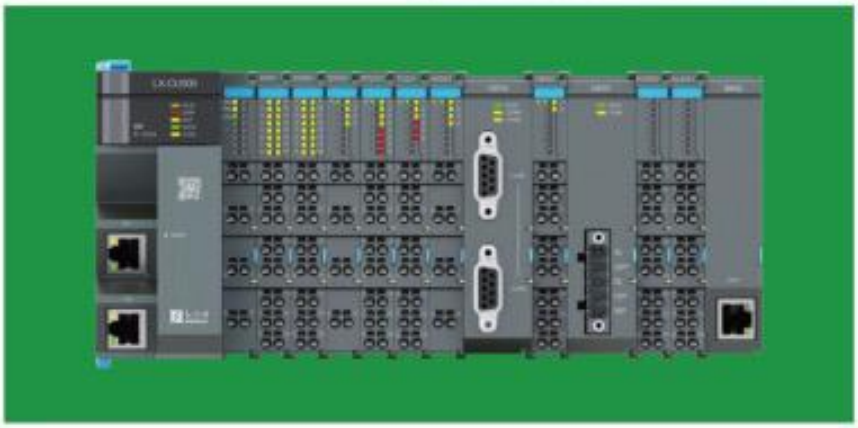


Motion Instruction Manual for
LX Series Programmable Logic Controllers

Motion Instruction Manual for LX Series Programmable Logic Controllers

2025.03

V1.2.0

Copyright Notice

Address: Di Sheng Middle Road, No.2,
Economic-Technological Development Area, 100176, Beijing, China
Tel: +86 010-5898 1588
Consulting Hotline: 400-811-1999
Technology Request: +86 010-58981514
Fax: +86 010-5898 1558
Hollysys Group Website: https://www.hollysys.com/

Factory Automation Business Website: https://fa.hollysys.com/
Email: PLC@hollysys.com

Sina weibo: http://weibo.com/hollysysplc

Contents

Chapter 1 Foreword

1.1 Purpose

This document will introduce the use of motion control instructions in the LX system, including function introduction, instruction parameter description, configuration examples, etc. For basic instructions, please refer to the Basic Instruction Manual for LX Series Programmable Logic Controller. When using this manual, please use it in conjunction with the Programming Manual for LX Series Programmable Logic Controllers to help users perform configuration more flexibly.

1.2 Object

This manual is intended for use by engineers or related professional technicians with expertise in automation and control systems.

1.3 Target Products

This manual is applicable to the following products:

- L X series PLC products

1.4 Statement of Use

- The contents of this manual have been tested according to regulations, and the diagrams are consistent with the software and hardware products described. However, errors are inevitable, and complete consistency cannot be guaranteed. If you have any suggestions for improving or correcting the contents of this manual, please let us know in time and we will make corrections in the next version.
- Due to product iteration updates, the relevant parameters and information in this manual may change without prior notice.
- Since the devices described in this manual can be used in a variety of ways, the user and the person responsible for the use of the devices must ensure that each method is admissible. HollySys will not be legally responsible for any direct or indirect losses resulting from the use or misuse of these devices.
- Due to uncertainties in the actual application, HollySys assumes no responsibility for the direct use of the data provided in this manual.

1.5 Safety Precautions

1.5.1 Safety Statement

1. Please read and comply with these safety precautions before using this product.

2. To ensure personal and device safety, when using this product, please strictly abide by the safety precautions on the product label and in the manual.

3. The words "Note", "Caution", "Warning" and "Danger" in this manual do not represent all the safety precautions that shall be observed but only serve as a supplement to all precautions.

4. This product shall be used in an environment that meets the design specifications. Otherwise, it may cause faults. Device damage caused by failure to comply with relevant regulations is not covered by the product quality guarantee.

5. HollySys shall not be responsible for any personal safety accidents or property losses caused by any operation that does not comply with the provisions of this manual or does not meet the requirements.

1.5.2 Warning

For your safety and to avoid property losses, please pay attention to the warnings in this manual. The following warnings are indicated according to the hazard level from high to low. When there are multiple hazard levels, only the highest level is prompted. If the highest level is a warning that may cause personal injury, there may also be a warning that may cause property loss.

- Warnings on personal safety: Indicated by a warning triangle  .
- Warnings on property loss: Without any warning triangle.

 Danger
It indicates that death or serious personal injury may be caused if operations are not carried out as specified.

 Warning
It indicates that death or serious personal injury may be caused if operations are not carried out as specified.

 Caution
It indicates that minor personal injuries may be caused if operations are not carried out as specified.

Note
It indicates that property losses may be caused if operations are not carried out as specified.

Chapter 2 Document Guide

2.1 Related Manuals

The types of manuals for this product are shown in the table below. Please select and read according to the purpose of use.

Manual Name	Purpose	Content
Module Manual for LX Series Programmable Controllers	Learn about the hardware modules of the LX system	Describes all hardware modules, including: Module functions Hardware interface Wiring Indicator Technical parameters
Communication Manual for LX Series Programmable Controllers	Guide users to build the LX system communication network	Describe the communication networks supported by the LX system, including: Introduction to communication protocols and use of functions
Programming Manual for LX Series Programmable Controllers	Learn about the use of FA-AutoThink programming software	Describe the interface, functions, and related operations of FA-AutoThink programming software, including: software interface introduction Project creation Programming Online debugging function
Basic Instruction Manual for LX Series Programmable Controllers	Learn about the use of basic control instructions of the LX system	Describe the use of basic control instructions, including: Instruction function Parameter Example
PLCopen Instruction Manual for LX Series Programmable Logic Controllers	Learn about the use of PLCopen instructions of the LX system	Describe the use of PLCopen instructions, including: Parameter Instruction function Example Precautions See also

2.2 Usage Convention

The following symbols are used in this manual:

- Tip: This symbol indicates helpful tips or suggestions for using the product more efficiently.

2.3 How to Get the Manual

If you need the electronic PDF file of this manual, please visit the official website of HollySys (https://cn.hollysys.com/other/download.html) to download the file.

Chapter 3 Revision History

Version	Revision Date	Revision Content
V1.0.0	July 2023	Created
V1.1.0	October 2023	Added a new instruction: HMC_SynchroToAlm (Synchronizing to algorithm tasks) Deleted the following instructions: HMC_SetSSIEncoderBit (Setting the number of SSI encoder bits), HMC_SetSSIEncoderDataType (Setting SSI encoder coding system), HMC_SetSSIEncoderFreq (setting SSI bus clock frequency), HMC_SetSSIEncoderTp (setting SSI bus data latch time), EtherCAT protocol related instructions
V1.1.1	December 2023	Added parameter table default values and power-failure protection
V1.1.2	August 2024	Added full text instruction type description
V1.2.0	March 2025	Update copyright notice information

Chapter 4 Overview of Instructions

4.1 List of Instructions

Instruction Library	Instruction	Name	Function Description
Basic Parameters	AxisID	Axis number	Axis number, indicating the axis number of each group of axis parameters.
	AxisType	Axis type	Axis types.
	AxisState	Axis status	Axis PLCopen status.
	Units	Unit conversion factor	Set the axis parameter Units for the unit conversion factor for the specified axis.
Status Parameters	MCmdType	Current instruction type	Current instruction type of the axis.
	NCmdType	Next instruction type	Next instruction type of the axis.
	AxisStatus	Axis status	Current running status.
	AxisErrMask	Axis status mask	Axis status exception mask.
	LimitState	Over-limit status	Describe the current over-limit status.
Position Related Parameters	MPos	Feedback position	Current measurement feedback position.
	DPos	Target position of this cycle	Current target position.
	FE	Deviation	Real-time follow-up error.
	StopFETol	Whether the feedback position reaches the dead band of the target position	Determine whether MPos reaches the target position in speed mode.
	RepMode	Position stroke mode	Set the position stroke mode of this axis.
	RepDist	Cycle stroke length	Set the cycle stroke length.
	Remain	Remaining positions	Remaining displacement from the instruction target position.
Speed Related Parameters	Speed	Target speed	Set the target solving speed of an instruction.
	Accel	Acceleration	Set acceleration.
	Decel	Deceleration	Set acceleration.

