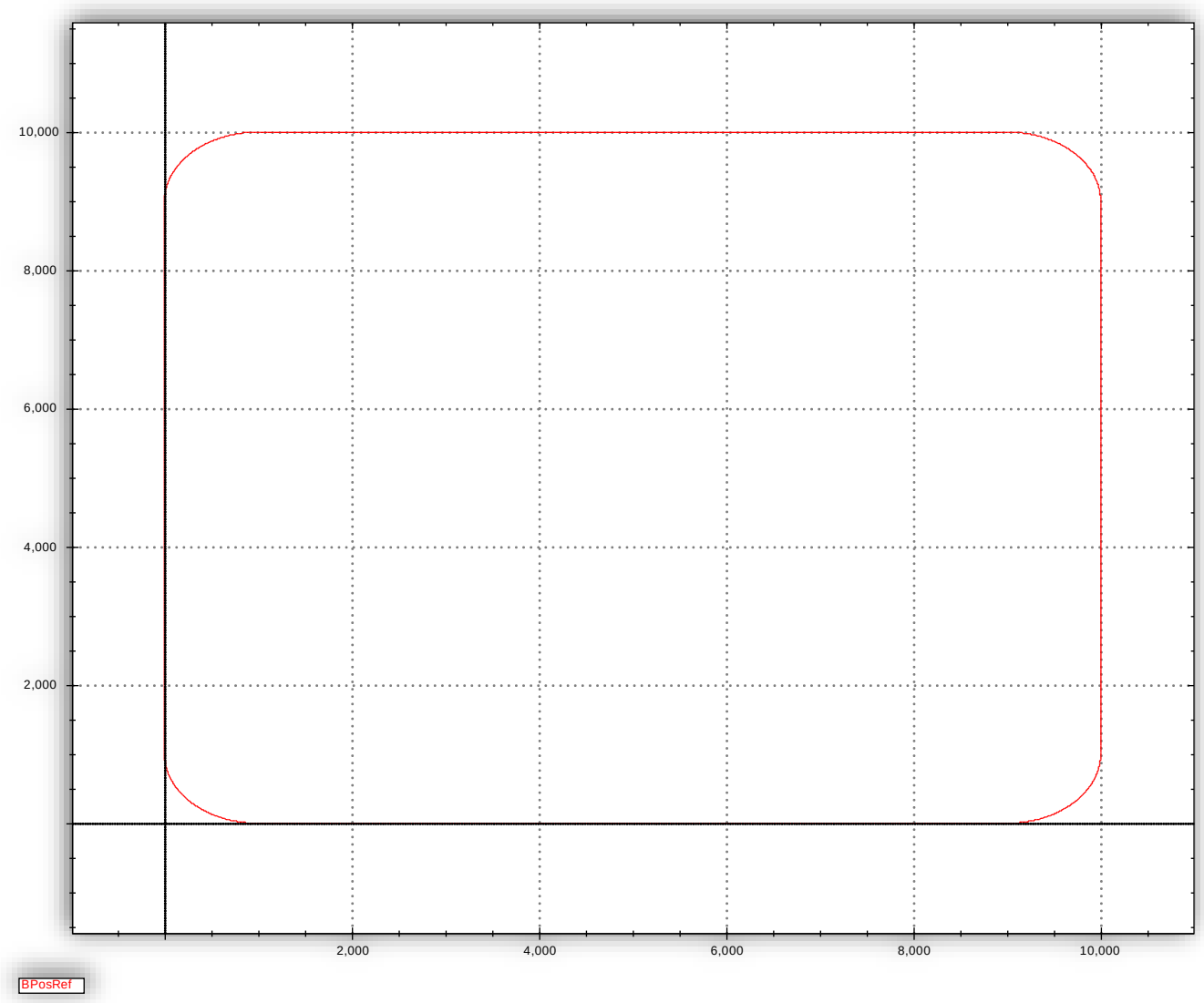
CNC FIFO is a special multi-axes motion mode in which the controller performs a sequence of motion segments, where each motion segment is a multi-axes motion segment, as defined by the user before the motion and optionally also during the motion.

The motion is created according to the motion segments which are stored in a CNC FIFO memory.

The motion segments definitions can be pushed into the CNC FIFO at any time (before or during the motion), providing that the CNC FIFO is not full. If the CNC FIFO is full, the push operation is rejected with a suitable error.

If, during a motion in this mode, the controller reaches the last segment in the CNC FIFO, and completing this motion segment, and yet no new segment was pushed (the CNC FIFO is empty), the motion is automatically ended.

The following figure shows a typical CNC motion, of two axes, consisting of linear segments and arc segments:



The CNC motion consists of a sequence of motion segments. The CNC profiler in the controller calculates the CNC motion along the CNC path, using parameters provided by the user for the vector acceleration, deceleration, speed and end speed. Once the desired motion along the CNC path is calculated, the relevant motion per each involved axis is derived from this desired vector motion.

Unlike Point-to-Point motion, where the end speed of the motion is always zero, here, per each segment, the end speed of each segment is provided as part of the segment definitions, to allow continuous motion between the segments. The end speed of a given segment is used as the start speed of the next segment.

Currently, the CNC motion assumes that all member axes has the same physical resolution. This means that the unit of the vector acceleration, deceleration, speed and end speed are identical to the units of each member axis. The CNC can be used also with non-identical axes' resolutions but in such case:

- The units of the vector motion's parameters will not be intuitive.
- Arc motions will not draw a circle.

Future firmware versions will support scaling between the member axes to compensate for non-identical resolution.

A CNC motion definition consists of a user defined sequence of segments. Segments are mainly Linear Motion or ARC motion. However, additional segment types are defined so that the user can control the vector motion parameters, as well as to synchronize controller events with the CNC motion.

While CNC motion is in process, the controller uses the segments definitions to calculate the motion. However, the user can perform the following on-the-fly changes of the motion:

- Change the vector speed: using the CNCAPercents keyword, the user can increase/decrease the nominal vector speed as defined within the CNC FIFO. For example, if a given segment is defined to create a motion with vector speed of 100000 (counts/sec), and CNCAPercents is set by the user to 50 (%), that the actual vector speed of this motion will be 50000 (counts/sec).

- Pause the CNC motion: using the CNCAPause keyword, the user can pause the motion (the CNC motion will decelerate to zero vector speed along the CNC motion path) and continue the motion (the CNC motion will accelerate to the desired vector speed of the current segment).

- Execute the segments one-by-one (step mode). The CNC motion will pause at the end of each segment and will wait for a user command to perform the next segment.

Note:

The CNC motion mode is a relatively complicated motion mode. Creating a motion requires to create and push the motion segments into the CNC FIFO. There are some restrictions regarding the initial sequence of the segments.

Therefore, we recommend to first use the PC Suite Motion/CNC tool window (see last chapter of this document) to practice the process of creating a CNC motion and only later to move into creating a user application for CNC motions.

CNC FIFO memory management and related keywords

- There is a single CNC FIFO buffer that is used globally to create multi axis motions to up to 6 axes per each segment (meaning, the protocol and FIFO structure is ready for up to 6 axes motion, although, currently, the controllers support less than 6 axes).

- This means that the keywords to manage the CNC FIFO are not axis related.

- Future multi axes controller may support few CNC FIFOs, to enable few independent spatial motions at the same time (using different physical axes). In order to support this future feature, the implementation of the single CNC FIFO has a name of CNC FIFO-A (future: CNC FIFO-B, CNC FIFO-C ... as many as will be supported by each product).

- CNC FIFO-X denotes one of these CNC-FIFO instances. Currently, our controllers support a single instance, named CNC FIFO-A (meaning, only a single CNC motion can be executed at any given time).

- The CNC FIFO-X is based on a large (size is product dependent) memory buffer that holds the type and parameters of the CNC motion segments. The name of this memory buffer is CNCXFIFO (for example: CNCAFIFO).

- The CNCXFIFO is a buffer of 32-bit entries (type: long) and it can be accessed just as any other array parameter of the controller (i.e.: CNCAFIFO[123]). The CNCXFIFO is a read only parameter and writing (pushing) data into the FIFO is done indirectly using a set of other keywords, as defined below.

- The CNCXFIFO is managed as a cyclic buffer with pointer to new data to add and pointer to oldest data in the FIFO. However, this is a special cyclic buffer in a way that the first entry of a new segment is not pushed at the last N items in the CNC FIFO (N is typically 20). Due to this implementation, pushing and popping items to/from the FIFO does not need to be performed with checking for the cyclic buffer. It is ensured that once a start point of a segment is known, the whole segment will fit into the buffer without crossing the end of the buffer.

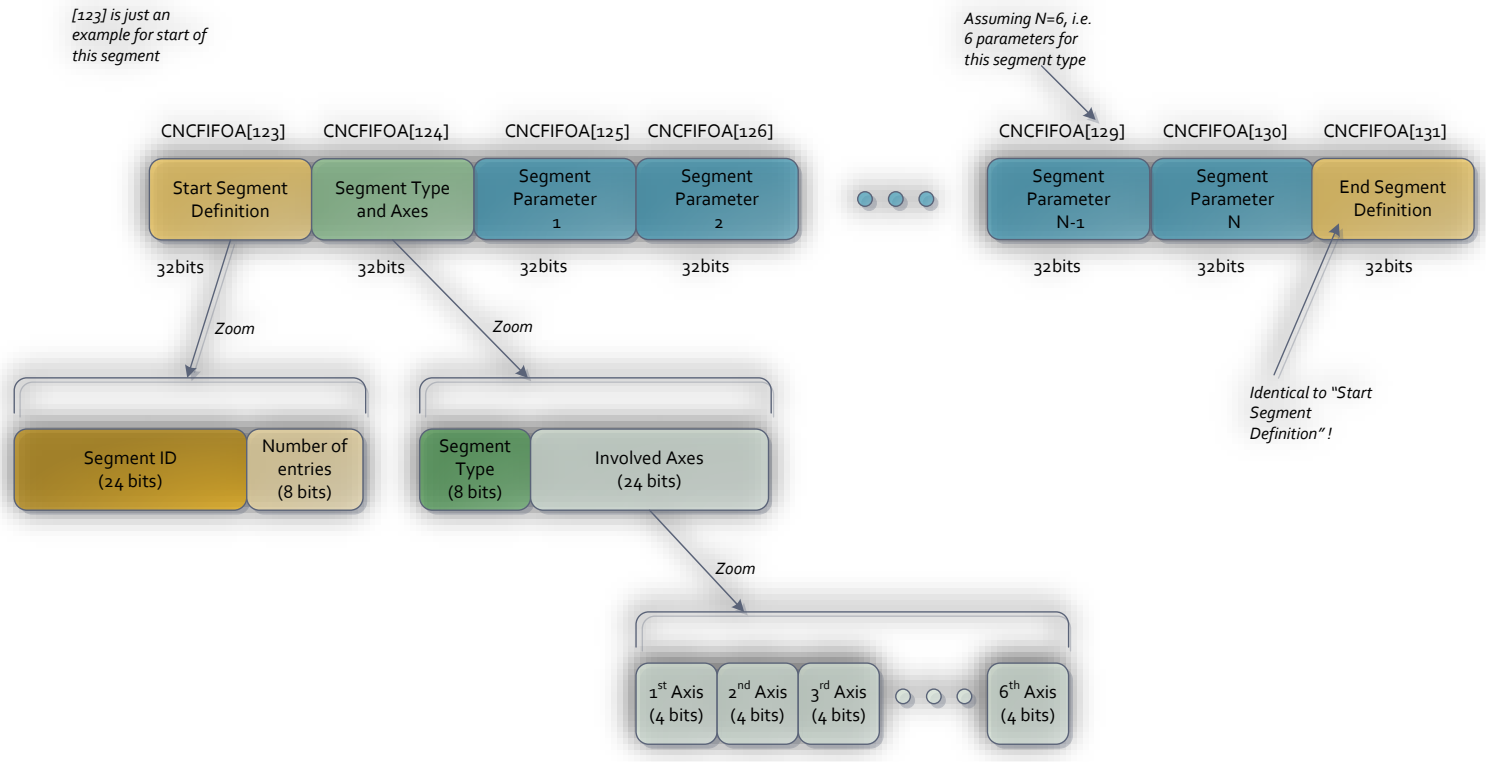
- Each spatial motion segment that is pushed into the CNC FIFO includes:

  - Segment ID (for tracking of motion location during motion).
  - Segment type (ARC, Linear, Helical, etc.)
  - Involved axes (A, B, etc.).
  - Segment parameters.

