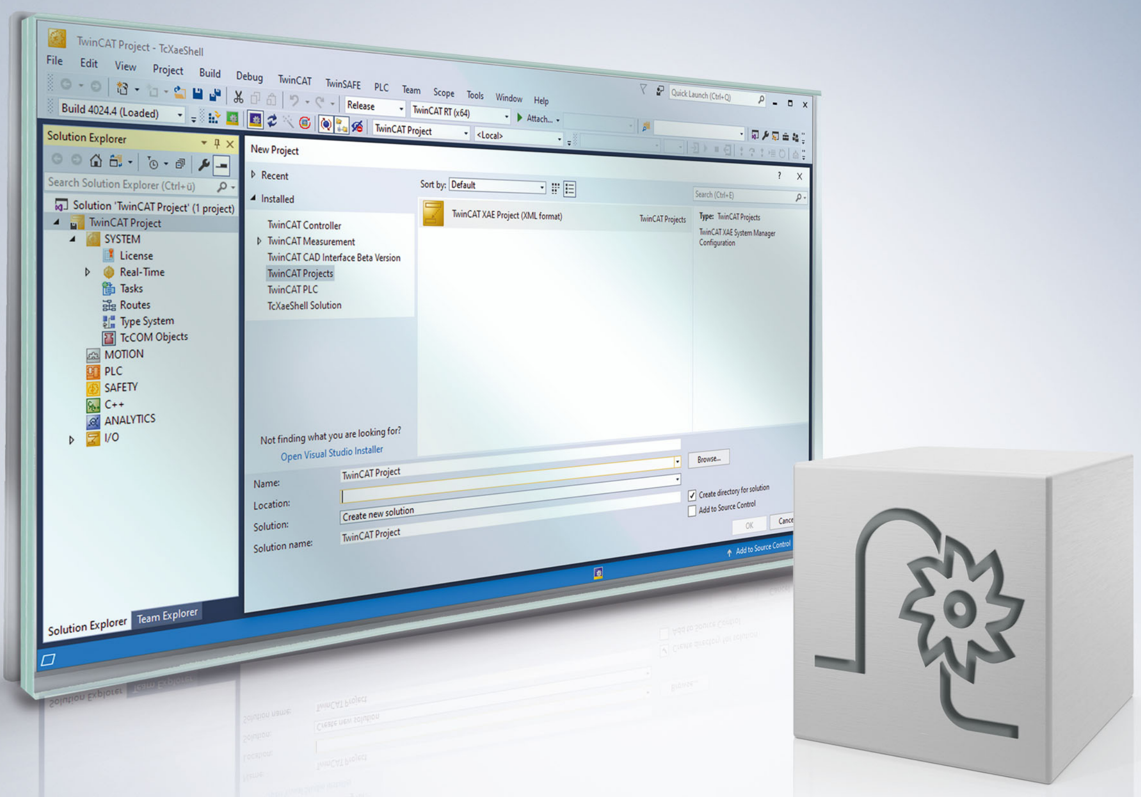
BECKHOFF New Automation Technology

Functional description | EN

TF5200 | TwinCAT 3 CNC

Job Manager



6/10/2025 | Version: 1.0

Table of contents

List of figures

1 Overview

Task

The Job Manager (JM) coordinates the start of NC programs in other channels. Batch mode is possible One or several channels and their CNC programs form the client (previously master) and the other channels are commanded channels (previously slave).

Besides the controlled forwarding of jobs, the Job Manager also manages feedback from agents. If several (subordinate) agents were commanded in a job, their feedbacks are summarised and sent to the client in one report.

The Job Manager uses specific statuses and basic functions of the PLCopen Organisation and extends them with functions for job identification and management.

This function is available as of CNC Build V3.1.3110.

Parameterisation

Channels are configured as clients or agents in the start-up list.

Programming

The Programming [▸ 18] section describes additional commands for commanding channels.

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

2 Description

2.1 General

Multi-channel CNC systems are often coordinated by complex PLC programs. The PLC programs:

- start parallelised or strictly sequential processes,
- command synchronous or asynchronous motions in CNC channels and
- recombine processes and motions.

In many cases, they are optimised for one application. “Batch mode” or operation with “recipes” are not possible, i.e. commands with deviating processes or rules for a program that the user can modify. The reasons for this are dependencies encoded in the process involving

- controller properties
- application programs
- program synchronisation
- the manufactured product itself.

When a Job Manager is used, it permits batch mode without PLC support. One or several CNC channels and their CNC programs assume the task of commander, also referred to as “client” (master). The other channels are service providers, referred to as “agents” (slaves). Commanding channels use additional commands for

- starting NC programs in agents
- blocking other agents from issuing commands and
- synchronising program flow in the commanding channel and its agents.

All channels not declared as clients display an error message when additional commands are called.

2.2 Job management

The primary task of the Job Manager is to manage jobs. A job is defined as a unit with the following attributes:

- CNC program to be started. It is identified by the CNC program filename.
- Start mode of the CNC program,
- optional parameters/values for transfer to the commanded program,
- a “Job D” and “Source ID” defined by the user.

When the Job Manager is enabled, each channel has a job queue containing jobs waiting to be started. When the channel is free – i.e. no program is running in the channel decoder – the next job in the queue is executed without user confirmation. The start sequence corresponds to the sequence in which the jobs were stored. The program described in the job is loaded and started in the same way as the previously known program start (see MCM automatic state indicator); in addition, optional start modes and parameters are transferred to the program. The job ID helps to keep the job identifiable and traceable for the Job Manager and the user throughout the entire runtime.

2.2.1 Start of job

There are three options to store a job in the job queue of a channel. If the options are used in parallel, they are processed from by the interfaces described below in descending order of priority. If the job queue is full, the client must wait. Implicitly, subordinate clients and their jobs also wait. The figure below shows start variants a) to c):

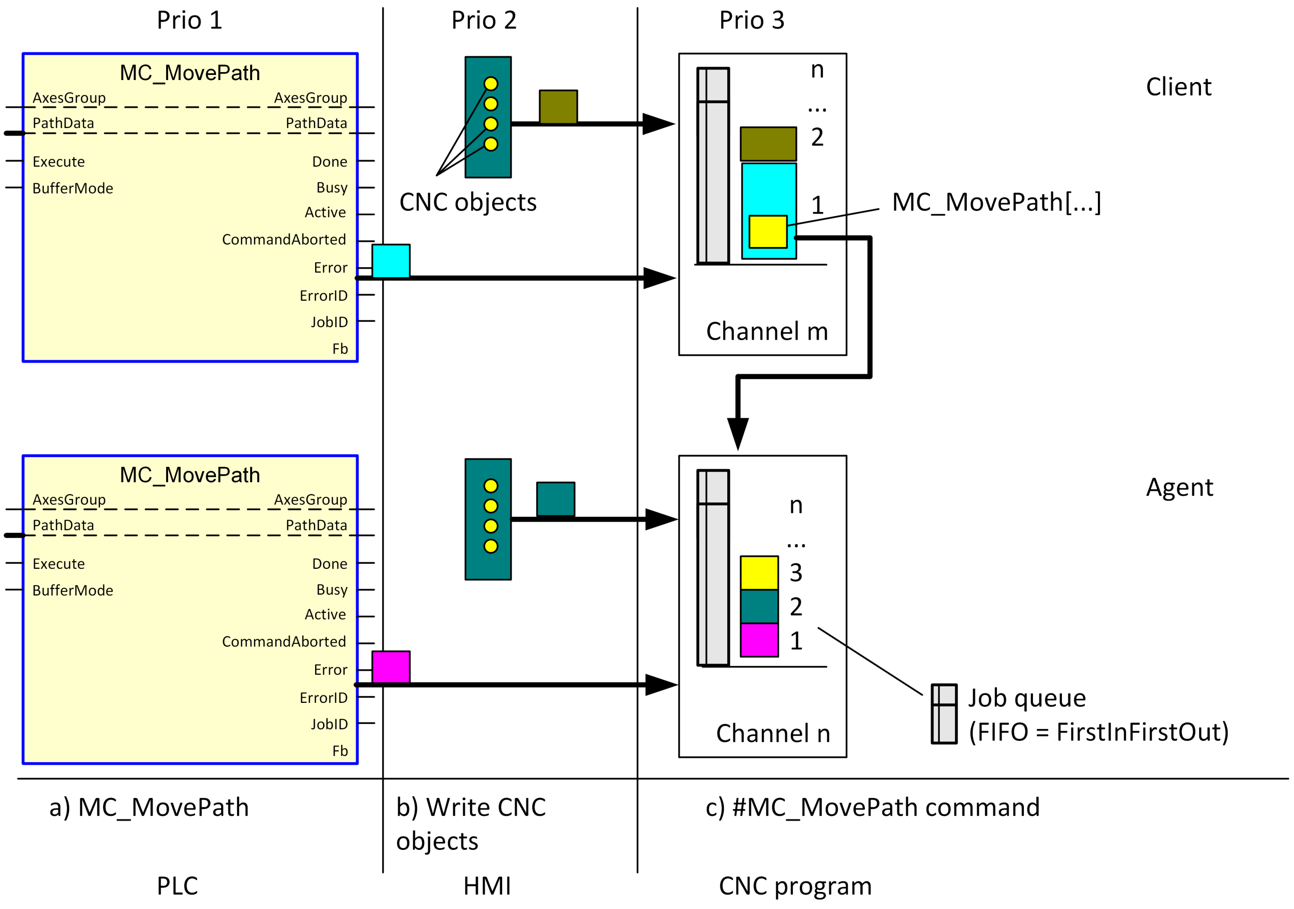


Fig. 1: Variants to start a job

- Variant a) Commanding via (PLC) PLCopen Part 4 block MC_MovePath [▸ 27]. The block can be used for both commanding and commanded channels.
- Variant b) Describe several CNC objects in specified order. The MC_MovePath function described in a) is triggered. Typically, this access is used by an HMI. The interface can be used for commanding and commanded channels. (see CNC objects [▸ 25])
- Variant C) A CNC program running in the commanding channel starts a job in another commanded channel using the #MC_MovePath command. (see Commanding agents [▸ 18])

Variant c) requires at least one commanding channel and one commanded channel.
Variant a) can only be commanded by the PLC.
Variants a) and b) can basically be used in any CNC channel irrespective of the Job Manager mode.