Instruction Library	Instruction	Name	Function Description
	MpSpeed	Feedback speed	Feedback speed measured in real time.
	VpSpeed	Solving speed	Speed value solved in real time by the motion instruction.
	Direction	Direction of solving speed	Speed direction solved in real time by the motion instruction.
	Merge	Merging function mode	Set the speed merging function.
	CreepSpeed	Origin regression creep speed	Set the creep speed.
	FastDecel	Fast deceleration	Set the fast deceleration.
	Jerk	Jerk	Set the jerk.
	JerkMode	Acceleration mode	Change acceleration, deceleration, and speed through jerk.
	CmdStopMode	Instruction end judgment method	The mode to determine whether the bus axis motion instruction has been executed.
	MoveAbsMode	Direction selection method for absolute motion of cycle stroke axis	Set the absolute motion direction for the cycle stroke axis.
	CancelMode	Instruction stop mode when Cancel	Set the instruction stop mode when Cancel.
	CancelPos	Stop position when Cancel	The position the axis will reach when the Cancel instruction is completed.
Coefficient Factor Parameter	Kp	PID proportional coefficient	Proportional coefficient operation
	Ki	PID integral gain	Integral gain operation
	Kd	Differential gain of PID	Differential gain operation
	Kov	Measured speed gain	Measured speed gain
	Kvff	Speed feedforward gain	Speed feedforward gain
	Kaff	Acceleration feedforward gain	Acceleration feedforward gain
	Ko	PID output amplification factor	Servo output ratio, which is the amplification factor of the servo output value after PID calculation.
Limit Parameters	FELimits	Follow-up error limit	Set the normal range of FE (Follow-up error).
	FsLimitMode/RsLimitMode	Instruction cancellation	Set the instruction cancellation

Instruction Library	Instruction	Name	Function Description
		method during forward/reverse limit	method during setting forward/reverse limit.
	FHLimitIn/RHLimitIn	Forward/Reverse hard limit channel number	Set the DI channel number corresponding to the forward/reverse hard limit stroke switch.
	FSLimit/RSLimit	Forward/Reverse soft limit boundary value	Setting forward/reverse soft limit boundary value
	DACLimitHigh/DACLimitLow	DAC output upper/lower limit	Set the upper/lower limits of DACOut.
Origin and Hold Parameters	HomeState	Origin regression status	Understand the current origin regression situation.
	HomeIn	DI channel used for origin regression	DI channel number used for origin regression.
	HomeCapIn	High-speed capture channel used for origin regression	It is the high-speed capture channel number used by HMC_HomeEx, which is set when the HMC_HomeEx instruction is issued.
	HoldSpeed	Hold speed	When the hold speed switch is triggered, the axis will accelerate or decelerate from the current speed to HoldSpeed.
	HoldIn	Hold speed switch DI channel number	It is the trigger source DI channel number of the hold speed function.
Input Related Parameters	KeNumerator/KeDenominator	Numerator/denominator of encoder input ratio Ke	"Unit conversion factor" for bus axis.
	EncoderValue	Internal encoder value	When the axis feeds back position information through the encoder, the axis parameter represents the internal encoder value.
	FeedBackSrcMode	Axis position feedback source mode	Set the axis feedback value Mpos input source method.
	FeedBackSrcValue	Axis feedback self-defined input real-time value	Configure a parameter address as the feedback input source of the axis at this time.
	FeedBackDataType	Axis feedback self-defined input variable data type	Data type of user-defined input variables.

Instruction Library	Instruction	Name	Function Description
Output Related Parameters	KsNumerator/KsDenominator	Numerator/denominator of stepper output ratio Ks	Numerator/denominator of stepper output ratio Ks, which is set through HMC_SetKs.
	PulseNum	Number of periodic output pulses of stepper axes	Number of pulses output by the stepper axis per servo cycle.
	ServoLoop	Closed-loop output switch	Set this switch to determine whether the position closed-loop takes effect.
	DAC	Speed mode open-loop output value	Output value of DACOut when ServoLoop=0.
	DACOut	Speed mode output value	Speed output result of the current axis.
	PosOffset	DACOut positive dead band value	Positive dead band value of bus axis output.
	NegOffset	DACOut negative dead band value	Negative dead band value of bus axis output.
	ZeroWindow	DACOut zero gap dead band	DACOut zero gap dead band.
	OutDstMode	DACOut output target mode	Set OutDstMode to specify the final output target of the output value DACOut.
	OutDstValue	DACOut output self-defined address real-time value	When OutDstMode=1, OutDstValue displays user-defined variable values in real time.
	OutDstDataType	DACOut output self-defined variable data type	OutDstValue displays user-defined variable values in real time.
System Startup Instruction	HMC_DriveEnable	Servo enable	Control the on/off of the enable signals of all servo drivers connected to the controller.
	HMC_GetDriveEnableState	Getting servo enable status	Get the servo total enable (DriveEnable) status.
Origin Search and Position Definition	HMC_Home	Origin regression	Determine the origin of the axis by detecting the origin switch signal or the Z-phase signal of the servo motor encoder.
	HMC_HomeEx	Advanced origin regression	Determine the origin of the axis by detecting the origin switch signal or the Z-phase signal of the

Instruction Library	Instruction	Name	Function Description
			servo motor encoder.
	HMC_DefOrigin	Setting origin	Change the origin of a certain axis coordinate system and set the position in the original coordinate system calibrated by the parameter dMpos as the new coordinate system origin.
	HMC_DefPos	Setting current position	Moving coordinate system, with the current position defined as dMpos.
One-way Motion	HMC_Forward	Forward motion	Continuous forward motion, with the motion speed determined by the axis parameter Speed.
	HMC_Reverse	Reverse motion	Continuous reverse motion, with the motion speed determined by the axis parameter Speed.
	HMC_Forward Ex	Advanced forward motion	During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.
	HMC_ReverseEx	Advanced reverse motion	During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.
Point Motion	HMC_Move	Relative motion	Let an axis move at the preset speed according to the acceleration and deceleration parameters to the relative displacement relative to the position when this instruction starts to be executed.
	HMC_MoveAbs	Absolute motion	Let an axis move to the coordinate position d Distance according to the preset acceleration, speed and acceleration parameters.
	HMC_MoveEx	Advanced relative motion	When this instruction is executed, first modify the axis parameter Speed = function parameter dSpeed, and then move at the

Instruction Library	Instruction	Name	Function Description
			Speed to the relative displacement relative to the position when this instruction starts to be executed.
	HMC_MoveAbsEx	Advanced absolute motion	Axis parameter Speed= function parameter dSpeed, allowing an axis to move to the absolute position according to the target speed set by this instruction.
Instruction Correction	HMC_Cancel	Canceling instruction	Cancel the instruction in the motion buffer of the specified axis.
	HMC_MoveModify	Target position change	Modify the target position of the HMC_Move/HMC_MoveAbs instruction in real time, and the modified target position is dDpos.
	HMC_MoveModifyEx	Advanced target position change	Modify the target position of the HMC_Move/HMC_MoveAbs instruction in real time.
Two-Axis Linear Interpolation	HMC_Move2D	Two-axis linear interpolation	Interpolation function where the motion profile on the two-dimensional plane is a straight line.
	HMC_Move2DEx	Advanced two-axis linear interpolation	Interpolation function where the motion profile on the two-dimensional plane is a straight line.
	HMC_MoveAbs2D	Two-axis absolute linear interpolation	Interpolation function where the motion profile on the two-dimensional plane is a straight line.
	HMC_MoveAbs2DEx	Advanced two-axis absolute linear interpolation	Interpolation function where the motion profile on the two-dimensional plane is a straight line.
Three-Axis Linear Interpolation	HMC_Move3D	Three-axis linear interpolation	Interpolation function in which the motion contour is a straight line in a three-dimensional space.
	HMC_Move3DEx	Advanced three-axis linear interpolation	Interpolation function in which the motion contour is a straight line in a three-dimensional space.
	HMC_MoveAbs3D	Three-axis absolute linear interpolation	Interpolation function in which the motion contour is a straight

Instruction Library	Instruction	Name	Function Description
			line in a three-dimensional space.
	HMC_MoveAbs3DEx	Advanced three-axis absolute linear interpolation	Interpolation function in which the motion contour is a straight line in a three-dimensional space.
Multi-Axis Linear Interpolation	HMC_MoveND	N-axis linear interpolation	Interpolation function in which the motion contour is straight in any N-dimensional coordinate system.
	HMC_MoveNDEx	Advanced N-axis linear interpolation	Interpolation function in which the motion contour is straight in any N-dimensional coordinate system.
	HMC_MoveAbsND	N-axis absolute linear interpolation	Interpolation function in which the motion contour is straight in any N-dimensional coordinate system.
	HMC_MoveAbsNDEx	Advanced N-axis absolute linear interpolation	Interpolation function in which the motion contour is straight in any N-dimensional coordinate system.
Planar Arc Interpolation	HMC_MoveCircle	Arc interpolation	It is an interpolation function in which the motion contour is an arc on a two-dimensional plane.
	HMC_MoveCircleEx	Advanced arc interpolation	It is an interpolation function in which the motion contour is an arc on a two-dimensional plane.
Spherical Interpolation	HMC_MoveSpherical	Spherical arc interpolation	It is an interpolation function in which the motion contour is a spherical arc in a three-dimensional space.
	HMC_MoveSphericalEx	Advanced spherical arc interpolation	It is an interpolation function in which the motion contour is a spherical arc in a three-dimensional space.
Helical Interpolation	HMC_MoveHelical	Helical interpolation	The three axes work together to complete the cylindrical helix trajectory.
	HMC_MoveHelicalEx	Advanced helical interpolation	The three axes work together to complete the cylindrical helix trajectory.
Spline	HMC_MoveSplineND	Spline interpolation	Taking the current point as the starting point, execute N-axis