- Each segment type (ARC, Linear, Helical, …) has different number of parameters (some few, some many).

- The following figure presents the structure (variable length!) of a motion segment within the CNC FIFO:

CNC FIFO – Structure of a Single Segment



Note:

In the drawing we use the notation CNCFIFOA[]. The final name of this parameter is CNCAFIFO[]. This is to keep compatibility with the names of all other CNC related keywords as listed below.

- The first entry (32 bits) of any segment is the "Start Segment Definition". It has a unique ID (unsigned 24 bits) and includes the number of entries (in the CNC FIFO) that are used for this segment (in this example, N=6 (number of parameters), so number of entries is N+3=9.

- The first entry is repeated identically also as the last entry of the segment. The purpose of this "End Segment Definition" (holding data that is identical to the "Start Segment Definition") is to enable safe removal of a segment. When removing a segment, the pointer into the CNC FIFO needs to move one segment backward, and we need to know what the length of this segment is, as otherwise we can't locate the "Start Segment Definition".

- The "Segment Type" field holds the type of the segment ("Linear", ARC", etc.) as defined in more details within the next chapters of this document.

- The "Involved Axes" field consists of flexible definition of axes. The meaning of 1st axis, 2nd axis and so on depends on the segment type. For example: Helical motion segment can define 1st axis and 2nd axis for the cyclic motion and 3rd axis for the linear motion.

- Each 1st/2nd/3rd... axis can be defined to be one of up to 8 physical axes (preparation for future multi-axes controllers) or even none (not participating). Therefore 4 bits are given per each Involved Axis definition.

- Each "Included Axis" field (4 bits) can get the following values:

  - 0: Axis A
  - 1: Axis B
  - ... up to the number of axes supported by this controller.
  - 15: indicates that this axis is not involved.

- A total of up to 6 axes can participate in any segment.

- The following keywords are defined to manage the CNC FIFO:

Notes:

- *The explanations and examples refer to the CNC FIFO A, which is the only CNC FIFO that is supported currently. Nevertheless, just replace CNCA with CNCB to refer to the CNC FIFO B, for example.*
- *The keywords name and format description are provided without the axis letter. Yet, when sending a message to the controller, the message must start with the relevant axis letter (or any axis letter if the keyword is non-axis related, as most of the CNC keywords).*

- CNCAPushType:

  Used to initiate a new segment in the FIFO.

  Format:

  CNCAPushType, <Value>

  Value consists of a 32 bits value, holding the segment type (upper 8 bits) and the involved axes (lower 24 bits).

  Based on the segment type, the controller knows the relevant number of entries for parameters.

ID is given automatically by the controller, as the ID of the previous segment plus 1. When reaching the maximal possible value of unsigned 24 bits (0x7FFFFF = 8,388,607), the ID is rolled to 0 and starts over. This has no effect on the operation of the mode. The user just needs to consider it when using the ID status report (at CNCAStatus[9], see below) to conclude what is the location of the motion.

The first ID when pushing a segment to an empty CNC FIFO is 1. Value of 0 is reached only after rolling from the maximal value, as explained above.

The controller check if all is OK (segment type is supported, CNC FIFO has space for the needed number of entries, number of involved axes is suitable to the segment type and involved axes are properly defined, etc.) and if all is OK, the "Start Segment Definition" and "Segment Type and Axes" entries are created and pushed to the CNC FIFO.

The controller replies with OK> or with the error code if error was detected.

In the case of error, nothing is pushed into the CNC FIFO and it remains as it was before this message.

CNCAPushType can be executed only if the previous segment was fully pushed and closed or if the CNC FIFO is empty. Otherwise it is an error.

- CNCAPushParam:

  Used to push values of parameters into the CNC FIFO.

  Format:

  CNCAPushParam, <Value>

  Value consists of a 32 bits value, holding the value of the currently pushed parameter.

  It is the Host responsibility to push the parameters in the correct order (and correct number of parameters) as suitable for this segment type.

  Once the last parameter is pushed (the controller is aware to the number of parameters needed for this segment type), the controller closes the segment by adding the "End Segment Definition" (identical to "Start Segment Definition") to the CNC FIFO.

  At this time, further CNCAPushParam messages will be rejected with an error, as the segment is closed. Only CNCAPushType is valid now.

  The controller replies with OK> or with the error code if error was detected.

  In the case of error, nothing is pushed into the CNC FIFO and it remains as it was before this message.

  Upon pushing in the last parameter, the controller also verifies that all input data for this segment is valid for the giver segment type (parameters in range, involved axes definition is reasonable, etc.). If not valid, this push value is rejected with a suitable error message.

- Additional keywords related to the CNC FIFO management and status:

  - CNCAStatus[]:

    A read only array parameter holding the following described status data.

    Some of the statuses are not valid after reset or power on, some others are not valid when the CNC FIFO is empty and some other when there is no CNC FIFO motion and so on. These

statuses get the value -1 when they are not valid (this value is not a valid value for any of these statuses when they are valid).

i. CNCAStatus[1] → OLDEST_ENTRY_INDEX
Index to CNCAFIFO[] (CNC FIFO memory buffer) to the entry of the oldest element in the CNC FIFO A. Note: it is also the location of the start of the oldest segment in the CNC FIFO A.

ii. CNCAStatus[2] → LAST_PUSHED_ENTRY_INDEX
Index to CNCAFIFO[] (CNC FIFO memory buffer) to the entry of the last pushed item.

iii. CNCAStatus[3] → LAST_SEGMENT_ENTRY_INDEX
Index to CNCAFIFO[] (CNC FIFO memory buffer) to the entry of the start of the last pushed (or being pushed) segment.

iv. CNCAStatus[4] → LAST_SEGMENT_TOTAL_NUM_PARAMETERS_INDEX
Total number of parameters that need to be pushed for the currently being pushed segment. Function of the segment type. 0 if there is no open segment that is being pushed now.

v. CNCAStatus[5] → LAST_SEGMENT_NEED_TO_PUSH_NUM_PARAMETERS_INDEX

Number of parameters that still need to be pushed for the currently being pushed segment. 0 if there is no open segment that is being pushed now.

vi. CNCAStatus[6] → IN_MOTION_SEGMENT_ENTRY_INDEX
Index to CNCAFIFO[] (CNC FIFO memory buffer) to the entry of the start of the currently executed segment (valid during CNC FIFO motion).

vii. CNCAStatus[7] → FREE_SPACE_INDEX
Free space within the CNC FIFO A (number of unused 32 bits entries).
Note that the controller avoids placing new segments at the last 20 entries of the CNC FIFO (helps for more efficient handling of the cyclic CNC FIFO handling). As a result, the user should not try to push segments if the free space is reaching a value close to 20.

viii. CNCAStatus[8] → LAST_USED_ID_INDEX
The last ID that was used (or is currently used if a segment was pushed and not yet closed).

ix. CNCAStatus[9] → IN_MOTION_ID_INDEX
During motion, the ID of the currently executed segment.

x. CNCAStatus[10] → CNC_FIFO_MOTION_STATUS_INDEX
See details below.

xi. CNCAStatus[11] → CNC_STATUS_CNC_FIFO_MEMBERS_AXES_INDEX
Holds (bit per axis) the axes that are participating in the active CNC motion (member axes), as defined when the motion was started (not specifically for the current motion segment which can involve a subgroup of the participating axes).

xii. CNCAStatus[12] → CNC_FIFO_INVOLVED_AXES_INDEX
Holds (bit per axis) the axes that are participating in the currently active motion segment (always a subgroup of the CNC member axes).

xiii. CNCAStatus[13] → CNC_STATUS_MAX_ACCELERATION_COUNTER_INDEX

Holds a counter of the number of pushed Automatic Corner segments at which the corner speed has to be reduced due to axis acceleration limitation. It is cleared to zero upon power on, reset and CNCAClear message.

xiv. CNCAStatus[14] → CNC_STATUS_MAX_VELOCITY_JUMP_COUNTER_INDEX

Holds a counter of the number of junctions between linear segments at which the end speed of the first segment was reduced due to violation of the maximal velocity jump for one of the involved axes. It is cleared to zero upon power on, reset and CNCAClear message.

Detailed description of CNCAStatus[10] (CNC motion status):

Note:

Bit 0 is the right-most bit of the value.

i. Bit 0: "1" if CNCA is in-motion. "0" otherwise.

ii. Bit 3: "1" if the CNCA performs a stop request (Stop keyword).
"0" if the CNCA does not perform a stop request.

iii. Bit 12: "1" if the CNCA performs a StopCNCA request (see below).
"0" if the CNCA does not perform a StopCNCA request.

Other bits are currently unused.

❖ CNCAClear:

A function keyword to clear and reset the entire CNC FIFO A.

This keyword is not allowed during motion.

CNC motions handling and related keywords

- CNC motion will start like any other motion using the message (function keyword) Begin.
- CNCA motion will be started if the motion mode is 11. Of course, the motor must be enabled, as for any other motion mode.
- Future CNCB motion (when will be supported for controllers with multiple CNC FIFOs) will have a separate motion mode. The description below focuses on CNCA.
- All axes to be participating in the CNCA motion should be enabled (motor on) and with MotionMode=11 when the Begin message is sent.
- The Begin message can be sent to any of the member (participating) axes and need to be sent only once.
- Upon getting the Begin message, and if MotorOn is 1 and MotionMode is 11, the controller will scan all other axes of the controller and will look for all axes that satisfy both conditions: motor is enabled, and motion mode is 11.
- All these axes (the one to which Begin was sent and all other that satisfy the conditions) will be marked as the participating (members) axes.

- The list of participating axes (members) cannot be modified after the Begin message. It can be modified only when sending the Begin to the next motion.

- If all is OK (for example CNCA FIFO is not empty …), the CNC motion will start:

  - ❖ The MotionStat of each of the participating axes will be set to "In Motion".

  - ❖ In addition, a dedicated bit (bit 10: CNCA_MEMBER_BIT) in this MotionStat will indicate that this axis is CNCA member.

  - ❖ The member axes, as marked upon Begin message, will be stored at: CNCAStatus[11] (CNC_STATUS_CNC_FIFO_MEMBERS_AXES_INDEX) with a bit for each axis. Axis A is bit 0, axis B is bit 1 and so on (up to 8 axes).

  While an axis can be a member of a CNCA motion (and this is reflected at its own MotionStat and at CNCAStatus[10], as explained above), it can be that only a subset of this members group will be involved at a given segment. The following definition is relevant for this status:

  - ❖ CNCAStatus[12] (CNC_STATUS_CNC_FIFO_INVOLVED_AXES_INDEX) is used to indicate, at real time, which axes are involved in the currently active segment, during CNC motion. Each axis has a related bit, similar to CNCAStatus[11] as explained above.

  - ❖ During a CNC motion, the MotionStat of each participating axis will identify that this axis is in motion and a member of the CNC motion (explained above). However, another bit in MotionStat (bit 11: CNCA_INVOLVED_NOW_BIT) will reflect if this axis is involved in the currently active motion segment (the axes involved in each specific segment are part of the segment definition).

  - ❖ Motion will start using the data from the CNCA FIFO, with details as described below.

  - ❖ If motion is ended due to any reason, the MotionStat of all involved axes will go back to zero. The "end of motion reason" (MotionReason status keyword of each member axis) will be set accordingly.

- Stopping a CNC motion:

  A CNC motion will stop upon reaching the last segment in the CNC FIFO or upon any of the following methods/events:

  - ❖ "StopCNCA": stops the motion by defining the currently executed motion segment as the last segment. The end speed of this segment is forced to zero. Can be applied on any axis that is a member. All member axes will be affected. The motion will end at the end of the currently executed motion segment.

  - ❖ "Abort" to any of the member axes aborts all member axes at the same time. The motion ends immediately (without any deceleration).

  - ❖ "Stop" to any of the member axes starts deceleration to zero of along the CNC path and the motion ends when the CNC motion profile reaches zero velocity.

  - ❖ Stopping or aborting by discrete input (DInMode[]) is also supported.