Starting an NC program is still possible using the previously known HLI MCM automatic state indicator despite the fact that the Job Manager is in use.

Regardless of whether it is started in the agent or client, only a single program is executed without parameters and without job identification. A program start commanded by the MCM automatic state indicator is not stored in the job queue. It does not recognise any job statuses, only known and documented (channel) statuses.

NOTICE

When both command types are used, the user is responsible for avoiding deadlocks or collisions with Job Manager jobs.

2.2.2 Job status

The status of each job is described by statuses that are defined in PLCopen and can assume the following values during the entire life span of a job:

- “Busy“ (MC_BUSY): The job is planned but is in the queue.
- “Started“(MC_STARTED): The job was started in the decoder.
- “Aborted“ (MC_ABORTED): The job was aborted. This was preceded by a stop or reset by the user.
- “Error“ (MC_ERROR): An error was detected during a job processing phase. The error may have occurred in the commanding channel itself or in a commanded channel started by the client.
- “Done“ (MC_DONE): Error-free end of the job and its commanded channels started by the client. When this status is signalled, the job is removed from the Job Manager system and deleted.

PLCopen uses proprietary output interfaces with the value range True/False to display statuses The implementation described here partially deviates from this and uses an enumeration or a numerical value for statuses. The name uses the PLCopen notation where “MC_” (for Motion Control) is placed in front for the sake of uniqueness.

2.2.3 Job ID and job tracking

The PLCopen specification does not specify how the completed message - i.e. the job status - of several jobs commanded one after the other by MC_MovePath can be identified at the "Done", "Busy" or "Active" pins. It only stipulates that the completed message at the “Done” pin of the MC_MovePath may only become active when no more jobs are active – i.e. when the system is “empty”.

The “Done” status is never active when the machine is at maximum utilisation and the commanding channel is in “continuous mode”. The user can assign a job ID (number) to still receive feedback for each of the commanded jobs and each of the status changes for these jobs.
Three blocks are used to command and request jobs:

- MCV_GrpPathMode [▶ 29]
- MCV_GrpPathPrepare [▶ 30]
- MCV_GrpReadJobAck [▶ 32]

MCV_GrpPathMode block

This block is optional and acts on the “mode” (operation mode) of the job to be started (=CNC program) (see also Channel operation mode).

MCV_GrpPathPrepare block

The MCV_GrpPathPrepare module adopts this mode and assigns additional job attributes that can be defined by the user, e.g. the job ID which can be specified by the JobID pin. All attributes are transferred via the “PathData” pin to the MC_MovePath block at job start.

MCV_GrpReadJobAck block

The MCV_GrpReadJobAck block is supplied by the “Fb” pin of the MC_MovePath block. On the rising edge, the block signals on the "Valid" output that a new state is displayed. Each job status change to MC_BUSY, MC_STARTED, MC_ABORTED, MC_DONE and MC_ERROR is displayed for each job started by MC_MovePath,. At the same time, each job ID assigned by the user at job start is displayed.

The figure below shows an example of a job with job ID "1" that was originally started by MCV_GrpPathPrepare and has just been completed.
The MCV_GrpReadJobAck module last showed the MC_DONE status of this job with job ID "1".
Previously, this function block first signalled the MC_BUSY status directly after commanding in MC_MovePath, and then the MC_STARTED status after starting in the decoder.
After starting the job with job ID "2", the MC_BUSY status for job ID "2" is readable in MCV_GrpReadJobAck. The statuses MC_STARTED, MC_DONE or, in the event of an error, MC_ABORTED or MC_ERROR, will follow later.

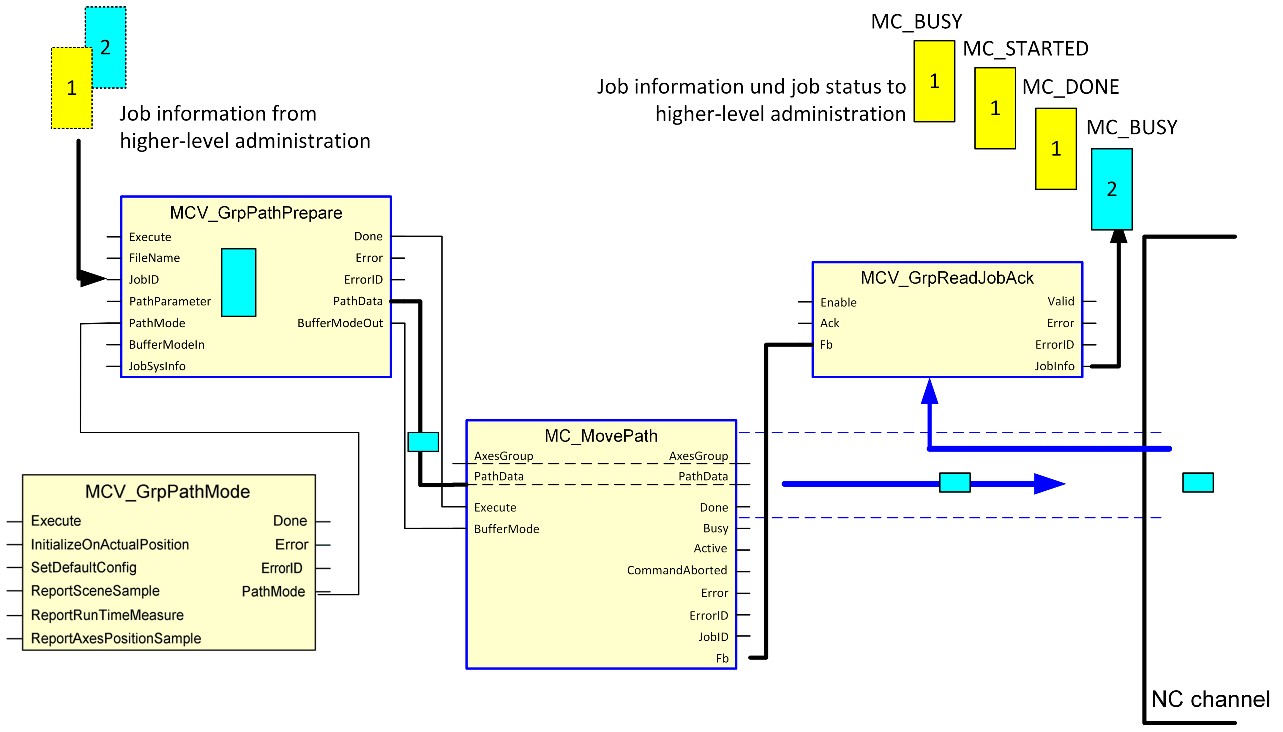


Fig. 2: Passing through the job ID from job start through to the display of a job status change

If MC_MovePath is executed in conjunction with MCV_GrpPathPrepare, MCV_GrpReadJobAck must be called permanently to read out the job statuses.

If this fails to happen, the internal buffers run full and block new commanded jobs.

As an alternative to the PLC blocks MCV_GrpPathPrepare, MC_MovePath and MCV_GrpReadJobAck, there are CNC objects that allow jobs to be enabled and status messages to be read. See CNC objects [▶ 25]

If the MC_MovePath function is triggered by CNC objects, acknowledgements must be read back by the corresponding CNC objects of the MCV_GrpReadJobAck.

If this fails to happen, the internal buffers of the Job Manager run full and block new commanded jobs.

If MC_MovePath is started from a CNC program, the status of the commanded job can be monitored as in Synchronisation with agent jobs [▶ 15]. Its job ID is required when called.

2.2.4 Aborting a job

The following PLC function blocks are intended for the controlled abort of one or several jobs.

- MCV_GrpResetForced
- MC_GrpStop

If one of the function blocks is used on a commanding channel, all waiting and active jobs in the channel are terminated. When they are aborted, all jobs not previously completed in commanded - i.e. subordinate - channels are also aborted. The status of each aborted job (not the channel!) changes to the MC_ABORTED status.

The selective abort of single client jobs is currently not supported.

Reason for this restriction:

In general, #SIGNAL / #WAIT events between channels cannot be reset in a coordinated process. The reason for this are the combinations of the interdependent statuses of all the channels involved and this can become very large. In addition, if an abort occurs, it is usually not possible to determine the actual system status.
If there is an incorrect restart, this could result in incorrect program starts or incorrect program flows.

If the function blocks for an abort are applied to an commanded channel, all waiting and active jobs are terminated there. If one of the aborted jobs was commanded by a commanding channel, the commanding job in the commanding channel is also set to the MC_ABORTED status and blocked. All jobs previously commanded by the commanding job for other commanded channels are not aborted.