Instruction Library	Instruction	Name	Function Description
Interpolation			spline interpolation motion.
	HMC_MoveSplineNDEx	Advanced spline interpolation	Taking the current point as the starting point, execute N-axis spline interpolation motion.
	HMC_CalcMoveSplineNDdata	Generating spline data	Generate an N-axis interpolated cubic spline curve.
Electronic Gear	HMC_Gear	Electronic gear	Achieve shaftless transmission.
	HMC_SetGearRatio	Setting electronic gear ratio	Call this instruction in real time to dynamically change the electronic gear ratio during operation.
	HMC_GetGearMaster	Getting electronic gear master axis	Get the running HMC_Gear master axis number.
	HMC_GetGearRatio	Getting electronic gear ratio	Get the currently running HMC_Gear instruction gear ratio.
	HMC_SetGearPhaseOffset	Setting electronic gear phase delay	Adjust the following phase of the electronic gear slave axis.
	HMC_GetGearPhaseOffset	Getting electronic gear phase delay	Get the set phase delay of electronic gear.
	HMC_SetGearTransationSwitch	Electronic gear transition switch	Turn on/off the electronic gear smoothing function.
Electronic Cam	HMC_Cam	Electronic cam	By selecting discrete coordinate points on the target motion trajectory (two-dimensional), the electronic cam function can be used to approximate the target motion trajectory through master-slave axis linkage.
	HMC_SplineCam	Spline cam	Use the polyline connection to connect the master-slave axis position points to generate a trajectory.
	HMC_GetCamMasterPosIndex	Getting electronic cam master axis position index	Get the currently running cam master axis position index.
	HMC_ClcMoveLinkData	Follow-up shearing operation	Generate master-slave axis position data according to the set relationship.
	HMC_ClcFlexLinkData	Flying shear/rotary cutting operation	Generate master-slave axis position data according to the set relationship.
	HMC_CancelREPCam	Cyclic cam condition	Use the condition cancellation

Instruction Library	Instruction	Name	Function Description
		cancellation control	function on a running cyclic cam linkage instruction.
	HMC_MoveLink	Follow-up shearing motion	Integrate cam and follow-up shear operation functions to control the slave axis to follow the master axis to move according to certain rules.
	HMC_MoveFlexLink	Flying shear/rotary cutting motion	Cam function with master-slave axis position and acceleration and deceleration planning.
Motion Superimposing	HMC_Superimpose	Superimposing	Achieve simple superimposing of the incremental displacements of the two master axes.
	HMC_SuperImposeEx	Advanced superimposition	Achieve linear superimposing of the incremental displacements of the two master axes.
	HMC_SetSuperimposeRatio	Setting superimposition ratio	Set the superimposition ratio.
	HMC_GetSuperimposeMaster1	Getting superimposed master axis 1	Get the first superimposed master axis from the actions executed from input to output and signal got.
	HMC_GetSuperimposeMaster2	Getting superimposed master axis 2	Get the second superimposed master axis.
	HMC_GetSuperimposeRatio1	Getting superimposed master axis 1 ratio	Get the first superimposed master axis ratio.
	HMC_GetSuperimposeRatio2	Getting superimposed master axis 2 ratio	Get the second superimposed master axis ratio.
	HMC_SetAddAxis	Setting summation motion axes	Used to complete the summation of two-axis motion.
	HMC_GetAddAxis	Getting summation motion superimposition axes	Used to get the summation motion superimposition axis number, with 255 indicating that no superimposed axis is associated.
Low Speed in-Position Output	HMC_NormalSwitch1Dconfig	Setting one-dimensional low-speed output	Configure the one-dimensional low-speed output.
	HMC_NormalSwitch2DConfig	Setting two-dimensional low-speed output	Configure the two-dimensional low-speed output.
	HMC_GetNormalSwitchNum	Getting the number of	Get the number of segments that

Instruction Library	Instruction	Name	Function Description
		outputs	the low speed in-position output with the specified ID has been executed.
	HMC_GetNormalSwitchState	Getting low speed output status	Get the current status of the low speed output with the specified ID.
	HMC_CloseNormalSwitch	Closing low speed output	Close the low speed in-position output of this ID.
Oscilloscope Function Instruction	HMC_SetSampleVar	Setting sampling variables	Collect system data such as axis parameters, I/O data, and intermediate/system variables.
	HMC_SetSamplePara	Setting sampling parameters	Configure the sampling interval, number of sampling points, and sampling data storage area address of the specified sampling channel.
	HMC_GetSampleData	Getting sample data	Get the sample data of a sub-channel of the specified channel with the serial number uiDataNum.
	HMC_TriggerSample	Triggering sampling	Trigger the specified sampling channel to start sampling.
	HMC_DisableSample	Stopping sampling	Disable the specified sampling channel from sampling.
	HMC_TriggerScope	Triggering oscilloscope	Trigger the oscilloscope to begin execution
	HMC_GetSampleNum	Getting sampled number	Get the current number of completed samples of the specified sampling channel.
	HMC_SetSampleSingleVar	Setting a single sampling variable	Configure the source of data to be sampled for sub-channel 0 of the specified sampling channel.
Motion Hold Function Instruction	HMC_HoldConfig	Setting hold speed switch	Set the hold speed switch.
Motion Look-Ahead	HMC_SetMerge	Setting merging function	Set the Merge mode of axis parameters.
	HMC_ClcJerk	Shock operation	Calculate Jerk based on Speed and curvature.
	HMC_SetMergeAngle	Setting merging angle	Set the merging angle of the axis.
	HMC_GetMergeStopAngle	Getting merging stop	Get the interpolation motion stop

Instruction Library	Instruction	Name	Function Description
		angle	angle.
	HMC_GetMergeDecelAngle	Getting merging deceleration angle	Get the interpolation motion deceleration angle.
	HMC_SetMergeSpeedTolerRatio	Setting speed fluctuation rejection ratio	Set the speed fluctuation rejection ratio parameter during motion look-ahead.
	HMC_GetMergeSpeedTolerRatio	Reading speed fluctuation rejection ratio	Read the current speed fluctuation rejection ratio.
Curve Speed Limit	HMC_SetCurveJerkLimit	Setting curve shock upper limit	Set the upper limit of physical shock during curve interpolation.
	HMC_GetCurveJerkLimit	Reading curve shock upper limit	Read the set upper limit of physical shock during curve interpolation.
Tangential Tracking	HMC_MoveTang	Tangential tracking	Automatically adjust the tool direction and the tangential direction of the cutting trajectory in real time.
	HMC_MoveTangConfigCornerMode	Tangential tracking corner retract settings	Perform the corner retract function to protect the tool and ensure the processing effect.
	HMC_GetMoveTangCornerState	Getting tangential tracking corner retract status	Get the tangential tracking corner retract status.
	HMC_MoveTangCornerFollowNextCmd	Positioning C-axis to the starting tangential angle of the next instruction	Drive the C-axis to position to the starting tangential angle of the next instruction.
	HMC_MoveTangCornerContinue	Continuing tangential tracking	When it is paused in the corner to wait for the tool to be retracted, use this instruction to resume tangential tracking and continue running.
	HMC_GetMoveTangDestDirection	Getting target direction position of tangential tracking	Get the target direction position of tangential tracking.
	HMC_GetMoveTangDestVelocity	Getting target speed of tangential tracking	Get the change speed of the tangential angle of the tangential tracking target.
	HMC_GetMoveTangDeviation	Get direction deviation of tangential tracking	Get the deviation between the current instruction position of the