- ❖ When a Member axis is moving into its RLS or FLS (Reverse/Forward Limit Switches), the CNCA motion decelerates, along the CNC path, using the CNCA emergency deceleration (CNCAEmrgDec). The motion ends when the CNC motion profile reaches zero velocity.

- ❖ When a member axis is moving into its RevPLim or FwdPLim (software position limits), the position reference of this specific member axis is limited to RevPLim (or FwdPLim, as relevant) and the CNCA motion decelerates, using the CNCA emergency deceleration (CNCAEmrgDec). The motion ends when the CNC motion profile reaches zero velocity. Note that the deceleration is not along the CNC path since one of the axes remains at its software limit position (RevPLim or FwdPLim).

All member axes motion statuses (MotionStat), as well as the end of motion reason (MotionReason) and the CNCAStatus[10] are updated accordingly to reflect the status of the motion and the reason why the motion was ended.

- In case of a fault:

  If one of the member axes, during motion, is disabled due to user command (MotorOn = 0) or due to any fault, all member axes are disabled and the ConFlt (reason of disabling the motor) is set properly (the faulted axis with the reason of the fault and all other members with the reason of "another member axis had a fault").

  The CNC motion of course ends immediately.

Other CNC related keywords

- CNCAAccel, CNCASpeed, CNCADecel, CNCAEndSpeed, CNCAAbsTrgt, CNCAPosRef, CNCAdPosRef:

  These parameters are read only parameters.

  They reflect the vector motion profiler's parameters that are used to calculate the vector motion along the currently in-motion segment.

  The CNCAAbsTrgt is, for example, the distance to move, at the currently active segment, along the CNC path.

  The CNCAPosRef is the current desired position along the CNC path (starting from zero and reaching CNCAAbsTrgt at the end of the segment).

  CNCAdPosRef is the derivative of CNCAPosRef, meaning the current vector velocity of the profiler.

  CNCASpeed represent the desired vector speed along the CNC path, for the currently active segment. However, the actual vector speed is this value multiplied by speed factor defined as part of the CNC FIFO definition and by a second factor defined on-the-fly by the user (CNCAPercents keyword).

  These factors also affect the actual acceleration, deceleration and end speed of each segment, meaning that CNCAAccel, CNCADecel and CNCAEndSpeed are also only the nominal values for this segment (before multiplications by the factors).

- CNCAEmrgDec:

  The CNC vector deceleration to use in case of emergency stop (hitting hardware Reverse/Forward Limit Switches, or reaching software RevPLim/FwdPLim position limits).

- CNCAPause

  When set to "1", the CNC motion decelerate to zero vector velocity.
  When set to "0", the CNC motion is performed normally. If it was paused, it accelerates back to the desired vector speed of the active segment.

- CNCAPercents:

  Used by the user to scale the CNC speed (and acceleration/deceleration) along the CNC path. By affecting the speed and acceleration/deceleration, the CNCAPercents actually affects the duration of time that will be needed to perform the CNC motion.

  A value of 100 (%) means that the motion will be according to the values defined in the CNC FIFO segments. A value of 50 (%), for example, means that the CNC will be performed at twice the time that was needed to perform the nominal CNC motion as defined in the CNC FIFO.

  CNCAPercents can get values higher than 100 (%).

  CNCAPercents can be modified at any time, including on-the-fly during the CNC motion.

  User need to consider what value to give to CNCAPercents before starting a CNC motion.

  Note:

  Starting from version 1.3.0.1-165 (implementation of CNC Jerk) and if CNCAJerk is 10 or higher (see below): the value of this CNCAPercents parameter is only used at the beginning of the segment. Any change during a segment will have no effect and will affect the motion only from the next motion segment.

- CNCAJerk:

  CNCAJerk is used only from FW version 1.3.0.1-165 and newer. It was not used in earlier versions, in which it meant to have a different meaning (Smoothing, as in normal Point to Point motions in Agito controller).

  From this 1.3.0.1-165 FW version, the CNCAJerk is being used and it is calculated as a true Jerk value (derivative of Acceleration).

  The actual Jerk used in the motion calculations is CNCAJerk multiplied (internally) by 1000 !

  As a result, if, over a Terminal, communication or user program, if CNCAJerk is equal to X (on assignment or in inquiry), the actual value of Jerk that is used in the trajectory calculations is 1000*X.

  An exception is the PC Suite CNC tool, in which (as part of the parameters of CNC Vector Parameters segment), in which the CNC Jerk is presented (or typed in) with its true value. The PC Suite, in this case, will perform the scaling o 1000 before writing to (or after reading from) the controller.

  The CNCAJerk is a read only parameter. Its value is set using the CNC Vector Parameters segment (see later int his document).

  Note:

  1. CNCAJerk has the range of 10 to 100,000,000 (actual value of 10,000 to 100,000,000,000).
  2. If CNCAJerk is set to any value below 10 (0 to 9), it will have no effect and the trajectory will be calculated without Jerk (or smoothing) and will use the original trajectory equations as used in earlier FW versions!). This enables backward compatibility.

- CNCAEncRatio:

  This parameter is axis related and it should be separately set for each member axis.

  The parameter defines the ratio between the resolution of an axis to the other axes, to allow accurate CNC calculations for non-identical physical resolutions of the member axes.

This is a future feature (the controller supports this parameter, but it has no effect on the CNC calculations).

Currently, the CNC motion calculations assume identical resolution for all member axes.

CNC Step Mode

The user can define a CNC Step Mode when executing a CNC motion. When this mode is enabled, the controller will perform the CNC segments one-by-one, halting (temporarily) the CNC motion after each segment and waiting for user command to execute the next segment.

The related keywords to use this mode ae:

- CNCAStepMode:

  This keyword is a Parameter keyword. Non-axis related. Not saved to Flash.
  Possible values: 0 or 1.
  Default value after power on or reset: 0.

  The value of CNCAStepMode can be written at any time, even during motion.

  When 0, the CNC motion will act normally (no step mode).

  When 1, the CNC engine will halt at the end of each segment and will wait for CNCADoStep = 1 (see below) to perform the next segment (and halt again at the end of that segment).

  The End Speed of each segment is forced to 0, even if a different value is defined as part of the segment definition.

  Any user command to stop the motion (StopCNCA, Stop, Abort) will force CNCAStepMode parameter to 0.

  Note that CNCAStepMode can be modified while the controller is in motion. So, user can enter the step mode at any time (the controller will halt at the end of the currently executed segment) by setting this parameter to 1. On the other hand, user can leave the step mode and the controller will continue to freely execute the NC segments by writing 0 to this parameter.

- CNCADoStep:

  This keyword is a Parameter keyword. Non-axis related. Not saved to Flash.
  Possible values: 0 or 1.
  Default value after power on or reset: 0.

  The value of CNCADoStep can be written at any time, even during motion.

  If CNCA motion is active, and if the CNC is in step mode (CNCAStepMode=1, see above), then setting CNCADoStep to 1 will instruct the controller to continue to the next step.

  The value of CNCADoStep has no effect beside the conditional effect as described above.

  Once CNCADoStep is set to 1, and once the controller reacts to this request and move to the next segment (only after reaching the end of the current segment), the controller clears CNCADoStep to 0, to ensure it will not perform more than a single segment.

  Upon beginning a CNCA motion (using the Begin message), the value of CNCADoStep is automatically set as follows:

  - If CNCAStepMode is 0, CNCADoStep is cleared to 0 as well (to be ready for activating the step mode during the motion).
  - If CNCAStepMode is 1, CNCADoStep is set to 1, to ensure the first segment is executed.

The PC Suite provides suitable user interfaces for these two keywords, to easily enable usage of this step mode.

CNC Segment types and parameters

Every CNC segment that is pushed into the CNC FIFO is defined by its Type. The segment's type is first pushed into the CNC FIFO (using the CNCAPushType keyword), together with the involved axes for this segment (the axes that this segment refers to), as described above.

Once the segment type and involved axes have been pushed, the CNCAPushParam keyword is used to push the required parameters.

Each Segment's type requires a different number of parameters and within each segment type, different number of involved axes can also affect the number of parameters.

The exact number of parameters needs to be pushed, and with the exact required order.

The following sections describe the list of supported segment's types, as well as the meaning of each segment type. Also provided is the number/meaning/order of the parameters that need to be pushed for this type of segment.

Note:

Some of the segment's types are motion segments. Some of the segment's types just define general parameters or provide definitions. Those (non-motion) segments do not consume time within the CNC motion trajectory. When the CNC engine reaches such segment, it is executed, and the next segment is immediately loaded (without waiting to the next control cycle).

Type 1 - Linear Motion segment

In this segment type, a linear, synchronized, multi-axes vector motion is performed for up to 4 axes (future firmware will support up to 6 axes).

The user defines the target position (absolute value) for each participating (involved) axis, as well as the desired vector speed and the desired end vector speed at the end-point of the segment.

General parameters such as vector acceleration, vector deceleration, vector jerk, and global vector speed factor are taken from the latest executed Vector Parameters segment (see below).

Segment type value: 1
Number of involved axes: 1 – 5
Parameters (left most parameter is pushed first):

One involved axis	=> TargetPos1, Speed, End Speed
Two involved axes	=> TargetPos1, TargetPos2, Speed, End Speed
Three involved axes	=> TargetPos1, TargetPos2, TargetPos3, Speed, End Speed
Four involved axes	=> TargetPos1, TargetPos2, TargetPos3, TargetPos4, Speed, End Speed
Five involved axes	=> TargetPos1, TargetPos2, TargetPos3, TargetPos4, TargetPos5, Speed, End Speed

Trying to push wrong number of involved axes or wrong number of parameters will result with an error during the push sequence.

Type 2 - Arc Motion segment

In this segment type, an arc, synchronized, dual-axes vector motion is performed for exactly 2 axes.

The arc can be a fraction of a full circle, a full circle and multiple number of circles plus, optionally, a fraction of a circle.

User can define if the motion between the start point to the end-point would be at the CW or CCW direction.

The user defines the target coordinate (absolute value) for the two participating axes, the center coordinates of the arc, the direction of the motion (CW or CCW), as well as the desired vector speed and the desired end vector speed at the end-point of the segment.

In addition, the user defines how many full circles to add to the motion.

Finally, two dummy parameters must be pushed (value 0) for internal reasons.

General parameters such as vector acceleration, vector deceleration, vector jerk, and global vector speed factor are taken from the latest executed Vector Parameters segment (see below).

Segment type value: 2
Number of involved axes: 2 (exactly)
Parameters (TargetPos1 is the first one to push, Dummy2 is the last one to push):

TargetPos1, TargetPos2,
Center1, Center2, Dir,
Speed, End Speed, Additional Cycles,
Dummy1, Dummy2

For the "Dir" parameter: a value of 0 means CCW. A value of 1 means CW. Any other value is illegal.

The distance from the segment start point (end point of the previous segment) to the center coordinates must be identical to the distance from the target coordinate to the center coordinates (both are the radius of the arc motion).

Note – Definition of CW and CCW directions:

The Arc segment consists of two involved axes. The 1st involved axis can be considered as "X" direction while the 2nd involved axis can be considered as "Y". With this XY definition, CCW direction means that X direction is rotating toward Y direction and CW direction is the opposite direction.

Type 3 - Helix Motion segment

Will be supported in future firmware versions.

Type 4 – Delay segment

When reaching this type of segment, the CNC engine will wait, in-position of all member axes, for a delay time defined, in msec, as the single parameter of this segment type.

Segment type value: 4
Number of involved axes: 0 (exactly)
Parameters:

DelayTime

Type 5 – Set Vector Parameters segment

This segment is used to define (set) the general parameters that are common to all relevant segments type.

The vector parameters that are defined within this segment are used for all following segments (as applicable), until the next Set Vector Parameters segment is encountered.

It is highly recommended to include a "Set Vector Parameters" segment at the beginning of a CNC FIFO motion, to ensure that the vector parameters are well defined.

Segment type value: 5
Number of involved axes: 0 (exactly)
Parameters:

SpeedPercentsInternal, VectorAcceleration, VectorDeceleration, VectorJerk

SpeedPercentsInternal: in [%]. An internal factor that is applied on the vector motion parameters in order to slow down (or accelerate) the time scaling of the motion.