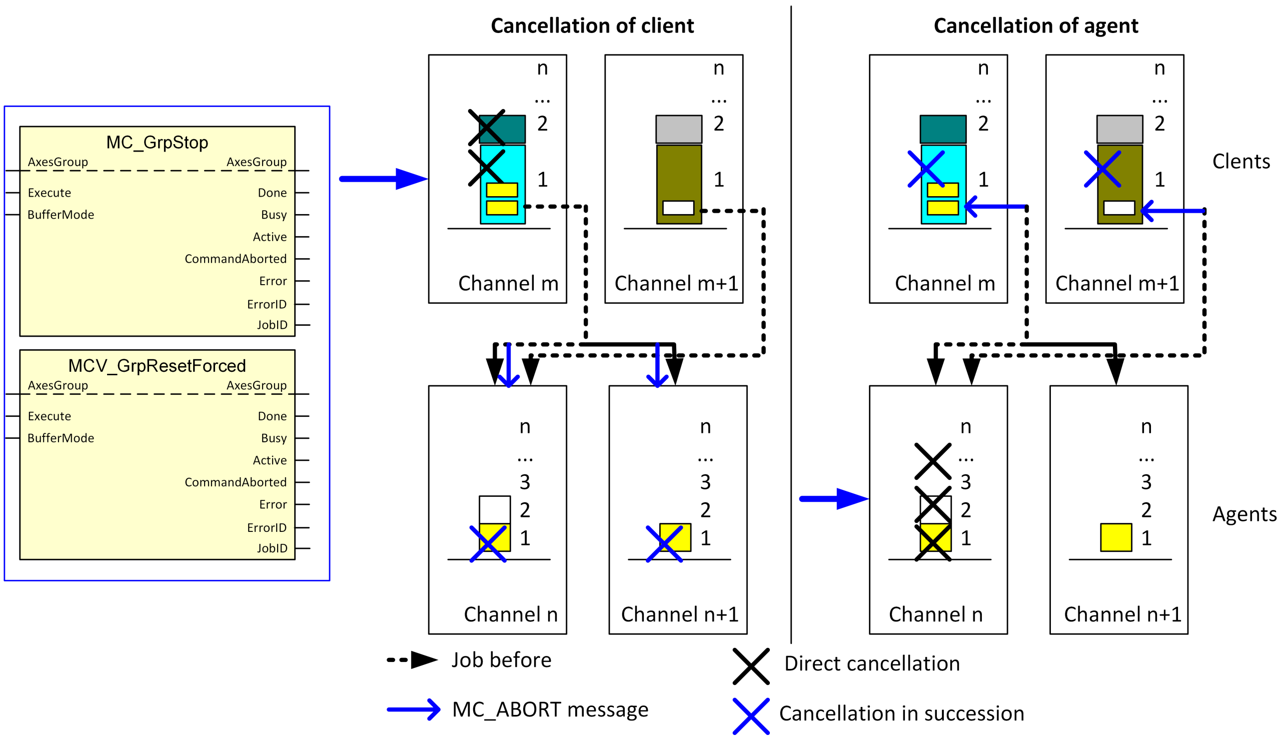


Fig. 3: Effect of job aborts on client and agent

The difference between **MCV_GrpResetForced** and **MC_GrpStop** is that MCV_GrpResetForced also resets all the axes of the commanding and commanded channels to their initial statuses.

An CNC object exists for MCV_GrpResetForced as well as for MC_MovePath and MCV_GrpReadJobAck.

#MCV_GroupResetForced can also be started from a commanding CNC program. As with MC_MovePath, the user must assign a job ID in order to permit monitoring via the described synchronisation function [▶ 15]. The abort behaviour is the same as the #MCV_GroupResetForced function triggered by the PLC on a commanded channel.

2.2.5 End of job

From the "classic” interpretation of a CNC program, M30 terminates the running program and "stops" the CNC channel.

When the Job Manager is used, M30 not only ends the CNC program in the *commanded channel* but also sends an MC_DONE to the commanding job or to the client, if available. The client then uses the message to manage its jobs internally. If there is another job in the client queue, it is automatically activated (message MC_ACTIVATED to the client) and the program described in the job is started.

On the other hand, if a program is terminated in the *commanding channel* with M30, the program identified by the next waiting job is started. However, the job belonging to the program containing this M30 is only terminated when all commanded agent jobs terminate by themselves. If this is the case, an MC_DONE is sent to the client, i.e. the PLC waiting for the MCV_GrpReadJobAck block. Since it can be assumed that commanded agent jobs have not yet been terminated when the next job is started, the result is that several jobs are active.

If programs with different agents are commanded consecutively in a client, it is possible that jobs that were started "later" in the sequence are completed "before" the jobs that were commanded previously.

The figure below shows the relationship between M30 and MC_DONE by means of a job "J1" and its sub-jobs "J1.1" and "J1.2".
If "J1.1" and "J1.2" are terminated with M30, they send an MC_DONE to the client. If the program of "J1" itself was terminated with M30 and all commanded jobs also have the MC_DONE status, an MC_DONE can be sent for "J1". All other jobs and their associated programs are not yet terminated since the corresponding MC_DONE is missing.

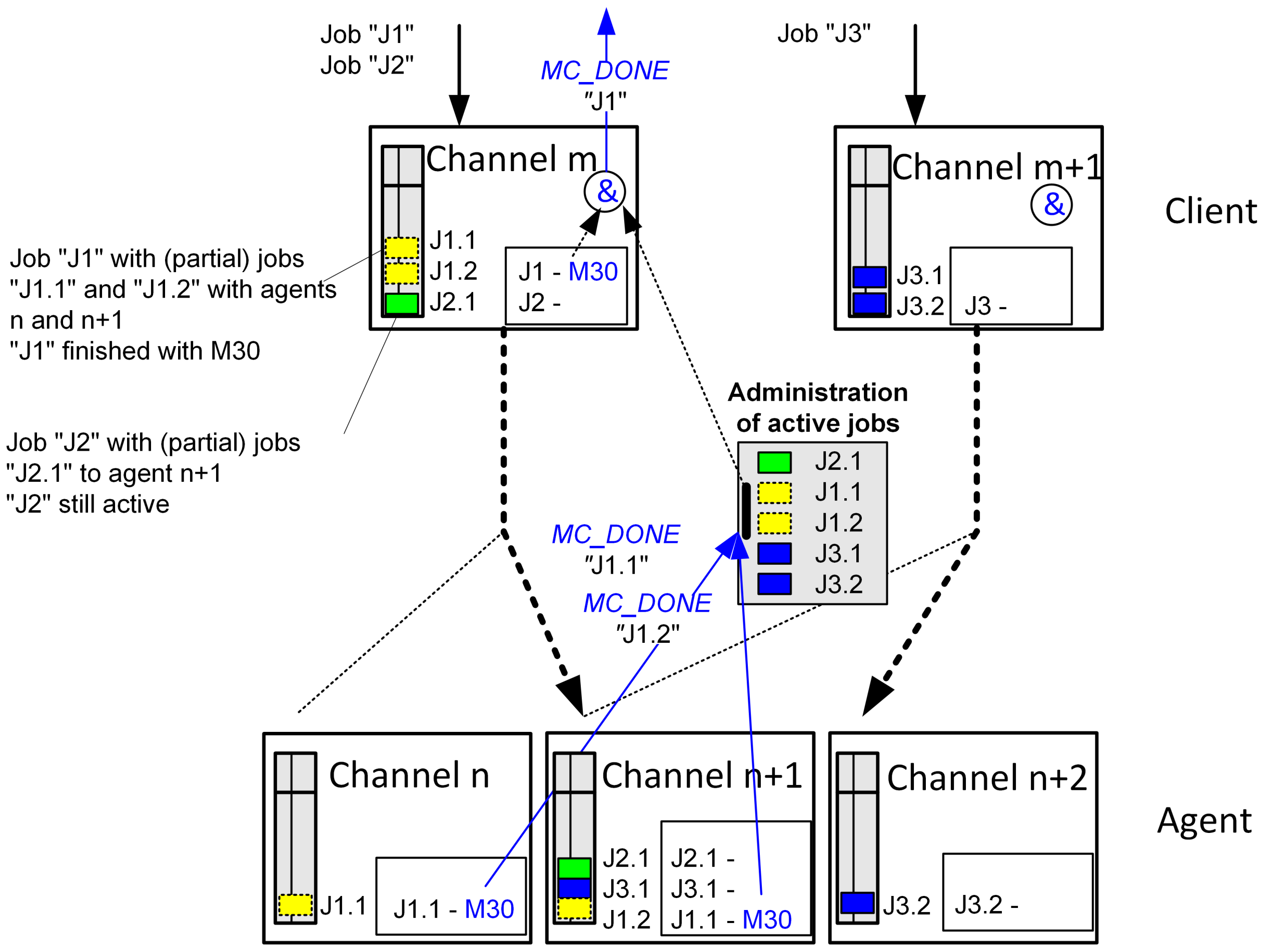


Fig. 4: End of job messages due to M30 relating to client and agent

2.2.6 Job stack

As described in section Job ID and job tracking [▶ 10], the job ID assigned by the user at job start is returned when the job is terminated. However, the job ID should also be visible in all agent jobs commanded by the client so that it can be used for testing or logging purposes.

The JM uses a “job stack” for this purpose: When a new job is created, it is generated and initialised with the user's job ID via an initial "PUSH". At each commanding level - including in the controller - the stack is extended by an additional "PUSH" of the internal job ID. After a job is terminated, the commanded user sends the current job stack back to its client - together with the status MC_DONE, MC_ABORTED or MC_ERROR. The receiver of the message accesses its job ID via a "POP" and updates its job management as required.

The Job Manager supports multi-client operation. This means that agents can be commanded by several clients. Since the job stack and the job status must always be returned to the correct client, an additional "source ID" is entered in each entry in the job stack in addition to the job ID. The source ID is used in the controller to determine the routes used by the job statuses (acknowledgements). The source ID is displayed in the MCV_GrpReadJobAck block.