Instruction Library	Instruction	Name	Function Description
			tangential tracking C-axis (ucAxisC, tool rotation axis) and the target tangential angle.
	HMC_GetMoveTangMasterAxisX	Getting tangential tracking master axis X	Get the X-axis of the current tangential tracking instruction.
	HMC_GetMoveTangMasterAxisY	Getting tangential tracking master axis Y	Get the Y-axis of the current tangential tracking instruction.
Multi-Axis Synchronization Control	HMC_GantrySynchro	Gantry synchronization control	Perform dynamic synchronization of master and slave axes.
	HMC_GantryInit	Alignment initialization of gantry synchronized back to reference point position	Initialize the alignment of gantry synchronization back to the reference point position.
	HMC_SetGantryAxisPID	Setting gantry synchronization axis compensation PID parameters	Set the dynamic correction PID parameters of the specified gantry axis in a gantry group.
	HMC_GetGantryAxisPID	Reading gantry synchronization axis compensation PID parameters	Read the dynamic correction PID parameters of the specified gantry axis in a gantry group.
	HMC_GantryGroupDefPos	Setting the current position of a gantry synchronization axis group	Redefine the position of all gantry axes in a gantry group.
	HMC_GetGantryInitState	Reading the current status of a gantry synchronization control function	Read the current execution status of the instruction.
Axis Position Compensation Function	HMC_SetAxisCompenPos	Axis position compensation quantity settings	Set axis position compensation quantity.
	HMC_SetAxisCompenPara	Setting axis position compensation motion parameters	Set the kinematic parameters of the axis position compensation motion, which takes effect in real time.
	HMC_TrigAxisCompens	Triggering position compensation	Trigger an axis position compensation action.
	HMC_StopAxisCompens	Stopping position compensation	Immediately stop the axis position compensation action in progress.
	HMC_GetAxisCompenState	Getting axis position	Get the axis position

Instruction Library	Instruction	Name	Function Description
		compensation status	compensation status.
Stewart 6-Degree-of-Freedom (DOF) Parallel Mechanism	HMC_StewartControl	Stewart mechanism control	This instruction completes the kinematic transformation of the Gough-Stewart 6-DOF parallel mechanism.
5-DOF SCARA Robot Arm	HMC_Scara5Control	5-DOF SCARA robot arm control	This instruction completes the kinematic transformation of the 5-DOF SCARA mechanism.
	HMC_Scara5Forward Kinematics	Forward kinematics of 5-DOF SCARA	This instruction completes the forward kinematics of the 5-DOF SCARA mechanism.
	HMC_Scara5ReverseKinematics	Inverse kinematics of 5-DOF SCARA	This instruction completes the inverse kinematics of the 5-DOF SCARA mechanism.
4-DOF Serial Robot Arm	HMC_Serial4Control	4-DOF serial robot arm control	This instruction completes the kinematic transformation of the 4-DOF serial robot arm.
	HMC_Serial4Forward Kinematics	Forward kinematics of 4-DOF serial robot arm	This instruction completes the forward kinematics of the 4-DOF serial robot arm.
	HMC_Serial4ReverseKinematics	Inverse kinematics of 4-DOF serial robot arm	This instruction completes the inverse kinematics of the 4-DOF serial robot arm.
Self-Defined Motion Control Function	HMC_SetDposSwitch	Dpos planning switch switching	This instruction switches the Dpos planning switch of the specified axis.
	HMC_SetDposVal	Dpos planning	Set the Dpos planning.
	HMC_SynchroToAlm	Synchronizing to algorithm tasks	This instruction can be used to synchronize hard real-time tasks with algorithms inside the MC.
Axis Parameter Setting Instruction	HMC_SetAxisType	Setting axis type	Set the axis type of the specified axis.
	HMC_HwLimitConfig	Setting hard limit switch	Set the hard limit switch of the specified axis.
	HMC_SetUnits	Setting user units	Set the axis parameter Units for the unit conversion factor for the specified axis.
	HMC_SetJerkMode	Setting trapezoidal/S-shaped acceleration	Set JerkMode.
	HMC_SetKe	Setting Ke	Set the numerator (axis parameter

Instruction Library	Instruction	Name	Function Description
			KeNumerator) and denominator (axis parameter KeDenominator) of the axis parameter Ke.
	HMC_SetKs	Setting Ks	Set the numerator (axis parameter KsNumerator) and denominator (axis parameter KsDenominator) of the axis parameter stepper output ratio Ks.
	HMC_SetFeedBackSrcAddr	Set user-defined feedback input address	Configure the user-defined input source for the actual axis position Mpos.
	HMC_SetOutDstAddr	Setting user-defined output address	It is to set the self-defined output address of the position closed-loop output value DACOut.
	HMC_SetSSIEncoderBit	Setting the number of SSI encoder bits	Set the number of SSI encoder bits.
	HMC_SetSSIEncoderDataType	Setting SSI encoder coding system	Configure the axis coding system for SSI encoders.
	HMC_SetSSIEncoderFreq	Setting SSI bus clock frequency	Configure the SSI bus clock frequency.
	HMC_SetSSIEncoderTp	Setting SSI bus data latch time	Configure the SSI bus data latch time.
	HMC_SetCmdBufferLoad Mode	Setting axis instruction loading mode	Set the axis instruction loading mode.
	HMC_GetCmdBufferLoad Mode	Getting axis instruction loading mode	Get the axis instruction loading mode.
	HMC_SetCmdStopMaxTime	Setting the Maximum Instruction End Waiting Time	Set the maximum instruction end waiting time.
	HMC_GetAxisSlaveAddr	Getting the Station Address Corresponding to the Axis	Obtain the slave station address corresponding to the specified axis number.
	HMC_GetAxisTorque	Getting Axis Torque	Get the actual torque value of the slave station corresponding to the specified axis number.
	HMC_AxisPotInput	Getting Axis Forward Hard Limit Status	Obtain the axis forward hard limit status corresponding to the slave station with the specified axis number.
	HMC_AxisNotInput	Getting Axis Reverse Hard Limit Status	Obtain the axis reverse hard limit status corresponding to the slave

Instruction Library	Instruction	Name	Function Description
			station with the specified axis number.
	HMC_AxisStandStillStatus	Getting Axis Stop Status	Check whether the slave station with the specified axis number is in the stop status.
	HMC_AxisSlaveErrorStatus	Getting Axis Slave Error Status	Get the error status of the associated slave station of the specified axis number.
	HMC_AxisSlaveErrorID	Getting Axis Slave Error ID	Get the error number of the slave station associated with the specified axis number.
	HMC_GetAxisStatus	Getting Axis Parameter AxisStatus	Get the axis parameter AxisStatus (axis status) of the slave station with the specified axis number.
	HMC_GetAxisMcmdType	Getting Axis Parameter McmdType	Get the current instruction type of the slave station with the specified axis number.
	HMC_GetAxisNcmdType	Getting Axis Parameter NcmdType	Get the next instruction type of the slave station with the specified axis number.
	HMC_GetAxisType	Getting Axis Parameter AxisType	Get the axis parameter AxisType (axis type) of the slave station with the specified axis number.
	HMC_GetAxisLimitState	Getting Axis Parameter LimitState	Get the axis parameter LimitState (over-limit status) of the slave station with the specified axis number.
	HMC_GetAxisUnits	Getting Axis Parameter Units	Get the axis parameter Units (unit conversion factor) of the slave station with the specified axis number.
	HMC_SetAxisSpeed	Setting Axis Parameter Speed	Set the axis parameter Speed (target speed) of the slave station with the specified axis number.
	HMC_SetAxisAccel	Setting Axis Parameter Accel	Set the axis parameter Accel (acceleration) of the slave station with the specified axis number.
	HMC_SetAxisDecel	Setting Axis Parameter Decel	Set the axis parameter Decel (deceleration) of the slave station with the specified axis number.
	HMC_SetAxisJerk	Setting Axis Parameter	Set the axis parameter Jerk (jerk)

Instruction Library	Instruction	Name	Function Description
		Jerk	of the slave station with the specified axis number.
	HMC_SetAxisDAC	Setting Axis Parameter DAC	Set the axis parameter DAC (speed mode open-loop output value) of the slave station with the specified axis number.
	HMC_SetAxisCreepSpeed	Setting Axis Parameter CreepSpeed	Set the axis parameter CreepSpeed (origin regression creep speed) of the slave station with the specified axis number.
	HMC_GetAxisHomeState	Getting Axis Parameter HomeState	Get the axis parameter HomeState (origin regression status) of the slave station with the specified axis number.
	HMC_GetAxisHardLimitCfg	Getting Axis Limit Configuration	Get the axis hard limit configuration data of the specified slave station.
	HMC_GetAxisVpSpeed	Getting Axis Parameter VpSpeed	Get the axis parameter VpSpeed (solving speed) of the slave station with the specified axis number.
	HMC_GetAxisMpSpeed	Getting Axis Parameter MpSpeed	Get the axis parameter MpSpeed (feedback speed) of the slave station with the specified axis number.
	HMC_GetAxisDpos	Getting Axis Parameter Dpos	Get the axis parameter Dpos (target position of this cycle) of the slave station with the specified axis number.
	HMC_GetAxisMpos	Getting Axis Parameter Mpos	Get the axis parameter Mpos (feedback position) of the slave station with the specified axis number.
	HMC_GetAxisEndPos	Getting Axis Parameter End Pos	Get the axis parameter End Pos (instruction target position) of the slave station with the specified axis number.
	HMC_GetAxisRemain	Getting Axis Parameter Remain	Get the axis parameter Remain (remaining positions) of the slave station with the specified axis number.
Status Read	HMC_GetSysErr	Getting system error	Get the latest system error that