This factor is in addition to the factor that is defined by the user: CNCAPercents.

If, for example, a given segment includes a definition of: Speed = 1000000, the actual vector speed will be:

Actual speed = 100000 * CNCAPercents/100 * SpeedPercentsInternal/100

The acceleration and deceleration parameters are also properly scaled to keep the motion time proportional to the user's and internal speed factors.

The VectorJerk parameter is used to assign the CNCAJerk parameter. Note that the internal trajectory calculations will use (CNCAJerk*1000) as the actual Jerk value.

Note:

If VectorJerk is set to a value in the range of 0 to 9, it means backward compatibility mode (compatible to FW versions earlier than 1.3.0.1-165), where this parameter has no effect. The trajectory calculations are identical to those of versions earlier to 1.3.0.1-165.

Type 6 – Set Corner Parameters segment

This segment is used to define (set) the general parameters that are common to all types of Automatic Corner Motion segments.

The corner parameters that are defined within this segment are used for all following segments (as applicable), until the next Set Corner Parameters segment is encountered.

It is highly recommended to include a "Set Corner Parameters" segment at the beginning of a CNC FIFO motion, to ensure that the corner parameters are well defined. Of course, it can be repeated later as many times as required.

Segment type value: 6
Number of involved axes: 0 (exactly)
Parameters:

CornerType, CornerRadiusMethod, CornerRadiusOrError, CornerAxisAcceleration, MinimalAngleForCorner, AxisAccelerationLimitType

CornerType: 1 for ARC (round) corner.
2 for Continuous Acceleration corner.

Refer to the corners' types chapter for additional explanations about the differences between the two types and about the relevant parameters per each type, as listed below.

CornerRadiusMethod: Relevant for CornerType 1 (round corners) only. Push zero if the corner type is set to Continuous Acceleration.

0 for specifying directly the radius of the corner.

1 for specifying the maximal error due to the corner, instead of its radius (the radius will eb calculated by the controller, based on the angle of the corner).

CornerRadiusOrError: Used only for ARC (round) corners type. Push zero if the corner type is set to Continuous Acceleration.

Its meaning depends on the value of CornerRadiusMethod.

If CornerRadiusMethod == 0:
Radius (in counts) of the corner.

If CornerRadiusMethod == 1:
Maximal error due to the corner (in counts).

Must be positive (not zero) for ARC type.
Must be zero for Continuous Acceleration type.

CornerAxisAcceleration: The nominal peak acceleration (counts/sec^2) of each axis for Continuous Acceleration corner. Push zero if the corner type is set to ARC.
Must be zero for ARC type.
Must be positive (not zero) for Continuous Acceleration type.

MinimalAngleForCorner: Defines the minimal angle between two linear segments to create a corner, in degrees. If the angle between the two consecutive lines is smaller (or equal!) than this value, the controller will not execute a corner between the two lines (meaning, there will be some sharp change in the reference velocity of each of the axes).
Must be in the range of [0, 180].

AxisAccelerationLimitType: 0 for avoid applying corner speed reduction for axis acceleration limitation.
1 for limiting Automatic Corner Motion segment speed according to the user provided acceleration limitation per axis. See more details below, within this document.

Please note that corners are added only if the user pushed an Automatic Corner Motion segment between the two relevant linear motion segments. If an Automatic Corner Motion segment was not pushed, the controller will not add a corner!

See more under the description of the Corner Motion segment, below.

Default values:

Upon starting a CNC motion and till encountering a "Set Corners Parameters" segment, the internal default values of the above parameters are:

CornerType:	1
CornerRadiusMethod:	0
CornerRadiusOrError:	1000
CornerAxisAcceleration:	0
MinimalAngleForCorner:	5
AxisAccelerationLimitType:	0

The PC Suite suggests this segment when starting a new CNC trajectory list (with the above default values) and forces this segment to appear at least once, as the 2nd or 3rd segment.

Type 7 - Set Initial Positions segment

In this segment type, the user defines the expected absolute position of all member axes when the CNC motion will start. This segment must appear, if used, as the first segment that is pushed into the CNC FIFO and must appear only once per CNC FIFO motion.

This type of segment is required in order to enable the CNC engine to perform many of the required calculations (and error detections) once the CNC segments are pushed without waiting to the Begin message to start the motion itself.

The segment must include the definitions of the start positions for all member axes (not more, not less). In case a wrong number of involved axes (and parameters) is defined, or in case there is no match between the specific involved axes of this segment and the member axes of the CNC motion, the Begin message will fail with a proper error.

The number of member axes is currently limited to four. Future firmware versions will support up to six member axes.

This segment is needed since the whole CNC FIFO motion is based on fixed definitions of absolute positions (as start point and target point for each segment). Future firmware versions will support a mode to define all motions as relative motions (or mixed absolute and relative) and, for relative motions, this "Set Initial Positions" segment will not be required.

Please refer to the chapter "First segments of a CNC motion trajectory" below for rules related to the "Set Initial Positions" segment type.

Segment type value:	7
Number of involved axes:	1 – 5
Parameters:	

One involved axis	=> StartPos1
Two involved axes	=> StartPos1, StartPos2
Three involved axes	=> StartPos1, StartPos2, StartPos3
Four involved axes	=> StartPos1, StartPos2, StartPos3, StartPos4

Five involved axes => StartPos1, StartPos2, StartPos3, StartPos4, StartPos5

Type 8 – Write Discrete Output Port segment

Using this type of segment, the user can force a value at the discrete output port (DOutPort parameter) at the specific time when the CNC motion arrived to this segment.

Segment type value: 8
Number of involved axes: 0 (exactly)
Parameters:

AxisIdentification, DOutPortValue

AxisIdentification: Value of 0 refers to A axis, value of 1 refers to B axis, and so on...
DOutPortValue: Overwrite the value of DOutPort when this segment is reached.

Type 9 – Write to User Array segment

Using this type of segment, the user can force an assignment of a user defined GenData[] or UserParam[] element with a user defined value at the specific time when the CNC motion arrived to this segment.

Segment type value: 9
Number of involved axes: 0 (exactly)
Parameters:

ArraySelection (GenData[] or UserParam[]), AxisIdentification (used for UserParam[] only), IndexValue, Value

ArraySelection: Value of 0 refers to GenData[], value of 1 refers UserParam[].
AxisIdentification: Value of 0 refers to A axis, value of 1 refers to B axis, and so on...

When reaching this segment, the CNC engine will perform:

GenData[IndexValue] = Value
Or:
<AxisValue>UserParam[IndexValue] = Value

Type 10 – Relative/Absolute Mode segment

Will be supported in future firmware versions.

Type 11 – Set Coordinates Offsets segment

Will be supported in future firmware versions.

Type 12 – Wait Using User Array segment

When reaching this type of segment, the CNC engine will wait, in-position of all member axes, till the trigger condition as defined by the segment's parameters is satisfied.

Segment type value: 12
Number of involved axes: 0 (exactly)
Parameters:

ArraySelection (GenData[] or UserParam[]), AxisIdentification (used for UserParam[] only), IndexValue, TriggerType, Trigger Value

ArraySelection: Value of 0 refers to GenData[], value of 1 refers UserParam[].
AxisIdentification: Value of 0 refers to A axis, value of 1 refers to B axis, and so on...

TriggerType:

0: None (CNC engine will not wait – see note below)
1: Greater than
2: Equal To
3: Not Equal To
4: Smaller Than
5: Rising Edge
6: Falling Edge
7: Reserved (will act as TriggerType = 0, see above)
8: Upon change (The Trigger Value parameter is ignored – although it must be included in the pushed segment).

Note:

The "Wait Using User Array" segment will create a segment which will halt the CNC segments execution for at least 1 sample time (even if the wait trigger condition is met immediately or even if the TriggerType is None.

Type 13 - Set (Assign) Positions segment

In this segment type, the user can set the current position of the member axes (or sub group of them) to a user defined value.

The CNC engine will consider these new values for the CNC calculations (while pushing the segments) and will also set the controller positions to this value, once reaching this segment during the CNC motion.

Notes:

1. All involved axes must not be moving at this time (end speed of previous motion segments must be 0). Unexpected results can happen if trying to assign position value to a moving axis.

2. This segment will have no effect (will not set the position) for those member axes that do not satisfy the conditions of SetPosition. Refer to the SetPosition keyword manual page.

The number of member axes is currently limited to four. Future firmware versions will support up to six member axes.

Segment type value: 13
Number of involved axes: 1 – 5
Parameters:

One involved axis	=> SetPos1
Two involved axes	=> SetPos1, SetPos2
Three involved axes	=> SetPos1, SetPos2, SetPos3
Four involved axes	=> SetPos1, SetPos2, SetPos3, SetPos4
Five involved axes	=> SetPos1, SetPos2, SetPos3, SetPos4, SetPos5

Type 14 – Automatic Corner Motion segment

(We recommend reading "Keeping continuous motion" chapter below, before reading this chapter).

In this segment type, a corner, synchronized, dual-axes vector motion is performed for exactly 2 axes.

The corner can be of one of two types: ARC (round corner) or Continuous Acceleration corner. Please refer to Appendix A for detailed description of each corner.

The type and the parameters of the corner are defined by the latest executed "Set Corner Parameters" segment (see above).

The user does not need to calculate the corner or to provide specific data per each Automatic Corner Motion segment. The user just pushes an empty corner motion segment and the controller will perform the required calculations once the next motion segment will be pushed (assuming it will be a Linear Motion segment of the same axes).

Segment type value: 14
Number of involved axes: 2 (exactly)
Parameters (Dummy1 is the first one to push):

Dummy1, Dummy2, Dummy3 … to Dummy10 (all 10 dummy parameters must be 0).

Tests performed by the controller once such segment is pushed:

1. That there are 10 parameters, and all are 0.
2. That the last motion segment is Linear Motion.
3. As any other motion segment, that the involved axes are identical to the involved axes of the previous motion segment.

Corner rules and calculations:

1. An empty corner is a corner with all 10 dummy parameters zero.
2. When executing the CNC motion, if an empty Automatic Corner Motion segment is encountered, it is just ignored (skipped).
3. Once pushing a Linear Motion segment and if the previous motion segment was an Automatic Corner Motion segment, then the controller calculates the corner characteristics (based on data of the Linear Motion segments before and after the corner and data of the latest executed Set Corner Parameters segment).

    a. The controller will change the definitions of the two linear segments (they will become shorter because of the corner).
    b. The controller will perform some internal calculations regarding the corner and place the results instead of the dummy parameters of the corner. From this time, this corner is no more an empty corner and once the CNC will be executed, the controller will perform the corner.
    c. Under some conditions (see Appendix A for details), the controller can decide to keep the Automatic Corner Motion segment empty, meaning to avoid placing a corner.

4. Note that if a corner is pushed (can be done only if the last motion segment was a Linear Motion segment) and then a none Linear Motion segment is pushed, the corner will remain as empty corner and will just be skipped during the CNC motion execution.

5. All the three relevant segments (Linear Motion, followed by the Automatic Corner Motion, followed by Linear Motion) must have exactly two involved axes, which must be the same (same axes, same order). This is not a special test as it is actually tested as part of the continuous motion conditions.

6. The corner speed is constant and is assumed to be (not need to set it) identical to the End Speed of the previous segment.

7. If the user defined to limit the corner speed in order to withstand the individual acceleration limitation of each axis (see Set Corner Parameters segment), the controller may change (reduce) the End Speed of the previous segment in order to avoid violation of the maximal axis acceleration per each of the involved axes (see the chapter about "Set Axes Maximal Accelerations" segment).

A new CNCAStatus[] entry was added to hold the counter of the number of such cases. It will be cleared to 0 upon power on or reset. It will be cleared to zero also upon CNCAClear message.

It counts the number of segments at which the controller found violation of axis acceleration and as a result made a change to the End Speed of the previous segment and to the speed of the corner itself.

Note:

The calculation considers the speed corner (End Speed of the previous segment) and the internal CNC speed factor as defined in the latest type 5 (Set Vector Parameters segment) segment.(parameter: SpeedPercentsInternal).
The calculation, however, ignores (assumes it is 100%) the user parameter CNCAPercents, since this parameter can be modified on the fly during the motion itself (while the calculation itself is done once the corner is pushed to the FIFO). As a result, if this parameter (CNCAPercents) will be set to a value higher than 100% during the motion, the acceleration limits of the involved axes can be exceeded. And if it will be set to a value lower than 100%, the corners speeds may be lower than what is possible with the actual vector motion speed.