The precondition for the “source ID” readable by the user in MCV_GrpReadJobAck is the assignment of a value in the MCV_GrpPathPrepare block. In addition to two job IDs that can be issued by the user, a separate source ID (number) can be assigned here. MCV_GrpReadJobAck returns all three entries at every status change. Both the source ID and the two job IDs, as well as the job IDs and source ID tuples assigned internally in the controller, can be tracked in the job stack at all levels of job processing via status displays or in the trace.

The user can use the job ID to identify workpieces in their user’s production processes. The source ID helps the user to return acknowledgements directly back to alternative commanding clients.

2.2.7 Prioritisation and lock blocks

If there are several commanding channels that command one or more jointly used commanded channels, their jobs - or more precisely their programs - compete to send jobs to the jointly used agent. The Job Manager determines the order of commands.

1. The job that has waited the longest for assignment is prioritised. An internal timestamp is used for this and is reassigned each time a new command is delivered for the first time. If they occur simultaneously, preference is given to the channel with the lowest configuration number ‘j’ in the parameter master[j].log_id of P-STUP-00206.
2. If the selected client cannot execute its command despite it being assigned (e.g. a job is placed in the agent queue - because the queue is full, for example), the client must wait. All other clients also wait. No reprioritisation takes place.
3. The wait can only be interrupted by an abort (MCV_GrpResetForced, MC_GrpStop), which is executed by preference.

In extreme cases without additional intervention, several active clients - i.e. several simultaneously running CNC programs - can result in the chaotic processing of competing jobs.

If this is not permitted and a commanding job requires exclusive access to the commanding sequence in "its batch", it can create a "lock block". The NC commands for this are:

- #LOCK and
- #UNLOCK

When the #LOCK command is executed, a client permanently blocks all other clients from sending their commands until #UNLOCK - or until the end of the client’s program with M30. The prioritisation described above is disabled for this time. This allows predefined job sequences in the queues of commanded channels and jobs in cooperating channels where their jobs/programs can be stored virtually "simultaneously" and deadlock-free in the agents.

The lock block only affects specific Job Manager commands. All other CNC commands continue to be executed regardless of this.

If no lock block is active, a newly commanded #LOCK command competes with all simultaneous commands in other commanding channels. The prioritisation rule described above applies.

A reset/stop command from any client is always enforced regardless of an active lock block. An active lock block is then deactivated.

2.2.8 Synchronisation with agent jobs

A client job can synchronise with one or more of its commanded jobs. The condition is that the "Job ID" assigned by the user at start-up via #MC_MovePath is present.

One or several events are waited for with

#WAIT MC_Status [JobID=.. , …]

and the known job ID. Events for #WAIT MC_Status are one or more of the four statuses MC_STARTED, MC_ABORTED, MC_ERROR, MC_DONE. If none of the parameters are specified when the call is made, this corresponds to the implicit specification of MC_DONE and MC_ABORTED.

Job IDs are stored in a history in order to return a correct status in a “later” request, even after the job is terminated/aborted. If the job ID specified is unknown because it was never commanded or is no longer in the history, an error is output.

The job ID history is deleted at program end.

It is therefore no longer possible to synchronise with jobs from previous (client) jobs. It is also not possible to synchronise with jobs from another client.

2.2.9 Job Manager groups

Currently, a maximum of two independent Job Managers can be configured with their own client and agent channels, whereby one or more channels can be defined as the commanding or commanded channel. A channel can only belong to one of the two Job Manager groups. As a result, a commanded channel of a Job Manager group cannot be the client of another Job Manager group at the same time.

It is not possible to change the Agent or Client property or to switch from one Job Manager group to another during controller operation.

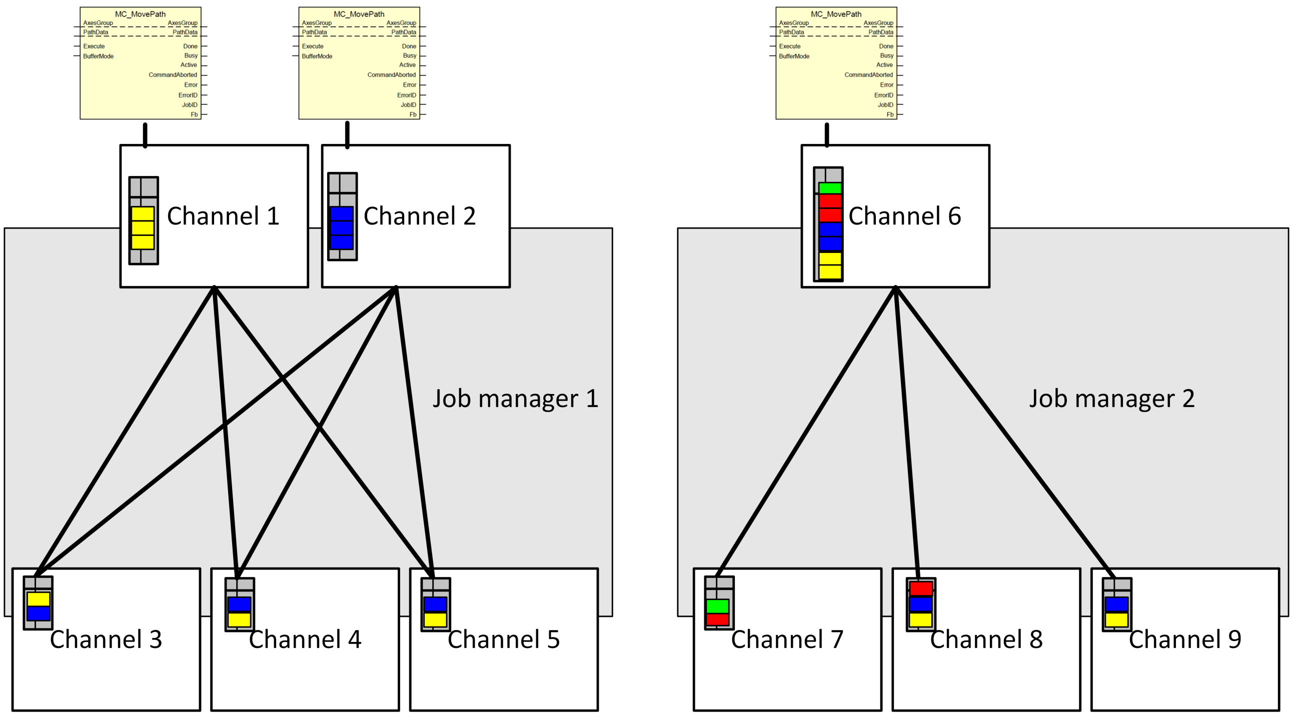


Fig. 5: Assigning channels to Job Manager groups

2.3 Application scenarios of the Job Manager

The examples below use the Job Manager in different application forms. Here, a commanding channel executes the "recipe" with its program while each commanded channel performs predefined services or functions. Each of the participating channels can be used simultaneously as a fully-fledged CNC channel.

2.3.1 Multi-station operation

A machine has two machining tables and a machine portal. The portal works on a workpiece or workpieces attached to one or both tables alternately or in parallel.
In this configuration, the "tables" are commanding channels, where each "table" commands its machining sequence as single jobs to the agent, i.e. the portal. The two tables can be set up separately from each other and released for production. Contiguous jobs are commanded exclusively for each table by a lock block.

For example, if the tool management system or tool changers are also used as independent channels, the commanding table can "simultaneously" command agents working in cooperation by means of a temporarily usable lock block. For example, tools can be provided "in advance" in the exchange position and the "in parallel commanded" CNC program of the machine portal has access to this later.
This can reduce auxiliary and set-up times.

The jobs commanded in the client channels correspond to a "pallet job".

2.3.2 Machining in several stations

A client job starts a sequence of work steps (=jobs) on a workpiece. The Job Manager distributes this sequence to separate machining units whose functions are mainly “autonomous”. An example of a rotary indexing table would be the sequence:

1. Transport_to_1
2. Machining_1
3. Transport_1-2
4. Machining_2
5. Transport_2-x
6. …

Since the jobs are stored in the job queues of the commanded channels, the client is free after sending its command. It can command further follow-up workpieces (even with a different machining sequence) until the job queue of an agent it requires reports "full” status. The agent then stops. When the station is free, the client sends its next command. Several workpieces (=jobs) are “simultaneously” machined in a machine.

The job commanded in the commanding channel corresponds to a “workpiece job”.

2.3.3 Die sinking

A special case of machining in multiple stations is “die sinking” (see FCT-C44]. Here, a commanding channel, i.e. the down channel, uses the Job Manager to supply two agent channels, i.e. the orbit channel and the escape channel, with coordinated NC programs.
A special feature of this application is that a resulting 3-dimensional movement is created by superimposing the 2-dimensional movements of the down channel and orbit channel. The escape channel also superimposes the movements of the down channel.

The job commanded in the commanding channel corresponds to a “workpiece job”.

3 Programming

3.1 Job Manager NC commands

The Job Manager commands and constants described below can only be executed in channels that are configured as clients.

If these commands are executed in channels that are configured as agents, error ID 280611 is output.

All commands are stopped until the job is successfully completed: In the case of #MC_MovePath and #MCV_GrpResetForced, this means placing the job in the queue; with synchronisation commands, it is when the expected status occurs.

All commands and parameter names are case-sensitive.

3.2 Commanding agents

A CNC program running in the commanding channel starts a job in another channel "agent" using the #MC_MovePath command.