Instruction Library	Instruction	Name	Function Description
Instruction		code	occurred on the system.
	HMC_GetUserErrID	Getting user error code	Return the user error code.
	HMC_GetUserErrNum	Getting the number of user errors	Returns the number of user errors that have occurred.
	HMC_GetUserErrMes	Getting additional information about user error codes	Return additional information about user error codes.
	HMC_GetAllAxisErr	Getting overall abnormal status of all axes	Get the overall abnormal status of all axes.
	HMC_GetCmdErr	Getting axis instruction error	According to the axis instruction ID, return the axis instruction error.
	HMC_GetCmdState	Getting axis instruction status	It is used to return the axis instruction status according to the axis instruction ID.
	HMC_WaitCmd Finish	Waiting for the instruction to complete	Wait until "A motion instruction has completed running" before returning.
	HMC_WaitCmdLoaded	Waiting for instructions loading	Wait until "A motion instruction (uiAxisCmdID) is loaded", that is, the instruction just starts executing, before returning.
	HMC_WaitAxisIdle	Waiting for axis to be free	Wait until "The motion instruction queue of an axis (ucAxisID) is empty" before returning, otherwise it will remain blocked and will not return.
	HMC_GetCmdBufferState	Getting axis instruction cache status	Get the instruction cache status of an axis (ucAxis).
Fault Clearing Instructions	HMC_ClearAxisErr	Clearing axis errors	Clear errors for an axis.
	HMC_ClearAllErr	Clearing all axis errors	Clear all axis errors.

4.2 Use of Instructions

Before using the instructions, please first understand the relevant information involved in the instructions, so that you can better understand and use the instructions.

Information Item	Description
Instruction/Functional	Indicate the instruction name, for example: ADD

Block	
Name	Indicate the Chinese name of the instruction, for example, addition instruction
FB/FUN	Indicate whether the instruction is a functional block (FB) or function (FUN) Functional block instructions: can be called by programs or FBs. Function instructions: can be called by any one of the programs, FBs, and FUNs.
Function	Explain the instruction function
Parameter	Describe the input, output, intermediate variables and other parameter information of the instruction.
Example	Instruction application examples are described by programming examples in LD and ST.
Precautions for Use	Explain the precautions when using instructions, including applications in special scenarios and the occurrence of abnormal situations.
Reference	Other content and information you need to refer to when using this instruction

Chapter 5 HMC Motion Instruction Library

5.1 AXIS (Axis Parameters)

Each axis corresponds to the same set of axis parameters. The user can understand the current operating status of the axis by viewing the real-time value of the axis parameters, and can also modify the axis parameters indirectly through direct modification or operation control instructions to change the operating status of the axis.

Axis parameters can be divided into "read-only" and "read/write" according to their read/write properties. "Read-only" parameters cannot be assigned a value during configuration, nor can they "write variables" through AutoThink, while "read/write" parameters can be assigned values during configuration, and can also write variables through AutoThink. "Read-only" parameters are divided into two types. The first type only reflects some statuses of the axis (such as LimitState) or the currently calculated or measured values (such as VpSpeed, MpSpeed), and such parameters are solved and refreshed by the control algorithm. The second is read-only parameters (such as AxisType) that the user can set through the settings of the function.

In AutoThink, axis parameters are organized in the form of functional blocks as their members. All axis parameters of each axis constitute a functional block variable, which supports user renaming. For example, when the user wants to access the Speed of axis 0 in a method similar to Axis0.Speed, if the user changes the functional block variable name of Axis 0 to AxisLaser, the user can access the Speed of Axis 0 in the method of AxisLaser.Speed.

In AutoThink, the axis parameters of all axes are defined in a fixed area of the data area. To prevent data conflicts, AutoThink does not allow users to define other variables in this area.

5.1.1 Basic Parameters (Basic)

5.1.1.1 AxisID (Axis number)

5.1.1.1.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	0~63
Related Chapters	None

5.1.1.1.2 Meaning description

Axis number, indicating the axis number of each group of axis parameters.

5.1.1.2 AxisType (Axis type)

5.1.1.2.1 Basic properties

Type	EmAxisType
Read/Write Property	Read only
Set Function	HMC_SetAxisType
Initial Value	Virtual (0): Virtual axis
Value Range	Unused (-1): The axis is not used Virtual (0): Virtual axis EtherCAT_Position (50): EtherCAT bus connection axis, position control EtherCAT_Speed (51): EtherCAT bus connection axis, speed control EtherCAT_Torque (52): EtherCAT bus connection axis, torque control EtherCAT_Encoder (53): EtherCAT bus connection axis, encoder counting (Encoder type axes do not accept motion control instructions, but only track encoder values)
Related Chapters	HMC_SetAxisType (Setting Axis Type)

5.1.1.2.2 Meaning description

The function HMC_SetAxisType can be used to set the axis type. Different models of controllers and different axis numbers of controllers with the same model support different axis types. See HMC_SetAxisType (Setting Axis Type).

Different axis types will affect the instruction types supported by the axis. For example, the encoder axis type does not support motion control instructions.

Different axis types will affect the input and output modes of the axis. The input and output signals of each axis type are shown in the table below. If the axis type is changed, the input and output signals of the corresponding terminals will change. Please confirm that they are correct before performing the corresponding operations.

Axis Type Designation	Axis Type Code	Connection Method with Controller	Input	Output	Remarks
Unused	-1	Axis not enabled	None	None	Motion control instructions are not allowed to be placed on this axis. The algorithm does not solve this type of axis, which can save algorithm solving time
Virtual	0	Built-in Controller	None	None	Generally used as the combined axis or

Axis Type Designation	Axis Type Code	Connection Method with Controller	Input	Output	Remarks
					intermediate solution axis of interpolation instructions
EtherCAT_ Position	50	EtherCAT bus connection	Get the encoder feedback position (axis position information) through the EtherCAT bus without position closed-loop control	Output the instruction position through the EtherCAT bus	-
EtherCAT_ Speed	51	EtherCAT bus connection	Get the encoder feedback position (axis position information) through the EtherCAT bus, and perform position closed-loop control operation when servoLoop = 1	Output the instruction speed through the EtherCAT bus	-
EtherCAT_ Torque	52	EtherCAT bus connection	Get the encoder feedback position (axis position information) through the EtherCAT bus, and perform position closed-loop control operation when servoLoop = 1	Output the instruction torque through the EtherCAT bus	-
EtherCAT_ Encoder	53	EtherCAT bus connection	Get the encoder feedback position through the ECI module extended with EtherCAT bus		

5.1.1.2.3 Example

Because AxisType is an enumeration type, it can be used as follows.

```
Result := HMC_SetAxisType(1, 51); (*The axis type enumeration value of  EtherCAT_Spee is 51*)

IF 51= Axis1.AxisType THEN

Axis1.Speed := 4000;
```

END_IF

5.1.1.3 AxisState (axis PLCopen status)

5.1.1.3.1 Basic properties

Type	MC_AXIS_STATE
Read/Write property	Read only
Set function	-
Initial Value	0
Value range	mcDisabled(0): Disabled mcErrorstop(1): Fault shutdown mcStopping(2): Stopped mcStandstill(3): Enabled mcDiscreteMotion(4): Discrete motion mcContinuousMotion(5): Continuous motion mcSynchronizedMotion(6): Synchronized motion mcHoming(7): Origin regression
Related chapters	PLCopen state machine

5.1.1.3.2 Meaning description

AxisState indicates the current axis state, which complies with the PLCopen standard single-axis motion control state machine. The axis state will change as the controller instruction changes. For details, refer to the PLCopen state machine section.

5.1.1.4 Units (Unit conversion factor)

5.1.1.4.1 Basic properties

Type	**USINT**
Read/Write Property	Read only
Set Function	HMC_SetUnits
Initial Value	1
Value Range	Positive floating point number
Related Chapters	HMC_SetUnits (Setting User Units)

5.1.1.4.2 Meaning description

It is a unit conversion factor (and can only be a positive value) in line/user unit.

It is Units, an axis parameter to set the unit conversion factor of the specified axis. This parameter can only be positive values in the line/user unit. "Line" refers to the scale unit corresponding to one pulse of the axis.

By setting the axis parameter Units, the count value or pulse output value fed back by the encoder can be converted into a user-friendly unit value, such as mm, angle, etc.