Type 15 – Set Maximal Velocity Jump Parameters segment

This segment is used to define (set) the general parameters that are common to all cases where a Linear Motion segment is followed by another Linear Motion segment (without an Automatic Corner Motion segment between them) and the End Speed of the first linear segment is not 0.

The maximal velocity jump parameters that are defined within this segment are used for all following segments (as applicable), until the next Set Maximal Velocity Jump Parameters segment is encountered.

It is highly recommended to include a "Set Maximal Velocity Jump Parameters" segment at the beginning of a CNC FIFO motion, to ensure that the maximal jump parameters are well defined. Of course, it can be repeated later as many times as required.

Segment type value: 15
Number of involved axes: 1 – 5
(must be identical to the number of CNC member axes, otherwise error)

Parameters:

One involved axis	=> MaxDV1, VelocityJumpMode
Two involved axes	=> MaxDV1, MaxDV2, VelocityJumpMode
Three involved axes	=> MaxDV1, MaxDV2, MaxDV3, VelocityJumpMode
Four involved axes	=> MaxDV1, MaxDV2, MaxDV3, MaxDV4, VelocityJumpMode
Four involved axes	=> MaxDV1, MaxDV2, MaxDV3, MaxDV4, MaxDV5, VelocityJumpMode

All MaxDV parameters are in counts/sec (positive, including 0).

VelocityJumpMode:
- 0: ignore the MaxDV parameters and perform the motion as defined by the segments pushed by the user.
- 1: Use the MaxDV parameters as defined below.

When VelocityJumpMode is 1, the CNC engine will perform the following modifications to the user defined CNC motion:

1. For any case of two consecutive Linear Motion segments (transparent non-motion segments are ignored) only, independent of the number of participating axes but only if the number of the participating axes and their order is identical for both of the Linear Motion segments.

2. The velocity change of each participating axis (considering the angles of the two linear segments and the End Speed of the first linear segment) at the point between the two segments is calculated.

3. This change, per axis, is compared to the relevant MaxDV parameter.

4. If required, the End Speed is reduced (smallest possible change) to enforce that none of the axes will be subjected to a velocity jump greater than its MaxDV parameter.

A new CNCAStatus[] entry was added to hold the counter of the number of such cases. It will be cleared to 0 upon power on or reset. It will be cleared to zero also upon CNCAClear message.

It counts the number of linear segments' junctions at which the controller found violation of axis velocity jump and as a result made a change to the End Speed of the first segment.

Note:

The calculation considers the End Speed of the first segment and the internal CNC speed factor as defined in the latest type 5 (Set Vector Parameters segment) segment.(parameter: SpeedPercentsInternal).
The calculation, however, ignores (assumes it is 100%) the user parameter CNCAPercents, since this parameter can be modified on the fly during the motion itself (while the calculation itself is done once the segments are pushed to the FIFO). As a result, if this parameter (CNCAPercents) will be set to a value higher than 100% during the motion, the velocity jump limits of the involved axes can be exceeded. And if it will be set to a value lower than 100%, the velocity jumps speeds may be lower than what is possible with the actual vector motion speed.

Default values:

Upon starting a CNC motion and till encountering a "Set Maximal Velocity Jump Parameters" segment, the internal default values of the above parameters are:

VelocityJumpMode: 0
All MaxDV parameters: 200,000,000

The PC Suite suggests this segment when starting a new CNC trajectory list (with the above default values) and forces this segment to appear at least once, as the 3rd or 4th segment.

Important note:

Before FW version 1.3.0.1-165:

This algorithm will not act on "junctions" (End Speed) between two linear motion segments unless at least 3 linear motion segments, all with the same involved axes exactly, are being pushed (non-motion-blocking segments can be pushed in between). The algorithm will first modify the End Speed (if necessary) between the 2nd and the 3rd linear motion segment (and any linear motions segments to follows, if all share the same involved axes).

Any change of the involved axes (even in order) or any non-linear motion segment, will reset the counting and next linear motion segments will have to count three segments again before correction as applied.

The End Speed of these first segments (some of them or all of them) can be 0.

From FW version 1.3.0.1-165 and newer:

The above restriction is now less demanding. Now, this algorithm will work on a "junction" (End Speed) between two consecutives ((transparent non-motion segments are ignored) linear motions segments at the following conditions:

1. Number of the participating axes and their order is identical for both of the Linear Motion segments.

2. Per each of the participating axes, the start position of the 1st linear motion segment (the 1st of the two segments) is known.

Type 16 – Set Axes Maximal Accelerations segment

This segment is used to define (set) the maximal acceleration allowed for all (or subgroup) of the member axes during Automatic Corner Motion segments.

The values defined in this segment will be used only If the user specified to limit the corner speed in order to withstand the individual acceleration limitation of each involved axis (see Set Corner Parameters segment, parameter: AxisAccelerationLimitType).

The maximal acceleration parameters that are defined within this segment are used for all following segments (as applicable), until the next Set Axes Maximal Accelerations segment is encountered (and if a new value is defined for the given axis).

It is highly recommended to include a "Set Axes Maximal Accelerations" segment at the beginning of a CNC FIFO motion, involving all member axes, to ensure that the maximal values are well defined. Of course, it can be repeated later as many times as required, each time for any subgroup of the member axes.

Segment type value: 16
Number of involved axes: 1 – 5

Parameters:

One involved axis	=> MaxAcc1
Two involved axes	=> MaxAcc1, MaxAcc2
Three involved axes	=> MaxAcc1, MaxAcc2, MaxAcc3
Four involved axes	=> MaxAcc1, MaxAcc2, MaxAcc3, MaxAcc4
Five involved axes	=> MaxAcc1, MaxAcc2, MaxAcc3, MaxAcc4, MaxAcc5

All MaxAcc parameters are in counts/sec^2 (positive, not including 0).

Refer to the chapters about Automatic Corner Motion segment fro detailed description of the axes acceleration limitation.

Default values:

Upon start pushing segments for a CNC motion and till encountering a "Set Axes Maximal Accelerations" segment, the internal default values of the above parameters are (for all axes):

All MaxAcc parameters: Maximal acceleration allowed for the controller (typically 2000000000)

Type 17 – Multiple Write to User Array segment

Using this type of segment, the user can force multiple assignments of user defined GenData[] or UserParam[] elements with user defined values at the specific time when the CNC motion arrived to this segment.

Segment type value: 17
Number of involved axes: 0 (exactly)
Parameters:

ArraySelection (GenData[] or UserParam[]), AxisIdentification (used for UserParam[] only), IndexValue, Value1, Value2, Value3 and Value4

ArraySelection: Value of 0 refers to GenData[], value of 1 refers UserParam[].
AxisIdentification: Value of 0 refers to A axis, value of 1 refers to B axis, and so on…

When reaching this segment, the CNC engine will perform:

GenData[IndexValue] = Value1
GenData[IndexValue+1] = Value2
GenData[IndexValue+2] = Value3
GenData[IndexValue+3] = Value4
Or:
<AxisValue>UserParam[IndexValue] = Value1
<AxisValue>UserParam[IndexValue+1] = Value2
<AxisValue>UserParam[IndexValue+2] = Value3
<AxisValue>UserParam[IndexValue+3] = Value4

Type 18 – Multiple Write to User Array and Wait Using User Array segment

When reaching this type of segment, the CNC engine will first assign the multiple values to GenData[] or UserParam[] (see the description for Type 17 above, it is done at the same way) and then wait, in-position of all member axes, till the trigger condition as defined by the segment's parameters is satisfied.

Segment type value: 18
Number of involved axes: 0 (exactly)
Parameters:

ArraySelection (GenData[] or UserParam[]), AxisIdentification (used for UserParam[] only), IndexValue, , Value1, Value2, Value3, Value4, TriggerIndexValue, TriggerType, Trigger Value

ArraySelection: Value of 0 refers to GenData[], value of 1 refers UserParam[].
AxisIdentification: Value of 0 refers to A axis, value of 1 refers to B axis, and so on…

When reaching this segment, the CNC engine will perform:

GenData[IndexValue] = Value1
GenData[IndexValue+1] = Value2
GenData[IndexValue+2] = Value3
GenData[IndexValue+3] = Value4
Or:
<AxisValue>UserParam[IndexValue] = Value1
<AxisValue>UserParam[IndexValue+1] = Value2
<AxisValue>UserParam[IndexValue+2] = Value3
<AxisValue>UserParam[IndexValue+3] = Value4

And then will wait according to the trigger definitions, using the same selected array (GenData[] or UserParam[]) but with the index of TriggerIndexValue.

TriggerType:

0: None (CNC engine will not wait – see note below)
1: Greater than
2: Equal To
3: Not Equal To
4: Smaller Than
5: Rising Edge
6: Falling Edge
7: Reserved (will act as TriggerType = 0, see above)
8: Upon change (The Trigger Value parameter is ignored – although it must be included in the pushed segment).

Note:

This "Multiple Write to User Array and Wait Using User Array" segment will create a segment which will halt the CNC segments execution for at least 1 sample time (even if the wait trigger condition is met immediately or even if the TriggerType is None.

First segments of a CNC motion trajectory

Starting from FW version 1.3.0, the following rules are applicable for the first segments of a CNC motion trajectory (replacing all older rules at older FW versions):

1. If the first segment is "Set Initial Positions" then there are no additional limitations.
    a. "Set Initial Position" segment is allowed only as the 1st segment.
    b. It can be avoided completely (if the start positions of the axes are not known when pushing the motion segments), but then the following rules are applicable":
2. If the first segment is **not** "Set Initial Positions", then the following limitations are applicable:
    a. The first motion segments (ignoring any proceeding non-motion segments) must be Linear Motion segment.
    b. The second motion segment can't be Automatic Corner Motion segment.
3. Relative mode is not yet supported.

The first two limitations (1 and 2) are to ensure that the controller has all required information in order ot perform all pre-calculations while the segments are being pushed to the controller.

Keeping continuous motion

The continuity rules and behavior, as described within this chapter, are relevant from formal FW version 1.3.0 and up.

The controller takes care that the pushed segments will keep continuous vector velocity when switching from any segment to the one following it. This must be kept among any type of pushed segments.

If there is discontinuity, the controller will act according to a user defined behavior, as described in detail below.

Two dedicated parameters are used to control and report the controller behavior at cases of vector velocity discontinuity:

- CNCAEndSegMod:

  Not axis related, R/W, Saved to Flash.

  Value of 0: Standard Error Mode.
  Value of 1: Force zero speed mode.

- CNCAEndErrCnt:

  Not axis related, R/W, Not saved to Flash.
  0 after power on or reset. Cleared to zero also upon CNCAClear message.
  Can be assigned to any value by the user, anytime.

  If CNCAEndSegMod is 0: No action.
  If CNCAEndSegMod is 1: Counts the number of segments at which the controller found discontinuity and as a result made a change to the segment.

All the decisions about continuity are taken when each segment is pushed into the CNC FIFO (in contrast to when the motion is reaching a given segment). This means that the outcome of the decisions (such as automatic decision regarding the End Speed of a given segment) are done as a function of the motion parameters of the segment when it is being pushed. If, for example, the user later reduces (or increases) the vector speed, the decision will not be changed, and the End Speed will remain as it is (multiplied by the user defined factor as set during the motion).

What are the rules used by the controller to check this continuous motion?

Some relevant definitions:

Motion segments are those types of segments that really performs a motion. Types: 1-3 and 14. Each motion segment has its own Speed and End Speed as part of the segment definition (exception: type 14, "Automatic Corner Motion" inherits its Speed and End Speed from the End Speed of the last Linear Motion segment).

Non-motion segments are those types of segments which do not create motion (all types except: 1-3 and 14). The Speed and the End Speed of these segments are inherited from the End Speed of the last motion segment.

The Speed and End Speed when starting a CNC motion are considered as 0.

"Motion-blocking" segments are a sub-group of the non-motion segments group and are segments in which all CNC participating axes must be in zero speed (the End Speed of the previous segment must be zero). The following segment types are "motion-blocking": 4, 7, 10, 11, 12 and 13. Note that from the above definitions, one can conclude that the Speed and End Speed of a "motion-blocking" segment are always 0.