Available as of V3.1.3110

```
Syntax of the NC command:
```

#MC_MovePath [SYN] **[CH=.. JobID=.. FileName=.. @PL<1…20>=..**
[InitializeOnActualPosition=..]
[SetDefaultConfig=..]
[ReportSceneSampling=..]
[ReportRunTimeMeasure=..]
[ReportAxesPositionSample=.]. **]**

SYN	Synchronous command execution of the ISO program. Before the command is executed, an implicit channel synchronisation takes place (implicit #FLUSH WAIT).
CH=..	Logical channel number of the channel in which the job is executed. The link between the CNC channel number and the CNC channels is specified in the start-up list. (See Parameter [▶ 36]).
JobID=..	User-specific job identification number (job ID). Every number must be unique in the commanding program of the master channel. For example, this ID is used in the #WAIT MC_Status command for job identification.
FileName=<*filename*>	Filename of the ISO program which is to be started.
@P<i>=..	Parameter transfer to the commanded ISO program. These parameters can be accessed in the called "main program" by @PL<i>.
InitializeOnActualPosition=..	Requests current positions at program start. See Channel operation mode- SUPPRESS_POSITION_REQUEST If InitializeOnActualPosition is not used, the current configuration specified applies. ON — Request the position regardless of the current configuration.

	OFF	"No position request" regardless of the current configuration.
	USE_ACTUAL	The current configuration specified applies
SetDefaultConfig =..	Initialises decoder working data at program start.	
	See Channel operation mode- SUPPRESS_PROG_START_INIT	
	If SetDefaultConfig is not used, the current configuration specified applies.	
	ON	Initialisedecoder working data.
	OFF	Deselect initialisation of decoder working data.
	USE_ACTUAL	The current configuration specified applies
ReportSceneSample=..	Enables the interface to log scene data.	
	See Channel operation mode- BEARB_MODE_SCENE	
	If ReportSceneSample is not used, the current configuration specified applies.	
	ON	Log scene data.
	OFF	Disable scene data logging.
	USE_ACTUAL	The current configuration specified applies
ReportRunTimeMeasure=..	Enables the interface to log time stamps.	
	See Channel operation mode - ONLINE_PROD_TIME	
	If ReportRunTimeMeasure is not used, the current configuration specified applies.	
	ON	Generate time stamp data.
	OFF	Disable logging of time stamp data.
	USE_ACTUAL	The current configuration specified applies
ReportAxesPositionSample=..	Enables the interface to log axis positions.	
	See Channel operation mode- ON_LINE	
	If ReportAxesPositionSample is not used, the current configuration specified applies.	
	ON	Generate log data of axis positions.
	OFF	Disable logging of axis positions.
	USE_ACTUAL	The current configuration specified applies.

Starting an NC program in another channel

Start the CNC program JM-1-ch2.nc in a channel where the log_id is "1" without position request and without initialising the decoder. The transfer parameter @PL5 can be read out in the commanded channel; all other @Plx cannot. An error occurs at a read attempt.

```
%ExampleMC_MovePath1 Master
N010 #MC_MovePath [CH=1, JobID = 5, FileName = "JM-1-ch2.nc", \
     InitializeOnActualPosition = OFF, SetDefaultConfig = OFF \
     @PL2 = 1000.5 @PL5 = 50]
N020 M30
```

Parameters can be accessed in the JM-1-ch2.nc program.

```
%ExampleMC_MovePath1 Slave „JM-1-ch2.nc"
N010 V.L.Parameter1 = @PL5  ;V.L.Parameter1 = 50
N020 X@PL2                  ;Commanded X Position 1000.5
;…
```

After reading in #MC_MovePath, the interpretation of the CNC program is not continued until the job commanded by the command is successfully stored in the job queue of the channel agent.

3.3 Locking competing clients

Executing the #LOCK command gives a client exclusive access. Locking access for all other clients (of the same Job Manager group) is cancelled with #UNLOCK.
It is not permitted to use further #LOCK commands within a locked block or to use further #UNLOCK commands outside the lock block.

If a lock block is still active at the end of the main program (M30), it is implicitly terminated and a warning is output.

```
Syntax:
```

#LOCK	Enable exclusive access
#UNLOCK	Disable exclusive access

The lock block only affects specific Job Manager commands. All other CNC commands continue to be executed regardless of this.

Properties of the #LOCK command:

Every new incoming #LOCK receives an initial timestamp in the controller. This timestamp is used to prioritise several (possibly already waiting) #LOCK requests. The channel with the oldest #LOCK timestamp is granted access and the lock property. If another #LOCK is pending after an #UNLOCK, the timestamp prioritisation takes effect again and enables the next oldest #LOCK.
If they occur simultaneously, the channel with the lowest configuration number ‘j’ in the parameter *master[j].log_id* has priority (P-STUP-00206 [▶ 36]).

3.4 Waiting for the status of the commanded job

The #WAIT command is used to synchronise a commanding program with a job that the client previously commanded (requested). The commanding program is stopped until the expected status occurs. If several alternative states are specified, it is enough to fulfil one of the requested statuses for the program to continue running. The last valid return value of #WAIT MC_Status can be read out by the MCV_WAIT_STATUS function and used to continue program execution.

Available as of V3.1.3110

```
Syntax of the NC command:
```

#WAIT MC_Status [JobID=.. [MC_NEW] [MC_BUSY] [MC_ACTIVE] [MC_DONE]
[MC_ABORTED] [MC_ERROR]]

JobID=..	User-specific JobID/job identification number. Every number must be unique in the commanding program.
MC_NEW	New job received but not yet processed in the commanded channel.
MC_BUSY	New job waits for execution in the commanded channel.
MC_ACTIVE	The job is active in the commanded channel but was not completed.
MC_DONE	The job was successfully completed in the commanded channel.
MC_ABORTED	The job was aborted in the commanded channel before completion. An abort can be forced by a reset command, e.g. #MC_GroupResetForced.
MC_ERROR	An error occurred while the job is in progress.

Properties:

- Job IDs are stored in a history when #WAIT MC_Status is called in order to return a correct status in a “later” request even after the job is completed/cancelled. If the job ID specified in JobID is unknown because it was never commanded or is no longer in the history, an error is output.
- The job ID history is deleted at program end. It is therefore no longer possible to synchronise with jobs from previous (client) jobs. It is also not possible to synchronise with jobs from another client.

If no expected status is defined for #WAIT MC_Status, an implicit MC_DONE and MC_ABORTED apply.

Waiting for job state

The program starts a “SlaveFile”.nc program in logical channel 3 with job 633, then waits for the job to be completed with the alternative states MC_DONE, MC_ABORTED or MC_ERROR. If none of the states is reached, the program stops.

```
%Example MC_Wait
N010 #MC_MovePath SYN[ CH=3 JobID=633 FileName=”SlaveFile.nc”]
N020 #WAIT MC_Status [JobID=633 MC_DONE MC_ABORTED MC_ERROR]
N100 M30
```

3.5 Requesting the last valid status in #WAIT MC_Status

After one or more previous #WAIT MC_Status [▶ 21] commands, the last valid return value of #WAIT MC_Status can be read out by the MCV_WAIT_STATUS function and used to continue program execution. The numerical values stored in the constants apply as return values.

The following constants for status are available in the CNC code:

- MC_NEW
- MC_BUSY
- MC_ACTIVE
- MC_DONE
- MC_ERROR
- MC_ABORTED

They are used to compare a stored variable after a #MCV_WAIT_STATUS.

Available as of V3.1.3110.

```
Syntax:
```

#MCV_WAIT_STATUS [JobID=..]

JobID=..	User-specific JobID/job identification number. The number must be unique in the commanding program.

To call this function, it requires one or more #WAIT MC_Status commands with the same JobID=... for the number stored in <JobID>.

If this is not the case, an error is output.

Waiting for a job signal

The program starts a “SlaveFile”.nc program in logical channel 3 with job 633, then waits for the job to be completed with the alternative states MC_DONE, MC_ABORTED or MC_ERROR.The valid state when #WAIT MC_Status is enabled can be analysed.

```
%Example MC_Wait
N010 #MC_MovePath SYN[ CH=3 JobID=633 FileName=”SlaveFile.nc”]
N010 #WAIT MC_Status [JobID=633 MC_DONE MC_ABORTED MC_ERROR]
N020 V.P.McStatus = MCV_WAIT_STATUS [633]
N030 $IF V.P.McStatus != MC_DONE
N040   #ERROR [ID455 MID0 RC2 PV1=V.P.McStatus \
              PV2=MC_DONE PM1=3 PM2=633]
N050 $ENDIF
N100 M30
```

3.6 Channel reset in a commanded channel

The MCV_GrpResetForced command forces a channel reset in another channel.

Available as of V3.1.3110.

```
Syntax:
```

#MCV_GrpResetForced [SYN] [CH=.. JobID=..]

SYN	Synchronous command execution of the ISO program. Before the command is executed, an implicit channel synchronisation takes place (implicit #FLUSH WAIT). Without SYN the command is executed without channel synchronisation.
CH=..	Logical channel number P-STUP-00208 [▶ 36] of the channel in which the job is executed.

JobID=..	User-specific job identification number(JobID). Every number must be unique in the current commanding program of the master channel. For example, this JobID is used in the #WAIT MC_Status [▶ 21] command for job identification.

Executes a reset in another channel

The executed program resets channel 3.

```
%ExampleMCV_GrpResetForced
N010 N030 #MCV_GrpResetForced SYN [ CH=3 ID=634]
N020 M30
```

3.7 V.G. variables

The state variable **V.G.IP_NR** is assigned in conjunction with the Job Manager.
The variable is read only. The value of the variable is the logical number of the commanding channel.

V.G.IP_NR returns its own logical channel number to the NC program of a commanding channel, . The variable is defined in the configuration of the Job Manager in P-STUP-00206 [▶ 36].

If the variable is used in a commanded channel, it returns the logical channel number of the commanding channel if the job was started by a commanding job.
If the program was started directly in the commanded channel, the value of P-STUP-00208 [▶ 36] in the configuration is returned.