The default value of Units in the system is 1, which means that the user unit is "line", so the units of motion parameters such as Speed, Accel, Decel, and FastDecel are all "line/second", or "line/(second*second)". When Units

are modified, the units of these motion parameters naturally change to "user unit/second", or "user unit/(second*second)", but the values of these motion parameters need to be modified manually by the user according to the project needs.

For how to set this parameter, see HMC_SetUnits (Setting User Units).

5.1.2 Status Parameters (Status)

5.1.2.1 MCmdType (Current instruction type)

5.1.2.1.1 Basic properties

Type	EmCmdType
Read/Write Property	Read only
Set Function	-
Initial Value	HMC_McIdle (null)
Value Range	All motion control instruction types of the system

5.1.2.1.2 Meaning description

Current instruction type of the axis. Each axis has a 64-level motion instruction cache (instruction queue/FIFO). The instruction at the head of the instruction queue is that currently being executed.

All motion control instruction types supported by the system

Value	Name	Description	Remarks
0	HMC__McIdle	Idle	No motion instruction
1	HMC_ERR1	Reserved 1	System reserved
2	HMC_ERR2	Reserved 2	System reserved
3	HMC_ERR3	Reserved 3	System reserved
4	HMC_ERR4	Reserved 4	System reserved
5	HMC_ERR5	Reserved 5	System reserved
6	HMC_Home	Origin regression	-
7	HMC_HomeB	Origin regression (blocking)	Not support
8	HMC_HomeEx	Advanced origin regression	Not support
9	HMC_HomeExB	Advanced origin regression (blocking)	Not support
10	HMC_Move	Relative motion	-
11	HMC_MoveB	Relative motion (blocking)	Not support

Value	Name	Description	Remarks
12	HMC_MoveEx	Advanced relative motion	-
13	HMC_MoveExB	Advanced relative motion (blocking)	Not support
14	HMC__MoveAbs	Absolute motion	-
15	HMC__MoveAbsB	Absolute motion (blocking)	Not support
16	HMC__MoveAbsEx	Advanced absolute motion	-
17	HMC__MoveAbsExB	Advanced absolute motion (blocking)	Not support
18	HMC__Forward	Forward motion	-
19	HMC__Forward Ex	Advanced forward motion	-
20	HMC_Reverse	Reverse motion	-
21	HMC_ReverseEx	Advanced reverse motion	-
22	HMC_Move2D	Two-axis linear interpolation	-
23	HMC_Move2DB	Two-axis linear interpolation (blocking)	Not support
24	HMC_Move2DEx	Advanced two-axis linear interpolation	-
25	HMC_Move2DExB	Advanced two-axis linear interpolation (blocking)	Not support
26	HMC_Move3D	Three-axis linear interpolation	-
27	HMC_Move3DB	Three-axis linear interpolation (blocking)	Not support
28	HMC_Move3DEx	Advanced three-axis linear interpolation	-
29	HMC_Move3DExB	Advanced three-axis linear interpolation (blocking)	Not support
30	HMC_MoveND	N-axis linear interpolation	-
31	HMC_MoveNDB	N-axis linear interpolation (blocking)	Not support
32	HMC_MoveNDEx	Advanced N-axis linear interpolation	-
33	HMC_MoveNDExB	Advanced N-axis linear interpolation (blocking)	Not support
34	HMC_MoveCircle	Arc interpolation	-
35	HMC_MoveCircleB	Arc interpolation (blocking)	Not support
36	HMC_MoveCircleEx	Advanced arc interpolation	-

Value	Name	Description	Remarks
37	HMC_MoveCircleExB	Advanced arc interpolation (blocking)	Not support
38	HMC_MoveHelical	Helical interpolation	-
39	HMC_MoveHelicalB	Helical interpolation (blocking)	Not support
40	HMC_MoveHelicalEx	Advanced helical interpolation	-
41	HMC_MoveHelicalExB	Advanced helical interpolation (blocking)	Not support
42	HMC__MoveSpherical	Spherical arc interpolation	-
43	HMC__MoveSphericalB	Spherical arc interpolation (blocking)	Not support
44	HMC__MoveSphericalEx	Advanced spherical arc interpolation	-
45	HMC__MoveSphericalExB	Advanced spherical arc interpolation (blocking)	Not support
46	HMC_Gear	Electronic gear	-
47	HMC__Superimpose	Superimposing	-
48	HMC__SuperimposeEx	Advanced superimposition	-
49	HMC_Cam	Electronic cam	-
50	HMC__MoveAbs2D	Two-axis absolute linear interpolation	-
51	HMC__MoveAbs2DEx	Advanced two-axis absolute linear interpolation	-
52	HMC__MoveAbs3D	Three-axis absolute linear interpolation	-
53	HMC__MoveAbs3DEx	Advanced three-axis absolute linear interpolation	-
54	HMC__MoveAbsND	N-axis absolute linear interpolation	-
55	HMC__MoveAbsNDEx	Advanced N-axis absolute linear interpolation	-
56	HMC__MoveTang	Tangential tracking	-
57	HMC__MoveSplineND	Spline interpolation	-
59	HMC__MoveSplineNDEx	Advanced spline interpolation	-
62	HMC_MoveLink	Follow-up shearing motion	-
63	HMC__StewartControl	Stewart mechanism control	

Value	Name	Description	Remarks
64	HMC__SplineCam	Spline cam	
65	HMC__Scara5Control	5-DOF SCARA robot arm control	
66	HMC__MoveFlexLink	Flying shear/rotary cutting motion	
67	HMC__GantrySynchro	Gantry synchronization control	
68	HMC__GantryInit	Alignment initialization of gantry synchronized back to reference point position	
69	HMC__Serial4Control	4-DOF serial robot arm control	
71	SMC__SyncMoveVelocity	Cycle synchronous speed control	
72	MC__TorqueControl	Torque control	

5.1.2.2 NCmdType (Next instruction type)

5.1.2.2.1 Basic properties

Type	EmCmdType
Read/Write Property	Read only
Set Function	-
Initial Value	HMC_McIdle (null)
Value Range	All motion control instruction types of the system

5.1.2.2.2 Meaning description

Next instruction type of the axis. Each axis has a 64-level motion instruction cache (instruction queue/FIFO). It is located next to the instruction at the head of the instruction queue.

5.1.2.3 AxisStatus (Axis status)

5.1.2.3.1 Basic properties

Type	UDINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	0 means normal; Other values are converted to binary numbers and then analyzed bitwise. See the Axis Status Analysis Table for the meaning of each bit.

Type	UDINT
Related Chapters	AxisErrMask (Axis Status Mask)

5.1.2.3.2 Meaning description

It indicates the current running status. When it is 0, it means there is no axis exception. When it is not 0, the bitwise analysis is as follows:

Axis Status Analysis Table

Bit	Meaning	Elimination Method	Possible Causes
0	FE exceeds FELimit for 10 Consecutive Times	The status is locked and needs to be cleared by the user	1. The axis encoder line is not connected; 2. The servo is not enabled due to motor driver wiring error; 3. The servo is disabled because the motor driver reports an error; 4. The axis parameter FELimit is set too small; 5. The axis parameter PID is unreasonable.
1	Warning of FE Exceeding Limit Once	This warning will be automatically lifted after 1,000 consecutive times within the limit	1. The axis parameter FELimit is set out of the critical value; 2. The axis parameter PID is unreasonable.
2	Axis Parameter Setting Error	The status is locked and needs to be cleared by the user	The axis parameters are set to invalid values (for example, Speed and CreepSpeed are negative; Accel and Decel are 0 or negative).
3	Reaching Forward or Reverse Limit, Including Soft Limit and Hard Limit	When the limit condition is not met, it will be automatically cleared	The occurrence of a finitely formed axis satisfies one of the following conditions: 1. Axis Dpos≥FSLimit (forward soft limit); 2. Axis Dpos≤RSLimit (reverse soft limit); 3. The associated DI of axis FHLimitIn is 1 (forward hard limit); 4. The associated DI of axis RHLimitIn is 1 (reverse hard limit);
4	Communication Error with the Bus Axis	Supported by LX controllers	1. The bus is disconnected or the input/output connections are incorrect; 2. The number of configured bus slave stations is inconsistent with the actual number of connected bus slave stations; 3. The bus master station configuration parameters are different from the slave station parameters; 4. The configured slave station model is different from the actual slave station model; 5. The bus slave station reports an error (see the slave station manual for error information);

Bit	Meaning	Elimination Method	Possible Causes
			(The global variable RtexDiagGroup in AT or EtherCATDiagGroup contains detailed diagnostic information of the bus part)
5	Gantry Axis Deviation over-Limit Alarm	It will be automatically cleared after the deviation is eliminated	The gantry axis has an inter-axis deviation that exceeds the alarm threshold
6	Gantry Axis Deviation over-Limit Stop	Manual elimination	The gantry axis has an inter-axis deviation that exceeds the emergency stop threshold
7~31	Unused	Reserved	Reserved

Note: When an axis status error occurs, after eliminating the cause of the error, you can use HMC_ClearAxisErr to clear the axis status error, or use HMC_ClearAllErr to clear all axis status errors. If there is a communication error with the bus axis, you need to use EtherCAT_BusReset() to reset the corresponding bus protocol stack before clearing the error.