"Non-motion-blocking" segments are segments which can be placed in the CNC FIFO without any limitation from the point of view of the End Speed of the previous segment. The following segment types are "non-motion-blocking": 1-3, 5-6, 8-9 and 14. Note that the "non-motion-blocking" segments consist of all the motion segments and of specific non-motion segments that have no effect on the motion, like type 8 ("Write To Discrete Outputs").

The sub-group of segment types that belongs both to the non-motion group and to the "non-motion-blocking" group (types: 5-6 and 8-9) are referred to as "no-motion-no-blocking" segments and they can be pushed anywhere without affecting or limiting the CNC trajectory motion calculations.

Once a segment is pushed to the controller, the controller performs this additional test (continuous motion):

1. If the segment is a "motion-blocking" segment, the End Speed of the previous segment must be 0.

   This test means that the system must be stopped (End Speed is 0) before performing any "motion-blocking" segment.

2. If the segment is a motion segment and its Involved Axes list is not identical (in contents and in order) to the Involved Axes of the last motion segment (ignoring all non-motion segments that were pushed between them), the End Speed of the last motion segment must be 0.

   This test means that if a given axis is moving (involved) at a given motion segment, the next motion segment must include this axis as well, or, alternatively, the End Speed of the given motion segment needs to be zero.

All these tests are to ensure that a motion of a given involved (participating) axis will reach 0 velocity if it does not continue to move at the next segment.

Note that the controller can't check the last segment of a CNC FIFO (what if it ends with End Speed not 0...) since the controller is not aware for a segment to be the last segment until it is executing it (too late...).

What the controller will do when a pushed segment is found to cause discontinuity with the previous segment?

- If CNCAEndSegMod == 0:

The controller will return an error and will reject the last CNCAPushParam message.

- If CNCAEndSegMod == 1:

  The controller will force the end speed of the previous segment to 0 (to ensure continuity).
  The controller will increment CNCAEndErrCnt by 1.
  The controller will accept the push action.

Note that when using the PC Suite, the PC Suite (versions later than 17 October 2018) performs these tests prior to downloading the CNC trajectory list to the controller and will generate proper error/warning messages if continuous motion conditions are not met.

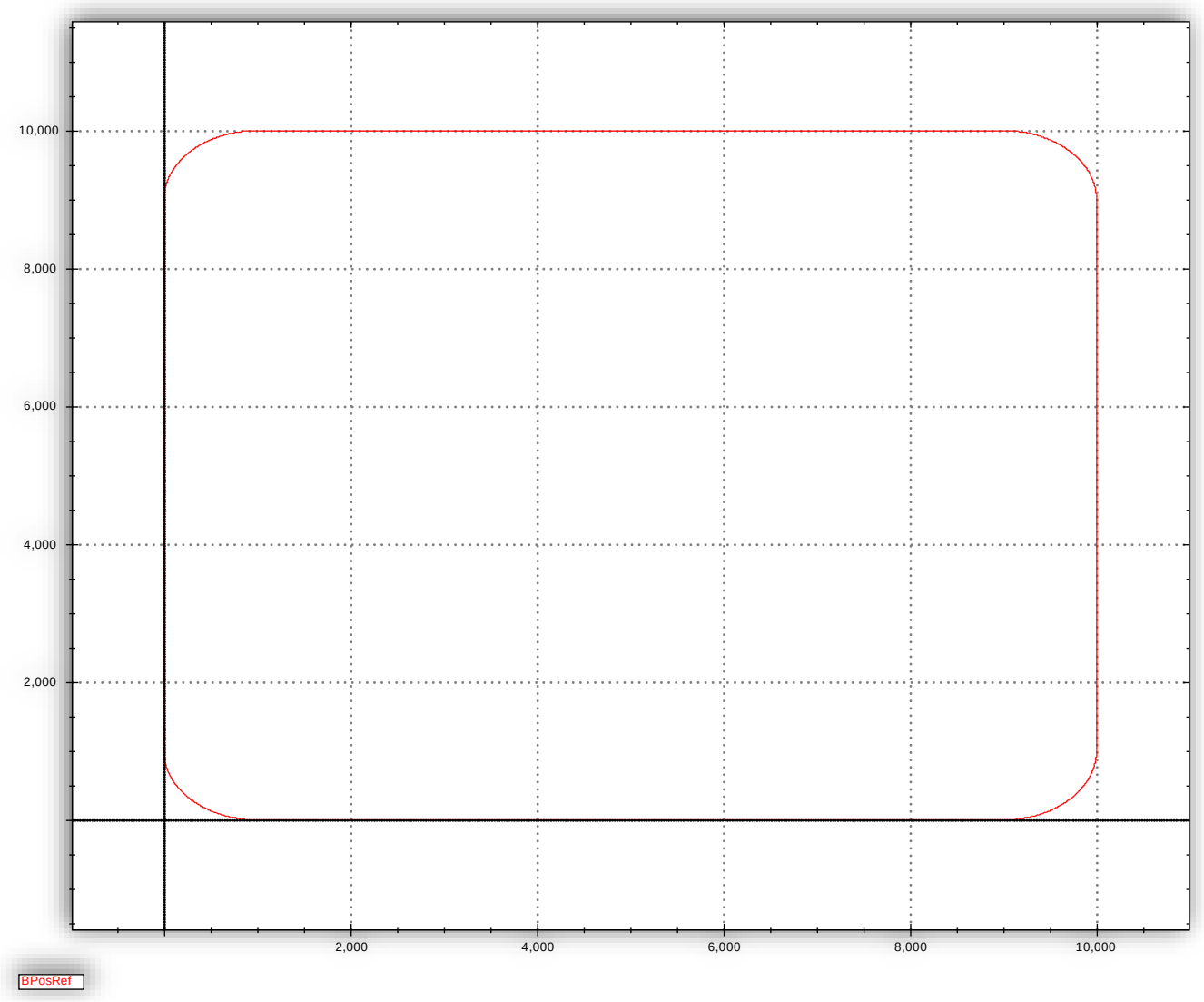
The PC Suite will provide interfaces to the above two related keywords and will perform the continuity validation (and respond to found cases of discontinuity) based on the setting of CNCAEndSegMod.

Important note:

The above tests do not protect the CNC motion trajectory from cases that the velocity reference of a given participating axis will be discontinuous. Actually, any two linear motion segments (without a pushed corner between them), unless they are along the same line, will cause a jump at the velocity reference of each of the involved axes (when a motion at a given constant speed changes its direction instantly, it means that the velocity reference of each axis is also instantly changes). However, the above protections protect the CNC trajectory from wrong motion definitions, where at least one of the involved axes commanded to abort its motion instantly.

Example – User Program for a CNC motion

The following user programs demonstrates the communication messages that needs to be sent to the controller in order to create the CNC motion as shown in the following graph:



This CNC motion uses axes A and B as members. The graph shows BPosRef (position reference of axis B) as a function of APosRef (position reference of axis A). The absolute start point of the motion is:

(A, B) = (1000, 0).

```
//
// CNC program for "SmoothedRectangle"
//
// Created:
//
//                  PC Suite version 8.0004.0002-04.00
//                  At 09 December 2017 08:03
//
// Member axes: A, B
//
//
// Encoder ratios
//
ACNCAEncRatio = 256
BCNCAEncRatio = 256
//
// Vector speed percents
//
ACNCAPercents = 100
//
// Clear CNCA FIFO
```

```
//
ACNCAClear
//
// Push Set positions, involved: A, B
// Positions: A=1000, B=0
//
ACNCAPushType,117571583
ACNCAPushParam,1000
ACNCAPushParam,0
//
// Push Vector Params.
// Percents: 100%
// Acceleration: 1000000
// Deceleration: 1000000
// Jerk: 100000000
//
ACNCAPushType,100663295
ACNCAPushParam,100
ACNCAPushParam,1000000
ACNCAPushParam,1000000
ACNCAPushParam,100000000
//
// Push Linear motion, involved: A, B
// Go to: A=9000, B=0
// Velocity: 30000
// End Velocity: 2000
//
ACNCAPushType,16908287
ACNCAPushParam,9000
ACNCAPushParam,0
ACNCAPushParam,30000
ACNCAPushParam,2000
//
// Push Arc motion, involved: A, B
// Go to: A=10000, B=1000
// Center: A=9000, B=1000
// Direction: 0 (CCW)
// Velocity: 2000
// End Velocity: 2000
// Additional circles: 0
// Dummy x 2: 0
//
ACNCAPushType,33685503
ACNCAPushParam,10000
ACNCAPushParam,1000
ACNCAPushParam,9000
ACNCAPushParam,1000
ACNCAPushParam,0
ACNCAPushParam,2000
ACNCAPushParam,2000
ACNCAPushParam,0
ACNCAPushParam,0
ACNCAPushParam,0
//
// Push Linear motion, involved: A, B
// Go to: A=10000, B=9000
// Velocity: 30000
// End Velocity: 2000
//
ACNCAPushType,16908287
ACNCAPushParam,10000
ACNCAPushParam,9000
ACNCAPushParam,30000
ACNCAPushParam,2000
//
// Push Arc motion, involved: A, B
```

```
// Go to: A=9000, B=10000
// Center: A=9000, B=9000
// Direction: 0 (CCW)
// Velocity: 2000
// End Velocity: 2000
// Additional circles: 0
// Dummy x 2: 0
//
ACNCAPushType,33685503
ACNCAPushParam,9000
ACNCAPushParam,10000
ACNCAPushParam,9000
ACNCAPushParam,9000
ACNCAPushParam,0
ACNCAPushParam,2000
ACNCAPushParam,2000
ACNCAPushParam,0
ACNCAPushParam,0
ACNCAPushParam,0
//
// Push Linear motion, involved: A, B
// Go to: A=1000, B=10000
// Velocity: 30000
// End Velocity: 2000
//
ACNCAPushType,16908287
ACNCAPushParam,1000
ACNCAPushParam,10000
ACNCAPushParam,30000
ACNCAPushParam,2000
//
// Push Arc motion, involved: A, B
// Go to: A=0, B=9000
// Center: A=1000, B=9000
// Direction: 0 (CCW)
// Velocity: 2000
// End Velocity: 2000
// Additional circles: 0
// Dummy x 2: 0
//
ACNCAPushType,33685503
ACNCAPushParam,0
ACNCAPushParam,9000
ACNCAPushParam,1000
ACNCAPushParam,9000
ACNCAPushParam,0
ACNCAPushParam,2000
ACNCAPushParam,2000
ACNCAPushParam,0
ACNCAPushParam,0
ACNCAPushParam,0
//
// Push Linear motion, involved: A, B
// Go to: A=0, B=1000
// Velocity: 30000
// End Velocity: 2000
//
ACNCAPushType,16908287
ACNCAPushParam,0
ACNCAPushParam,1000
ACNCAPushParam,30000
ACNCAPushParam,2000
//
// Push Arc motion, involved: A, B
// Go to: A=1000, B=0
// Center: A=1000, B=1000
```

```
// Direction: 0 (CCW)
// Velocity: 2000
// End Velocity: 0
// Additional circles: 0
// Dummy x 2: 0
//
ACNCAPushType,33685503
ACNCAPushParam,1000
ACNCAPushParam,0
ACNCAPushParam,1000
ACNCAPushParam,1000
ACNCAPushParam,0
ACNCAPushParam,2000
ACNCAPushParam,0
ACNCAPushParam,0
ACNCAPushParam,0
ACNCAPushParam,0
//
// Enable motors, set motion mode and start motion
//
AMotorOn=1
BMotorOn=1
AMotionMode=11
BMotionMode=11
ABegin
```

On the fly changes

The CNC motion is fully defined within the CNC FIFO, based on segments (and their parameters) that were pushed into the FIFO by the user.

Yet, the user can perform few changes during the motion without changing the FIFO itself, as follows:

- CNCAPercents keyword:

  Can be changed at any time and will affect the vector velocity of the CNC motion immediately (if CNCAJerk = 0 to 9) or from the next motion segment otherwise. See detailed description of this keyword earlier within this document.

- CNCAPause:

  The user can pause the CNC motion at any time, and resume the CNC motion at any time, using the CNCAPause keyword. See detailed description of this keyword earlier within this document.