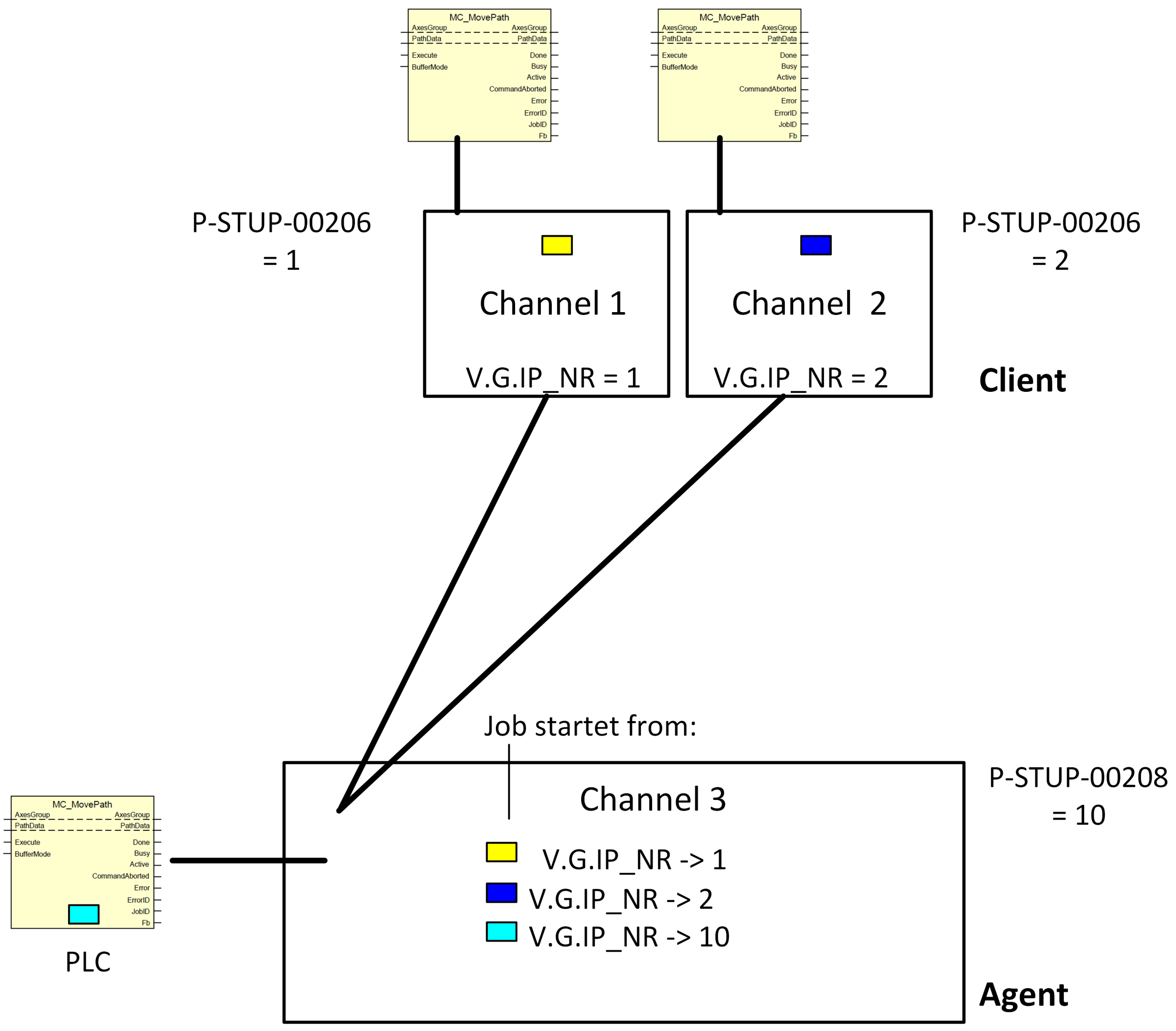


Fig. 6: Values of the V.G. variable V.G.IP_NR relating to the commanding and commanded channels

4 CNC objects

The CNC object described below are accessible via the COM task.

They represent interfaces of PLC blocks. The tables describe the initial values which can be overwritten if necessary and the corresponding input pins of the PLC blocks.

4.1 MC_MovePath + MCV_GrpPathPrepare + MCV_GrpPathMode

Object	Initial value	Corresponding element or constant in PLCopen
mc_plco_move_path_file_name_w	“”	MCV_GrpPathPrepare >FileName
mc_plco_move_path_fb_id_w	HMI	-
mc_plco_move_path_jstack_count_w	0	2, if (0) and (1) are described 1, if only (0) contains valid data 0, if no JobID data is valid
mc_plco_move_path_jstack_id_0_w	0	MCV_GrpPathPrepare >JobID.JobID (0)
mc_plco_move_path_jstack_index_0_w	0	MCV_GrpPathPrepare
mc_plco_move_path_jstack_id_1_w	0	MCV_GrpPathPrepare >JobID.JobID (0)
mc_plco_move_path_jstack_index_1_w	0	MCV_GrpPathPrepare >JobID.IfcID.i_index [Source-ID (1)]
mc_plco_move_path_parameter_w	“”	MCV_GrpPathPrepare >PathParmeter
mc_plco_move_path_coord_sys_w	0	-
mc_plco_move_path_buffer_mode_w	0	-
mc_plco_move_path_trans_mode_w	0	-
mc_plco_move_path_init_on_act_pos_w	USE_ACTUAL	MCV_GrpPathMode >InitializeOnActualPosition
mc_plco_move_path_set_def_config_w	USE_ACTUAL	MCV_GrpPathMode >SetDefaultConfig
mc_plco_move_path_rep_scene_samp_w	USE_ACTUAL	MCV_GrpPathMode >ReportSceneSample
mc_plco_move_path_rep_ax_pos_samp_w	USE_ACTUAL	MCV_GrpPathMode >ReportAxesPositionSample

Writing mc_plco_move_path_file_name_w triggers the command; the other objects can be previously written in any order.

4.2 MCV_GrpResetForce

Object	Initial value	Corresponding element or constant in PLCopen
mc_plco_reset_w	“”	1

Object	Initial value	Corresponding element or constant in PLCopen
mc_plco_reset_fb_id_w	HMI	-
mc_plco_reset_jstack_count_w	0	2, if (0) and (1) are described 1, if only (0) contains valid data 0, if no JobID data is valid
mc_plco_reset_jstack_id_0_w	0	JobID.JobID (0)
mc_plco_reset_jstack_index_0_w	0	JobID.IfcID.i_index [Source-ID (0)]
mc_plco_reset_jstack_id_1_w	0	JobID.JobID (1)
mc_plco_reset_jstack_index_1_w	0	JobID.IfcID.i_index [Source-ID (1)]

Writing mc_plco_reset_w triggers the command; the other objects can be previously written in any order.

4.3 MCV_GrpReadJobAck

Object	Corresponding element or constant in PLCopen
mc_plco_resp_job_state_r	0: no data read 1: data read and valid
mc_plco_resp_fb_id_r	Current value of HMI interface
mc_plco_resp_error_id_r	MCV_GrpReadJobAck>ErrorID
mc_plco_resp_jstack_count_r	2, if (0) and (1) are described 1, if only (0) is described 0, if no parameter is valid
mc_plco_resp_jstack_id_0_r	MCV_GrpReadJobAck>JobID.JobID (0)
mc_plco_resp_jstack_index_0_r	MCV_GrpReadJobAck>JobID.IfcID [Source-ID (0)]
mc_plco_resp_jstack_id_1_r	MCV_GrpReadJobAck>JobID.JobID (1)
mc_plco_resp_jstack_index_1_r	MCV_GrpReadJobAck>JobID.IfcID [Source-ID (1)]

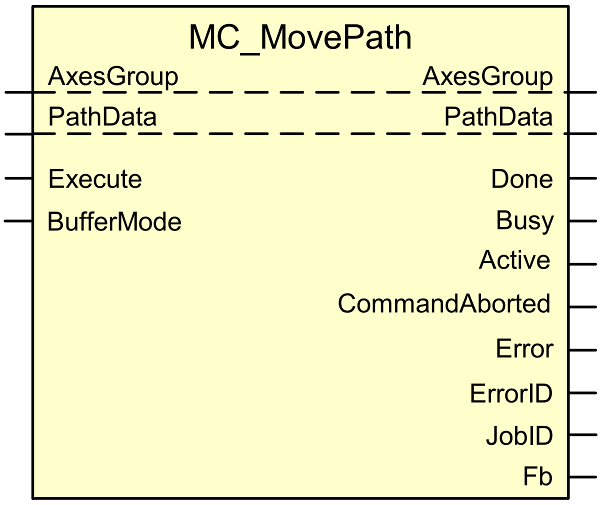
Reading mc_plco_resp_job_state_r triggers the read operation; then the other objects each have their value.

5 PLCopen function blocks

5.1 MC_MovePath

When implemented, this function block enables the execution of an NC program. The name of the file containing the NC program is specified in the data structure MC_PATH_DATA_REF.

Block diagram



FB parameters

VAR_IN_OUT		
Variable name	**Data type**	**Description**
AxesGroup	AXES_GROUP_REF	Axis group reference
PathData	MC_PATH_DATA_REF	Reference to the path description. This implementation means an NC program.
VAR_INPUT		
Variable name	**Data type**	**Description**
Execute	BOOL	Starts the command on the rising edge.