5.1.2.3.3 Example

When axis 0 reaches the forward or reverse limit, turn off the servo enable switch. The procedure is as follows:

```
IF (UDINT_TO_DWORD (Axis0.AxisStatus) AND 8) >0 THEN (*Axis 0 AxisStatus and 8-bit AND, determine whether the forward or reverse limit is reached*)
HMC_DriveEnable(0); (*Turn off the servo enable switch*)
END_IF
```

5.1.2.4 AxisErrMask (Axis Status Mask)

5.1.2.4. 1. Basic properties

Type	UDINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	1
Value Range	0 means no exception handling will be performed for various AxisStatus values. For other value setting methods, please refer to the meaning of each bit of AxisStatus, and you can set one or more bits to 1.
Related Chapters	AxisStatus (Axis status)

5.1.2.4.2 Meaning description

It is the axis status exception mask, with 32 bits corresponding to the 32 bits of AxisStatus. When a certain bit of AxisErrMask is 1 and the corresponding bit of AxisStatus is also 1, it is considered that a system exception has occurred. At this time, the system processing method is: Disable the DriveEnable signal and set ServoLoop to 0.

5.1.2.4.3 Example

When AxisStatus Bit0 and Bit4 of axis 0 are 1, turn off the servo enable switch and set the closed-loop output switch ServoLoop to 0 (open loop). The procedure is as follows:

Axis0.AxisErrMask= 17; (*Set the AxisErrMask parameters Bit0 and Bit4 of axis 0 to 1, then when the AxisStatus Bit0 and Bit4 of axis 0 are 1, turn off the servo enable switch and set the closed-loop output switch ServoLoop to 0 (open loop)*)

5.1.2.5 LimitState (over-limit status)

5.1.2.5.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	If LimitState is 0, it means it is normal, that is, it is within the limit. If it is not 0, first convert it into a binary number. When each bit is 1, the meaning is as follows: Bit 0: The forward hard limit is exceeded Bit 1: The reverse hard limit is exceeded Bit 2: The forward soft limit is exceeded Bit 3: The reverse soft limit is exceeded Bit 4: Soft-limit-related parameter error (forward soft limit ≤ reverse soft limit) Bit 5: Status conflict error (both the forward soft/hard limit and the reverse soft/hard limit are exceeded)
Related Chapters	RepMode(Position stroke mode) FsLimitMode/RsLimitMode (Instruction Cancellation method during forward/reverse limit) FHLimitIn/RHLimitIn (Forward/Reverse hard limit channel number) FSLimit/RSLimit (Forward/Reverse soft limit boundary value)

5.1.2.5.2 Meaning description

When the axis has a limited stroke (RepMode=0), this axis parameter is used to describe the current over-limit status.

5.1.2.5.3 Value

Bit	Description	Value	Remarks
0	Forward hard limit reached	1	The forward hard limit associated DI channel is 1
1	Reverse hard limit reached	2	The reverse hard limit associated DI channel is 1
2	Forward soft limit reached	4	Dpos≥FSLimit
3	Reverse soft limit reached	8	Dpos≤RSLimit
4	Soft-limit-related parameter error	16	Forward soft limit ≤ reverse soft limit
5	Forward and reverse limits reached at the same time	32	Including soft limit and hard limit
6~7	Unused	-	Reserved

5.1.2.5.4 Example

When axis 0 reaches the forward soft limit, the reverse displacement is 100.

The procedure is as follows:

IF (USINT_TO_WORD (Axis0.LimitState) AND 4) >0 THEN(*Axis 0 LimitState and 4-bit AND, determine whether the forward soft limit is reached*)

HMC_Move(0,- 100); (*Axis 0 reverse displacement 100*)

END_IF

Remarks

(1) When the axis reaches the forward limit (including the soft limit and hard limit), the instruction to send the aggravated forward limit will be canceled immediately, and the axis will not move forward. At this time, the instruction to slow down the forward limit can be successfully sent, and the axis can move in the direction of slowing down the forward limit;

(2) When the axis reaches the reverse limit (including the soft limit and hard limit), the instruction to send the aggravated reverse limit will be canceled immediately, and the axis will not move in the reverse direction. At this time, the instruction to slow down the reverse limit can be successfully sent, and the axis can move in the direction of slowing down the reverse limit;

(3) If the axis that is executing a single-axis instruction exceeds the limit, the instruction will be canceled and Max(FastDecel,Decel) will be used to quickly stop the motion of the current axis;

(4) If the combined axis that is executing an interpolation instruction exceeds the limit, the entire interpolation instruction will be canceled and Max(FastDecel,Decel) of the combined axis will be used to quickly stop the motion of the current axis;

(5) If the split axis that is executing the interpolation instruction exceeds the limit, the entire interpolation instruction will be canceled, and the over-limit split axis will use Max (FastDecel split axis, Decel split axis,

combined axis decomposed deceleration) to quickly stop the movement of the current split axis. The combined axis and the split axis that has not exceeded the limit are not affected and continue to move;

(6) If the slave axis that is executing a linkage instruction exceeds the limit, the instruction will be canceled and Max(FastDecel,Decel) will be used to quickly stop the motion of the current slave axis;

(7) When the axis reaches the forward and reverse limits (including soft limit and hard limit) at the same time, the axis instruction stops solving but is not canceled. When it returns to a limit, it will be handled according to the above situations.

5.1.3 Position Related Parameters (Positions)

5.1.3.1 MPos (Feedback position)

5.1.3.1.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	Floating point range
Related Chapters	DPos(Target position of this cycle) RepMode(Position stroke mode) RepDist(Cycle stroke length) HMC_DefPos (Setting current position)

5.1.3.1.2 Meaning description

It is the current measured feedback position in user units. For axis types with actual input signals, MPos reflects the current actual position .

(1) Mpos value principle

When the axis type has actual input signals, such as EtherCAT bus axis, its Mpos comes from the real-time measured value of the encoder, but if FeedBackSrcMode=1, Mpos comes from the user-defined input source.

When the axis type is a virtual axis, its MPos is equal to DPos.

(2) Mpos calculation method

Mpos adopts the incremental calculation method. By accumulating the increment of the encoder feedback value in the two cycles before and after the MPos value of the previous cycle, the MPos value of this cycle can be obtained.

5.1.3.2 DPos (Target position of this cycle)

5.1.3.2.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	Floating point range
Related Chapters	RepMode(Position stroke mode) RepDist(Cycle stroke length)

5.1.3.2.2 Meaning description

It is the target position of this cycle calculated in real time by the algorithm, based on which the final output signal, in user units, is obtained.

The system processes the axis parameter according to the following principles:

(1) The axis parameter is updated in real time during system operation, and the value of this parameter is calculated in real time by the motion instruction;

(2) In cycle stroke mode, MPos is always in [0,RepDist) or [-RepDist,RepDist), and the actual distance is maintained between Dpos and Mpos, so that Dpos is also in [0,RepDist) or [-RepDist, RepDist) in most cases, but in some individual cases, for example, when the RepDist value is set to a relatively small value, the Dpos of the bus axis will be outside the cycle stroke range;

(3) For EtherCAT_Speed axes (axis type 51), when ServoLoop is 0, Dpos is equal to Mpos in real time.

5.1.3.3 FE (Deviation)

5.1.3.3.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	Floating point range
Related Chapters	StopFETol (Whether the feedback position reaches the dead band of the target position) FELimits(Follow-up error limit)

5.1.3.3.2 Meaning description

It is the real-time follow-up error, FE=DPOS-MPOS.

In servo mode, PID is calculated based on FE to obtain the final output value. In bus position mode, FE does not participate in the PID calculation but is used for deviation alarm. When the axis type is non-bus axis, this value

is always 0.

5.1.3.4 StopFETol (Whether the feedback position reaches the dead band of the target position)

5.1.3.4.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	10
Value Range	Positive floating point number
Related Chapters	FE (Deviation)

5.1.3.4.2 Meaning description

It is the FE deviation tolerance range, which is used to determine whether MPos reaches the target position in speed mode.

When the axis type is speed mode, including bus speed axis, if CmdStopMode=1, to determine whether the instruction is over, in addition to meeting the condition that the DPos solution is completed and it reaches End Pos, it also needs to meet Abs(FE)<StopFETol, that is, it is considered that Mpos reaches the instruction target position. If CmdStopMode=0, it only needs to meet the condition that DPos calculation is completed and reaches End Pos, and thus it can be considered that the instruction is over.