- CNCAEmrgDec:

  The CNC emergency deceleration (to be used when hitting hardware or software position limits) is defined externally of the CNC FIFO (meaning: it is a global definition and not a definition per segment). The user can modify its value at any time, including during the CNC motion. Yet, changing this parameter will have no affect on normal motion, unless one of the hardware or software limits are reached.

Checking the motion status

During motion the position, velocity, current and any other motion related parameters (whether of a specific member axis or the CNC vector motion itself) can be queried or recorded.

Most commonly used status parameters are:

- CNC motion: CNCAPosRef, dCNCAPosRef, CNCAStatus[9], CNCAStatus[10].
- Each member axis: PosRef, dPosRef, CurrRef, MotorCurr and PosErr.

If the motion ends due to an error or fault, the following status parameters will provide the required information about the reason for ending the motion (per each member axis):

- MotorOn.
- ConFlt.
- MotionStat.
- MotionReason.

Faster methods to push segment and inquire statuses

The controller provides faster method to push a new segment into the CNC FIFO and faster methods to inquire multiple statuses of the controller and to send messages to the controller. These methods are briefly described within this chapter.

The detailed description of these methods is not within the scope of this document.

For a CNC application, where the communication throughput of filling the CNC FIFO is a major performance criterion, we recommend using these methods (some of them supported only over Ethernet communication connection).

CNCAPushSeg keyword

This keyword is supported only over Ethernet communication connection to the controller. It will return an error if used over any other communication connection (such as RS232, CAN, etc.).

Using this keyword, the host/user can push a complete segment (type and parameters) using a single Ethernet message.

This replaces a sequence of messages consisting me CNCAPushType followed by multiple CNCAPushParam as required for the segment type.

As a result, it significantly improves the throughput of pushing segments into the CNC FIFO.

CNCAPushSeg keyword

This keyword is supported only over Ethernet communication connection to the controller. It will return an error if used over any other communication connection (such as RS232, CAN, etc.).

Using this keyword, the host/user can push a complete segment (type and parameters) using a single Ethernet message.

This replaces a sequence of messages consisting me CNCAPushType followed by multiple CNCAPushParam as required for the segment type.

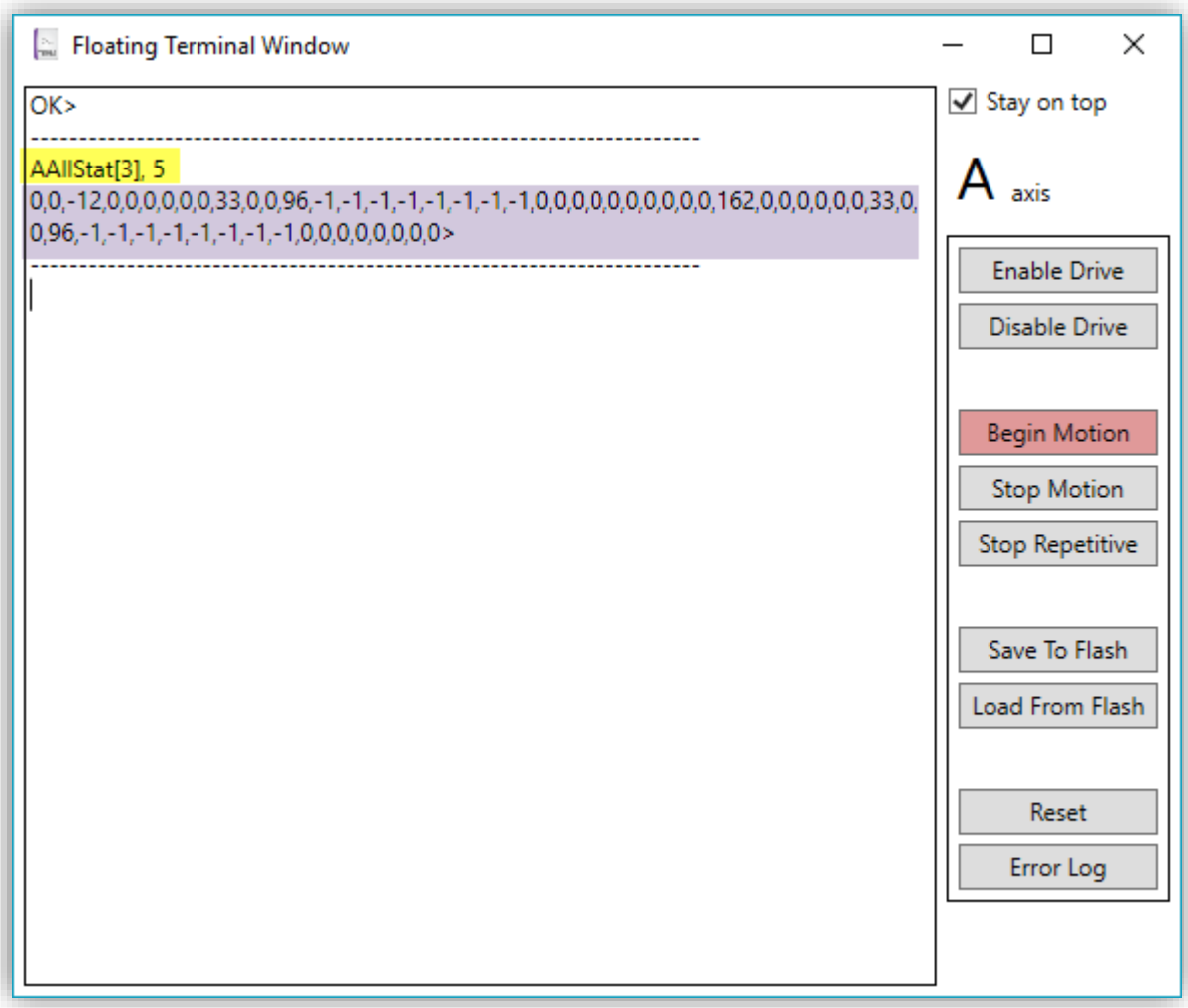
As a result, it significantly improves the throughput of pushing segments into the CNC FIFO.

AllStat keyword

Using this non-axis-related keyword, the user can ask the controller to report a list of statuses (as selected by the user) for a group o axes (again, as selected by the user) as a respond to a single AllStat inquiry.

This keyword is supported over all communication connections. However, it is extremely efficient over the Ethernet connection, where a bulk of statuses can be reported within a single (or few, as needed) Ethernet packets.

A typical usage of AllStat is as follows (over RS232, for example):



The message sent to the controller is marked in yellow:

AAllStat[3], 5

The "A" axis indication has no effect since the AllStat keyword is non-axis-related keyword.

"AllStat" is of course the keyword itself.

[3] ("array index") is a bitwise definition of the axes which are inquired. In this case, the value of 3 means axes A and B (right most bit – bit 0) is A axis, bit 1 is B axis and so on).

The value of 5 is used to select which group of statuses to report, again, using bitwise coding. Bit 0 (right most) refers to Group0 of statuses, bit 1 refers to Group1 and so on. Accordingly, a value of 5 refers to Group0 and Group2.

The groups of statuses (which statuses to report) are hardcoded within the controller, as described below.

Group0 consists of 13 statuses, while Group2 consists of 16 statuses, to a total of 29 statuses per axis.

Since statuses were requested for 2 axes (A and B, as reflected by "[3]"), the total number of statuses is 2x29 = 58 statuses. This is exactly the number of statuses at the controller reply (marked above in purple).

The reply first lists the statuses of the first axis (in this case, axis A), from the lowest group number to the highest group number (here: first group 0 and then group 2) and then, similarly to the next axes (here it is only axis B statuses, following the statuses of axis A).

Using the AllStat keyword, the user can inquire many statuses with a single message to the controller. On the other hand, the user has no control on which statuses are exactly provided (user can select which group of statuses, but not the content of each group), and thus the reply may contain statuses that are not really required.

Over Ethernet communication, where the reply is encapsulated into a single (or few if required) Ethernet packet, this method of inquiring the controller statuses is extremely efficient.

Bulk messages over Ethernet

Over Ethernet communication connection only, the user can encapsulate multiple messages into a single Ethernet packet.

The controller, when receiving this Ethernet packet, will handle the messages one-by-one and will reply with an Ethernet packet(s) that will include a list of replies, one per each incoming message.

Using this method, a faster communication can be established, as the overheads of multiple communication messages are significantly reduced (we have only one message instead of a list of messages).

Future features

The following features are expected in future firmware versions:

- Relative CNC motion:

  Currently, a CNC motion is defined using absolute target position only, including the start point of the motion which must be specifically defined as part of the CNC motion definition in the CNC FIFO.

  New firmware version will support relative definitions of motions, where the starting positions (of the member axes) will nto have to be defined in advance and each segment can be defined using absolute or relative target positions.

- Initial offset:

  New firmware versions will support definitions of offsets for all member axes, so that a CNC motion that was defined for a given machine can be adjusted to another machine, which has, for example, slightly different homing position, by changing only the offset values and keeping all CNC motion definition the same.

- Helix motion segment:

  Today the CNC supports Linear motion segments and ARC motion segments. New firmware versions will support also the Helix motion segment (two axes perform an ARC while a third axis performs a linear motion).

- More motion status bits at CNCAStatus[10]:

  New firmware versions will support additional status bits (in acceleration, in deceleration,) within the CNC motion status (CNCAStatus[10]).

- CNCAEncRatio:

  New firmware ratio will support compensation for different physical resolution of the member axes.

Converting G-code CNC trajectory

The PC Suite (see details within the PC Suite chapter below) includes a software tool that can convert G-code CNC files into our CNC trajectory definitions and even create a user program to create the CNC motion from the G-code file.

Known issues

- If, during a CNC motion, the controller will reach a segment (from CNCA FIFO) that will have a definition of involved axis that is not in the list of the member axes as marked during the Begin (see above), the controller will not consider it as an error and this involved axis will possibly move. This will be fixed in future firmware versions.

  For now, it is the user responsibility to define segments with involved axes that are always subgroup of the member axes.

Using the PC Suite CNC tool window

This chapter describes how to use the PC Suite Motion/CNC tool window to easily create CNC motions.

To be completed ???

Note:

The PC Suite Motion/CNC tool window supports the following future features:

- *Relative/Absolute mode.*
- *Set Coordinates Offset.*
- *Helix motion segment.*

Please avoid using these features since the controller's firmware does not include, yet, the support of these features.

Appendix A - About Automatic Corner Motion Segments

Note:

A corner segment is generally placed between two consecutive Linear Motion segments. However, as explained in the chapter "Keeping continuous motion" within this document, there are some types of segments ("no-motion-no-blocking") which can be placed in between motion segments without affecting the motion calculations (like type 8: Write to Discrete Outputs). These types of segments are "ignored" from the point of view of the motion trajectory calculations. For the simplicity of the explanation, this appendix assumes that the Linear Motion segment, following by the Automatic Corner Motion Segment and followed by the second Linear Motion segment, are all pushed one after the other without any other ("no-motion-no-blocking") segment pushed between them. However, while the explanation here is made simpler, the controller does support pushing these types of segments in-between, as explained within the above-mentioned chapter.