BufferMode	MC_BUFFER_MODE	The input defines when a job is activated provided that other jobs are already active when the FB is commanded or are waiting for execution. It also defines the path generated at the transition between 2 jobs. The following values are possible:
		mcAborting = 0
		mcBuffered = 1
		mcBlendingPrev = 3

VAR_OUTPUT		
Variable name	**Data type**	**Description**
Done	BOOL	NC program execution is completed.
Busy	BOOL	TRUE indicates that the function block is executing a job.
Active	BOOL	The command for the FB is not yet completed.
CommandAborted	BOOL	NC program execution was aborted.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code

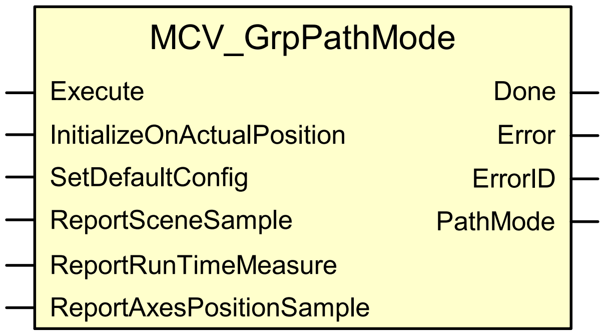
JobID	UDINT	Ordinal number of the last job sent by the FB.
Fb	MCV_FB_REF	Optional for use in applications with job management (MCV_GrpReadJobAck) [▸ 32] This output is available as of CNC Build V3.1.3103.1.

5.2 MCV_GrpPathMode

This function block is available as of CNC Build V3.1.3103.1.

The function block is used for entering options to execute a job to be started in the data structure MCV_PATH_MODE of the "PathMode" output. The output is linked to the “PathMode” input of an instance of the type MCV_GrpPathPrepare [▶ 30].

Block diagram



FB parameters

VAR_INPUT		
Variable name	**Data type**	**Description**
Execute	BOOL	On the rising edge at the input, the values of the input pins referring to start options are entered in a data structure of the type PATH_MODE and output at the PathMode output.
InitializeOnActualPosition	MCV_START_MODE	
SetDefaultConfig	MCV_START_MODE	
ReportSceneSample	MCV_START_MODE	
ReportRunTimeMeasure	MCV_START_MODE	
ReportAxesPositionSample	MCV_START_MODE	
VAR_OUTPUT		
Variable name	**Data type**	**Description**
Done	BOOL	TRUE indicates that the values of the start options at the PathMode output are available in a structure of the type MCV_PATH_MODE.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code
PathMode	MCV_PATH_MODE	Parameter for program mode of a job

5.3 MCV_GrpPathPrepare

This function block is available as of CNC Build V3.1.3103.1.

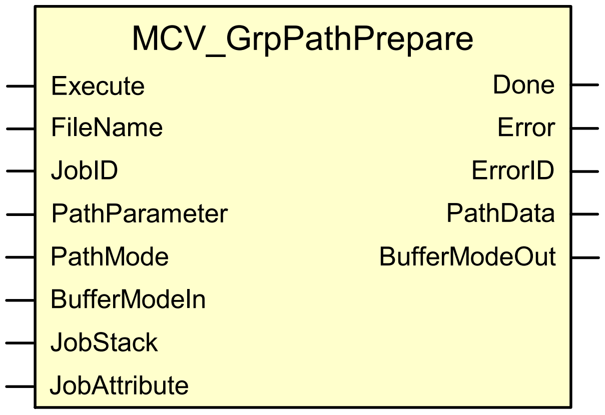
The function block is used to transfer job data, program parameters and program modes to a data structure of the MC_PATH_DATA_REF type. This structure is output at the PathData output and must be transferred to the corresponding input of an MC_MovePath [▶ 27] instance.

The function block inputs can be written by the user. Alternatively, the function block can be linked to the outputs of an MCV_GrpGetJobRequest instance to forward jobs generated by an HMI or another NC channel, for example.

The structure MC_PATH_DATA_REF may only be described by instances of the type MCV_GrpPathPrepare. This ensures that future changes to the library have no impacts on existing PLC programs.

What remains permitted is the direct entry of a name or path for an NC program on MC_PATH_DATA_REF. Existing PLC applications can then be used unchanged without job management or parameterisation. In this case, it is not permitted to use MCV_GrpPathPrepare.

Block diagram



FB parameters

VAR_INPUT		
Variable name	**Data type**	**Description**
Execute	BOOL	The function block is executed on a rising edge at the input.
FileName	STRING(MCV_PROG_NAME_STRLEN)	Program name of a job.
JobID	MCV_GRP_JOB_ID	Job information
PathParameter	MCV_PATH_PARAM	Program parameters of a job. Parameters can be accessed in the called NC program (FileName). PL[0] corresponds to the variable **@PL1** and PL[19] corresponds to the variable **@PL20**. Unused parameters are prefixed by “0”.
PathMode	MCV_PATH_MODE	Parameter for program mode of a job. The following are available:

		- InitializeOnActualPosition, - SetDefaultConfig, - ReportSceneSample, - ReportRunTimeMeasure, - ReportAxesPositionSample where 0 = HLI_MC_START_MODE_NOT_USED 1 = HLI_MC_START_MODE_OFF 2 = HLI_MC_START_MODE_ON 3 = HLI_MC_START_MODE_USE_ACTUAL

BufferModeIn	MC_BUFFERMODE	BufferMode of a job.
		The following values (similar to the BufferMode interface in the MC_MovePath [▸ 27]) are possible:
		mcAborting= 0
		mcBuffered= 1
		mcBlendingPrev= 3

JobStack	HLI_MC_JOB_STACK	Information on the job source Only needed if the job was read in by an instance of the MCV_GrpGetJobRequest. Otherwise, the input must remain unassigned.
JobAttribute	UDINT	Identification number which is freely assignable by the user.
VAR_OUTPUT		
Variable name	**Data type**	**Description**
Done	BOOL	TRUE indicates that the data structure MC_PATH_DATA_REF was correctly filled out and can be transferred together with the value of the "BufferModeOut" output to an instance of the MC_MovePath [▸ 27] function block.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code
PathData	MC_PATH_DATA_REF	Structure for use at the "PathData" input of an instance of the MC_MovePath [▸ 27].
BufferModeOut	MC_BUFFERMODE	BufferMode for use at the "BufferMode" input of an instance of the MC_MovePath function block.

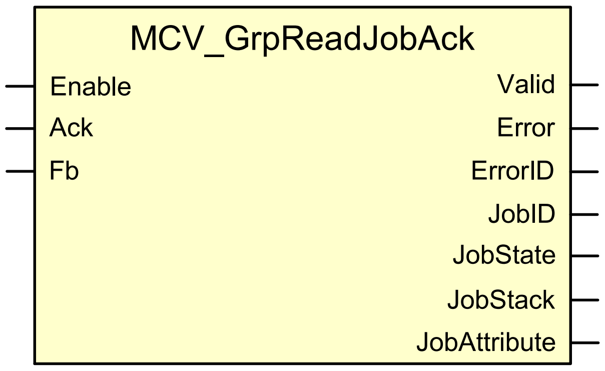
5.4 MCV_GrpReadJobAck

This function block is available as of CNC Build V3.1.3103.1.

The function block is used to evaluate the end or abort operations of jobs commanded by an MC_MovePath [▶ 27] . The instance of the MC_MovePath [▶ 27] whose job acknowledgements are evaluated is identified via the “Fb” input.

When the function block is created and the “Enable” input is set to TRUE, the function block must be called cyclically. Otherwise, job acknowledgements to any function blocks are blocked, even to other PLCopen Part 4 function blocks.

Block diagram



FB parameters

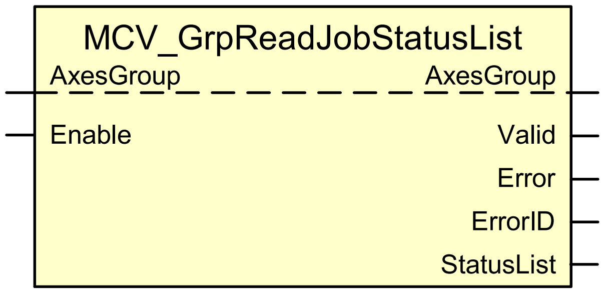
VAR_INPUT		
Variable name	**Data type**	**Description**
Enable	BOOL	As long as “Enable” is TRUE, job acknowledgements are read. Only acknowledgements are displayed if they refer to jobs sent by the instance of an MC_MovePath [▶ 27] registered at the “Fb” input. New acknowledgements are then only displayed if the “Ack” input is FALSE.
Ack	BOOL	The input defines that the information indicated at the function block outputs was read.
Fb	MCV_FB_REF	Reference to an instance of an MC_MovePath [▶ 27] function block with the output of the same type.
VAR_OUTPUT		
Variable name	**Data type**	**Description**
Valid	BOOL	TRUE indicates that new data is ready to be fetched.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code
JobID	MCV_GRP_JOB_INFO	Job description and job state.
JobState :	MCV_JOB_STATE	Job information
JobStack	HLI_MC_JOB_STACK	Information on the job source Only needed if the job is to be read by an instance of the MCV_GrpSetJobResponse.
JobAttribute	UDINT	Identification number specified by the user and output together with the state of a job.

5.5 MCV_GrpReadJobStatusList

This function block is available as of CNC Build V3.1.3103.1.

The function block is used to display the states of all jobs currently commanded for an axis group.

Block diagram



FB parameters

VAR_IN_OUT		
Variable name	**Data type**	**Description**
AxesGroup	AXES_GROUP_REF	Axis group reference
VAR_INPUT		
Variable name	**Data type**	**Description**
Enable	BOOL	If the input is TRUE, the function block outputs are updated.
VAR_OUTPUT		
Variable name	**Data type**	**Description**
Valid	BOOL	TRUE indicates that the requested update was executed.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code
StatusList	MCV_GRP_JOB_DISPLAY	Structure that describes the states of all current MC_MovePath [▶ 27] commands in a channel.

5.6 MCV_GrpResetForced

This function block is available as of CNC Build V3.1.3103.1.