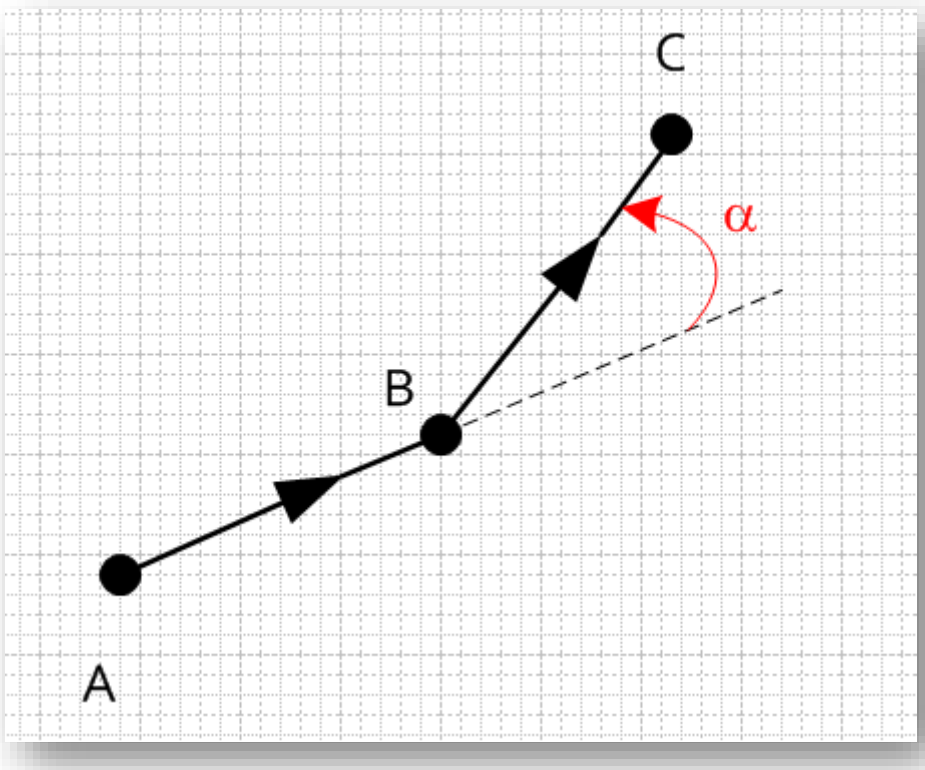
The controller automatically calculates a corner upon and based on the following conditions:

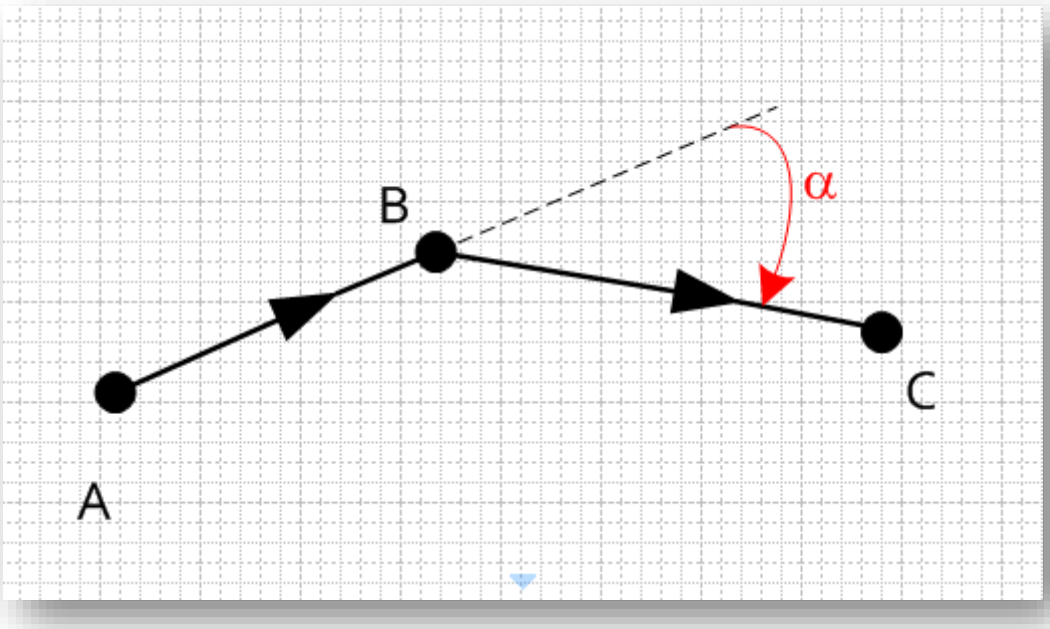
1. The user pushed an Automatic Corner Motion segment between two Linear Motion segments and if:
    a. All these 3 segments have exactly two identical involved axes.
    b. The first Linear Motion segment has end velocity not zero. If that velocity is zero, the controller does not calculate the corner and it remains empty (skipped during the CNC execution).
2. The Automatic Corner Motion segment, when pushed by the user into the FIFO is an empty segment which has no effect on the CNC trajectory.
3. Once the second Linear Motion segment is pushed (after the corner), the corner is being validated and if needed, calculated, and the results are placed in the corner segment parameters (and the segment is no more an "empty" segment).
4. The corner validation and calculations are done based on the two Linear Motion segments and the corner's parameters (as defined in the last Set Corner Parameters segment).
5. Under the following conditions the corner will not be inserted, and the Automatic Corner Motion segment will remain empty (meaning, will have no effect):
    a. The End Speed of the first (incoming) Linear Motion segment is 0.
    b. The angle between the two lines is smaller (or equal) to the MinimalAngleForCorner parameter.
    c. The angle between the two linear is greater than 160 degrees
    d. The lines (or one of them) are too short (**more details to be provided on future versions of this document**).

    What is the angle between two lines?

    Please refer to α in the following two figures (positive and negative cases, moving from A to B along a line and then from B to C along a line). In both cases, the angle α is defined to be positive, in the

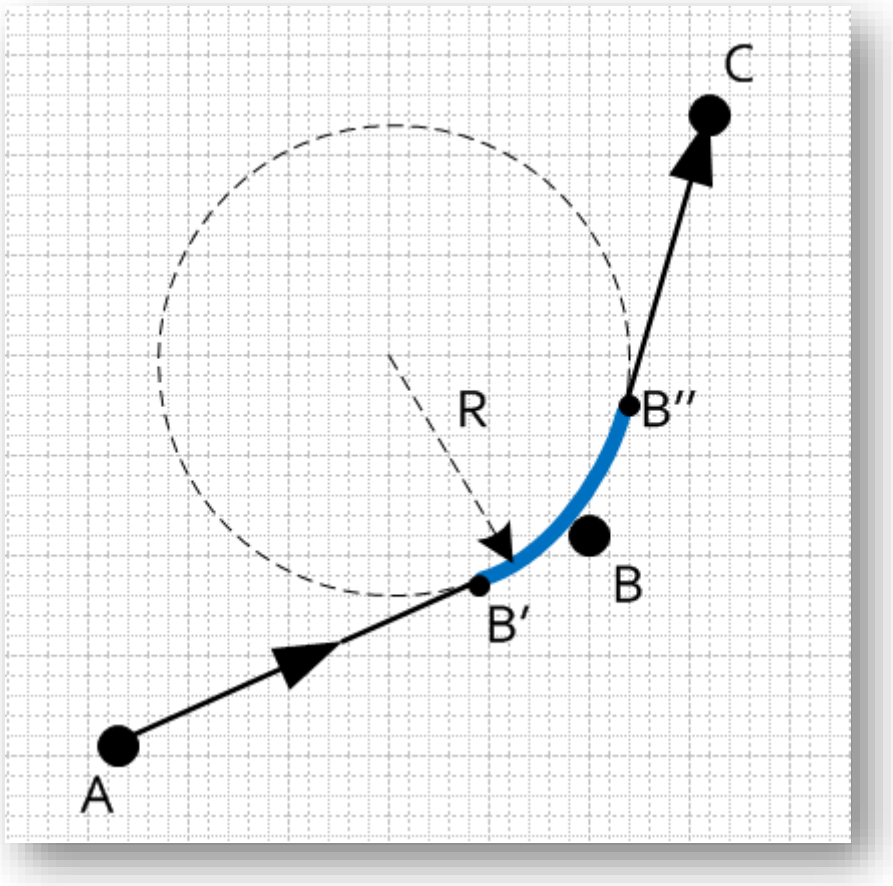
range of 0 to 180 degrees. It means that α of zero means continuing the same line while α of 180 degrees means returning along the same line.



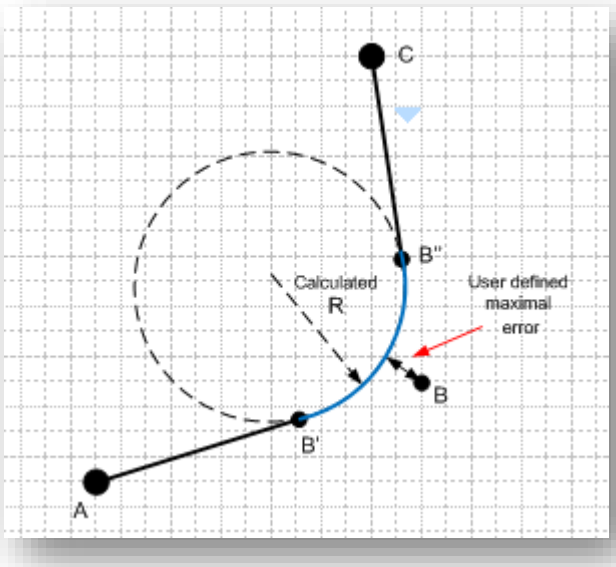


6. Two types of corners are supported: ARC (round corner) and Continuous Acceleration corner. The type of the corner is defined by the last pushed (executed) "Set Corner Parameters" segment.

7. ARC (round corner) is achieved by placing an ARC (part of a circle) between the two lines, with a radius based on the corner general parameters (see the Set Corner Parameters chapter) and properly shortening each of the lines, so the two lines will be tangent to the arc, as shown in the following picture:



The R (radius of the corner) is calculated based on the following parameters (set using the Set Corner Parameters segment): CornerRadiusMethod and CornerRadiusOrError. If CornerRadiusMethod is 0, then the corner radius is directly the value of CornerRadiusOrError. If CornerRadiusMethod is 1, the radius is calculated so that the maximal error imposed by the corner will be equal to: CornerRadiusOrError, as shown in the following picture:



Note that the resulted CNC trajectory does not pass through the original point B and that the original linear segment AB is shortened to be AB' and similarly BC is shortened to be B''C. All the required calculations are automatically performed internally by the controller.

Note that this kind of simple corner consists of sharp jump in the acceleration (infinite jerk) of each axis.

Future versions of this document will include sample graphs of the position, velocity and acceleration of each axis during such a corner, as well as the vector velocity and acceleration during the corner.

Once the controller calculates this type of corner, it actually replaces the empty Automatic Corner Motion segment with a suitable and properly calculated ARC segment (in addition to shortening the two linear segments as necessary).

8. Continuous Acceleration corner is based on a dedicated algorithm which satisfies the following conditions:

    a. The position references of both axes are continuous (same as ARC corner).
    b. The velocity references (1st derivative of the position references) are continuous (same as ARC corner).
    c. The acceleration references (2nd derivative of the position references) are continuous (this is different from the ARC corner).
    d. The vector velocity along the corner trajectory is constant.

The disadvantage of this type of corner is that when the corner is sharp (the angle α between the lines is high), the peak acceleration required from each axis is becoming very higher and higher.

Future versions of this document will include sample graphs of the position, velocity and acceleration of each axis during such a corner, as well as the vector velocity and acceleration during the corner.

Appendix B – Friction Compensation in CNC motions

Agito controllers supports friction compensation for most (scalar) motion modes.

The friction compensation is implemented at the beginning of the motion, to overcome (fully or partially) the static friction and to enable faster start of the motion and reduced position error at the beginning of the motion.

For CNC motion, this definition is extended, to cover all cases where a given axis starts to move after it was in zero speed.

The exact definition is:

If a given axis is participating in a CNC motion segment (any type) and
If this axis is to be moved in this segment and
If it was not moving at the previous motion segment (didn't participate or reached zero velocity)

The Friction Compensation will be applied on this axis as part of starting this motion segment.

Appendix C – Gain Scheduling for CNC motions

Agito controllers supports gain scheduling in various modes, all as a function of the single axis state.

For CNC motions, it may be required to support a gain scheduling mode in which the gains switching is performed as a function of the CNC motion segment type.

A new mode is added to the Gain Scheduling mode list (defined by the ScheduleMode parameter):

For CNC motions

In this mode:

- The 1st control set is used when not in CNC motion and when in motion, but in segments that are non-motion segments (like: Delay).
- The 2nd control set is used during linear motion segments.
- The 3rd control set is used during other types of motion segments (targeting corners).
- The 4th control set is used for a given time after the non-linear motion segment (targeting the time after exiting the corner segment)

Using this gain scheduling mode, the user can temporarily increase the closed loop BW during the corner and a given time following the corner, to improve the position tracking during the corner.

Typically, the Gain Scheduling mode and its ScheduleTime parameter should be identically set for all involved axes. However, user can decide to make the scheduling only for specific axis/axes and with different parameters.