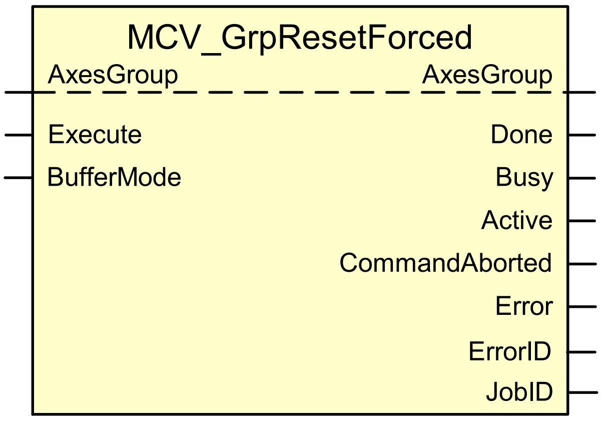
The function block leads to a controlled motion stop. It aborts every ongoing command by other motion FBs.

The axis group changes to the “GroupStopping” state until velocity 0 is reached. If the “Done” and “Execute” outputs are set to FALSE, the axis group state changes to “GroupStandby”.

The function block is not interruptible.

In addition to the motion stop which is implemented identically to the MC_GrpStop [▶ 35], this FB reverts the axis group to its initial state.

Block diagram



FB parameters

VAR_IN_OUT		
Variable name	**Data type**	**Description**
AxesGroup	AXES_GROUP_REF	Axis group reference
VAR_INPUT		
Variable name	**Data type**	**Description**
Execute	BOOL	Starts the command on the rising edge.

BufferMode	MC_BUFFER_MODE	The input defines when a job is activated provided that other jobs are already active when the FB is commanded or are waiting for execution. The following modes are supported:
		mcAborting = 0
		mcBuffered = 1

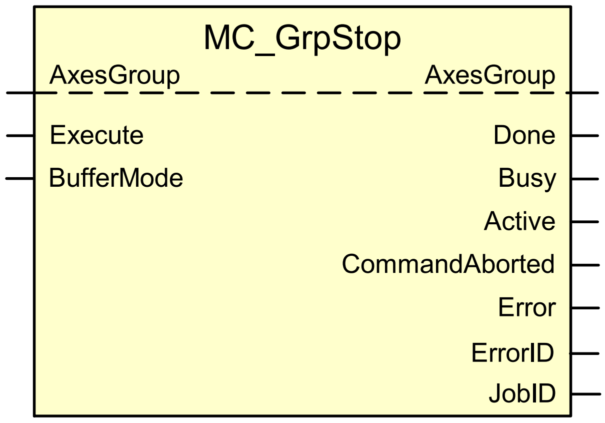
VAR_OUTPUT		
Variable name	**Data type**	**Description**
Done	BOOL	TRUE indicates that 0 velocity was reached. The axes are at standstill.
Busy	BOOL	TRUE indicates that the function block is executing a job.
Active	BOOL	The command is not yet completed.
CommandAborted	BOOL	The command to stop was aborted by another command.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code
JobID	UDINT	Ordinal number of the last job sent by the FB.

5.7 MC_GrpStop

The function block leads to a controlled motion stop. It aborts every ongoing command by other motion FBs.

The axis group changes to the “GroupStopping” state until velocity 0 is reached. If the “Done” and “Execute” outputs are set to FALSE, the axis group state changes to “GroupStandby”. The function block is not interruptible.

Block diagram



FB parameters

VAR_IN_OUT		
Variable name	**Data type**	**Description**
AxesGroup	AXES_GROUP_REF	Axis group reference
VAR_INPUT		
Variable name	**Data type**	**Description**
Execute	BOOL	Starts the command on the rising edge.

BufferMode	MC_BUFFER_MODE	The input defines when a job is activated provided that other jobs are already active when the FB is commanded or are waiting for execution. The following modes are supported:
		mcAborting = 0
		mcBuffered = 1

VAR_OUTPUT		
Variable name	**Data type**	**Description**
Done	BOOL	TRUE indicates that 0 velocity was reached. The axes are at standstill.
Busy	BOOL	TRUE indicates that the function block is executing a job.
Active	BOOL	The command is not yet completed.
CommandAborted	BOOL	The command to stop was aborted by another job.
Error	BOOL	TRUE indicates that an error occurred.
ErrorID	WORD	Error code
JobID	UDINT	Ordinal number of the last job sent by the FB.

6 Parameter

P-STUP-00206	Logical ID of a client channel in a Job Manager group
Description	The parameter defines the logical ID of a client channel. This logical ID can be requested in the agent channel by V.G.IP_NR. Due to the necessary uniqueness, no second agent channel may use the same logical ID in the same Job Manager group.
Parameter	jobmanager.group[i].master[j].log_id where i = 0, 1 (index of the Job Manager group, max. 1) where j = 0 … n (index of a continuous list element. n: application-specific)
Data type	UNS16
Data range	1 … 65536
Dimension	----
Default value	0
Remarks	Available as of V3.1.3110 IMPORTANT: If the agent is not commanded by a client, the call of "V.G.IP_NR" returns the "log_id" of the agent.

P-STUP-00207	Logical ID of a client channel in a Job Manager group
Description	The client channel (master) designated by this parameter in a Job Manager group corresponds to an existing CNC channel. It cannot be assigned to any other group, neither as an agent (slave) nor as a client (master) The number used must correspond to an existing channel number.
Parameter	jobmanager.group[i].master[j].channel_id (application-specific) where i = 0, 1 (index of Job Manager group, max. 1) where j = 0..n (index of a continuous list element. n: application-specific)
Data type	UNS16
Data range	1 to 12 (application-specific)
Dimension	----
Default value	0 *
Remarks	Available as of V3.1.3110 * The value corresponds to the statement: Parameter is not used. If all jobmanager. group[i].master[j].channel_id parameters in a Job Manager group are "0", the Job Manager is deactivated for the group. The client channel corresponds to a “normal” channel. The Job Manager configuration extends the command set to include Job Manager commands [▶ 18] (see [FCT-M10] [▶ 6]). This also includes commanding all agents (slaves) in the same Job Manager group with jobs.

P-STUP-00208	Logical ID of an agent channel in a Job Manager group
Description	The parameter defines the logical ID of an agent channel. Every agent (slave) in a Job Manager group is invoked by commands from the client by its logical ID "log_id” at the start. Due to the necessary uniqueness, no second agent channel may use the same logical ID in the same Job Manager group.
Parameter	Jobmanager.group[i].cnc_slave[j].log_idwhere where i = 0, 1 (index of Job Manager group, max. 1) where j = 0..n (index of a continuous list element. n: application-specific)
Data type	UNS16
Data range	1 … 65536
Dimension	----
Default value	0
Remarks	Available as of V3.1.3110 There are two types of agent: CNC channels and PLC units. The logical ID always refers to a particular type

P-STUP-00209	**Agent channel in a Job Manager group**
Description	The agent channel (slave) designated by this parameter in a Job Manager group corresponds to an existing CNC channel. It cannot be assigned to any other group, neither as an agent (slave) nor as client (master). The number used must correspond to an existing channel number.
Parameter	jobmanager.group[i]. cnc_slave[j].channel_id (application-specific) where i = 0, 1 (index of Job Manager group, max. 1) where j = 0..n (index of a continuous list element. n: application-specific)
Data type	UNS16
Data range	1 … (application-specific)
Dimension	----
Default value	0
Remarks	Available as of V3.1.3110 The agent channel behaves like a "normal" channel. It has the additional property of being requested by any master in the same Job Manager group to execute a job. Job completion is signalled back to the client in the controller.

P-STUP-00210	**Logical ID of agent PLC unit in a Job Manager group**
Description	The parameter defines the logical ID of an agent PLC unit. Every PLC agent (slave) in a Job Manager group is invoked by commands from the client by its logical ID "log_id”. Due to the necessary uniqueness, no second agent PLC unit may use the same logical ID in the same Job Manager group.
Parameter	jobmanager.group[i]. Plc_slave[j].log_id where i = 0, 1 (index of the Job Manager group, max. 1) where j = 0…n (index of a continuous HLI list element. n: application-specific)
Data type	UNS16
Data range	1 … 65536
Dimension	----
Default value	0
Remarks	Available as of V3.1.3110

P-STUP-00211	**Agent PLC unit in a Job Manager group**
Description	The agent PLC unit (slave) in a Job Manager group designated by this parameter corresponds to an interface on the HLI. After assignment it cannot be assigned to any other group.
Parameter	jobmanager.group[i].hli_index (application-specific) where i = 0, 1 (index of the Job Manager group, max. 1) where j = 0..n (index of a continuous HLI list element n: application-specific)
Data type	UNS08
Data range	0 to 31 (application-specific)
Dimension	----
Default value	0
Remarks	Available as of V3.1.3110 An agent PLC unit has the property of being requested by any master in the same Job Manager group to execute a job. Job completion is signalled back to the client in the controller.

P-STUP-00212	**Parameter list of an agent PLC unit in a Job Manager group**
Description	The agent PLC unit (slave) in a Job Manager group can receive parameters from the client at the start. The data structure described a declared parameter list in "jobmanager.coding[i].list" (P-STUP-00204). The parameter required here corresponds to the index [i] in "jobmanager.coding[i].list".

Parameter	Jobmanager.group[i].plc_slave[j].coding where i = 0, 1 (index of the Job Manager group, max. 1) where j = 0..n (index of a continuous HLI list element n: application-specific)
Data type	UNS08
Data range	0 to 2 (application-specific)
Dimension	----
Default value	0
Remarks	Available as of V3.1.3110

7 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.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

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 our internet page: 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
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
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Index

More Information:
www.beckhoff.com/TF5200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com