5.1.3.5 RepMode (Position stroke mode)

5.1.3.5.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	2
Value Range	0: Limited stroke, in which case RepDist is meaningless, FsLimit and RsLimit are soft limits, and the hard limits are configured through the function HMC_HwLimitConfig 1: Cycle stroke, in which case both FsLimit/RsLimit and FHLimitIn/RHLimitIn are meaningless,and [0, RepDist) is the upper and lower limits of the cycle stroke 2: Cycle stroke, in which case both FsLimit/RsLimit and FHLimitln/RHLimitIn are meaningless, and [-RepDist, RepDist) is the upper and lower limits of the cycle stroke
Related Chapters	RepDist(Cycle stroke length) MPos (Feedback position)

5.1.3.5.2 Meaning description

It is used to set the position stroke mode of this axis: cycle stroke/limited stroke.

(1) Definition of limited stroke

It means that the axis can only move within a limited range and cannot exceed this range. For example, a moving object driven by ball screws can only move within a limited stroke range.

When the user sets RepMode=0, it means that the axis has a limited stroke, and in this case, RepDist is meaningless and the axis has a stroke limit.

FsLimit and RsLimit represent soft limits, and the hard limit channels (FHLimitIn/RHLimitIn) are configured through the function HMC_HwLimitConfig. When the axis with a limited stroke exceeds the hard limit or soft limit, the system's processing method is detailed in the section "Limited-stroke Axis Over-limit Processing".

(2) Definition of cycle stroke

The axis is constantly rotating and its position is constantly repeated without limit, such as the motion of a clock.

When the user sets RepMode=1 or RepMode=2, it means that the axis has a cycle stroke. Among them, when RepMode=1, [0,RepDist) is the upper and lower limits of the cycle stroke; when RepMode=2, [-RepDist,RepDist) is the upper and lower limits of the cycle stroke.

For cycle stroke, both FsLimit/RsLimit and FHLimitIn/RHLimitIn are meaningless.

(3) Automatic modification of stroke mode after axis type settings

When the user sets an axis type through the function HMC_SetAxisType, the system will automatically modify the stroke mode of the axis. When it is set to a virtual axis or bus encoder axis, the system will set RepMode=2; when the user sets it to other axis types, the system will set RepMode=0.

5.1.3.6 RepDist (Cycle stroke length)

5.1.3.6.1 Basic properties

Type	**LREAL**
Read/Write Property	Read/Write
Set Function	-
Initial Value	10,000,000,000
Value Range	Positive floating point number
Related Chapters	RepMode(Position stroke mode)

5.1.3.6.2 Meaning description

It is valid in cycle stroke mode (RepMode is 1 or 2): value range limits displayed by End Pos and MPOS. In the case of a bus axis, DPos may exceed the cycle stroke for a short period of time.

When the user sets RepMode=1, it means that the axis has a cycle stroke, and [0,RepDist] is the upper and lower limits of the cycle stroke. When the user sets RepMode=2, it means that the axis has a cycle stroke, and [-

RepDist,RepDist] is the upper and lower limits of the cycle stroke.

5.1.3.7 Remain (Remaining positions)

5.1.3.7.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	Floating point range
Related Chapters	-

5.1.3.7.2 Meaning description

It means the remaining displacement from the instruction target position, in user units, during the execution of the current motion instruction. The axis parameter is updated in real time during system operation, Remain=End Pos-DPos.

5.1.4 Speed Related Parameters (Velocity Profile)

5.1.4.1 Speed (Target Speed)

5.1.4.1.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	2000
Value Range	Positive floating point number
Related Chapters	-

5.1.4.1.2 Meaning description

It is used to set the target instruction solving speed in user units/second. This parameter can be modified in real time. It can only be a positive number or 0. If it is set to a negative value, the system automatically takes its absolute value and reports a warning in Bit 2 of AxisStatus.

When the axis is executing a motion instruction, Speed can be modified in real time, with the system processing method as follows:

(1) When the axis is executing a single-axis instruction, Speed can be modified in real time and takes effect in real time, and then the instruction will perform motion calculation according to the new Speed;

(2) When the axis is executing an interpolation instruction and it is an interpolating combined axis, Speed can be modified in real time and take effect in real time, and then the combined axis will perform motion calculation according to the new Speed; because the interpolation relationship between the combined axis and the split axis remains unchanged, these real-time calculated motion quantities will naturally be broken down into the split axis;

(3) When the axis is executing an interpolation instruction and it is an interpolating split axis, Speed can be modified in real time, but it will not affect the motion of the interpolating split axis;

(4) When the axis is executing a linkage instruction and it is a linkage slave axis, Speed can be modified in real time, but it will not affect the motion of the axis.

5.1.4.2 Accel (Acceleration)

5.1.4.2.1 Basic properties

Type	**LREAL**
Read/Write Property	Read/Write
Set Function	-
Initial Value	200,000
Value Range	Positive floating point number
Related Chapters	-

5.1.4.2.2 Meaning description

It means the acceleration, in user units/(second*second).

This value can only be a positive number. If it is set to a negative value, the system automatically takes its absolute value; if it is 0, the parameter returns to the system default value, and; for invalid values, a warning is reported in Bit 2 of AxisStatus.

When an axis is executing a motion instruction, Accel can be modified in real time, with the system processing method as follows:

(1) When the axis is executing a single-axis instruction, Accel can be modified in real time and takes effect in real time, and then the instruction will perform motion calculation according to the new Accel;

(2) When the axis is executing an interpolation instruction and it is an interpolating combined axis, Accel can be modified in real time and take effect in real time, and then the combined axis will perform motion calculation according to the new Accel; because the interpolation relationship between the combined axis and the split axis remains unchanged, these real-time calculated motion quantities will naturally be broken down into the split axis;

(3) When the axis is executing an interpolation instruction and it is an interpolating split axis, Accel can be modified in real time, but it will not affect the motion of the interpolating split axis;

(4) When the axis is executing a linkage instruction and it is a linkage slave axis, Accel can be modified in real time, but it will not affect the motion of the axis.

5.1.4.3 Decel (Deceleration)

5.1.4.3.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	200,000
Value Range	Positive floating point number
Related Chapters	-

5.1.4.3.2 Meaning description

It means the deceleration, in user units/(second*second).

This value can only be a positive number. If it is set to a negative value, the system automatically takes its absolute value; if it is 0, the parameter returns to the system default value, and; for invalid values, a warning is reported in Bit 2 of AxisStatus.

When an axis is executing a motion instruction, Decel can be modified in real time, with the system processing method as follows:

(1) When the axis is executing a single-axis instruction, Decel can be modified in real time and takes effect in real time, and then the instruction will perform motion calculation according to the new Decel;

(2) When the axis is executing an interpolation instruction and it is an interpolating combined axis, Decel can be modified in real time and take effect in real time, and then the combined axis will perform motion calculation according to the new Accel; because the interpolation relationship between the combined axis and the split axis remains unchanged, these real-time calculated motion quantities will naturally be broken down into the split axis;

(3) When the axis is executing an interpolation instruction and it is an interpolating split axis, Decel can be modified in real time, but it will not affect the motion of the interpolating split axis;

(4) When the axis is executing a linkage instruction and it is a linkage slave axis, Decel can be modified in real time, but it will not affect the motion of the axis.

5.1.4.4 MpSpeed (Feedback Speed)

5.1.4.4.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0

Type	LREAL
Value Range	Floating point number
Related Chapters	-

5.1.4.4.2 Meaning description

It means the feedback speed measured in real time in a unit is the same as Speed, with the speed measurement cycle being the system servo cycle.

The calculation principle of MpSpeed is:

(1) For EtherCAT_Position (axis type 50), EtherCAT_Speed (axis type 51), EtherCAT_Torque (axis type 52), and EtherCAT_Encoder (axis type 53), MpSpeed is calculated based on the real-time measured position value from the encoder, that is, this reflects the changing speed of MPos;

(2) For other axes, this value is numerically equal to VpSpeed;

(3) The positive and negative values of MpSpeed represent the actual motion direction of the current axis.

5.1.4.5 VpSpeed (Solving Speed)

5.1.4.5.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	Positive floating point number
Related Chapters	Direction(Direction of solving speed)

5.1.4.5.2 Meaning description

It means the speed value in the same unit as the Speed calculated in real time by the motion instruction, with the calculation cycle being the system servo cycle. It reflects the changing speed value of Dpos. This parameter only represents the speed value, not including the direction, and so it is a positive value.

5.1.4.6 Direction (Direction of Solving Speed)

5.1.4.6.1 Basic properties

Type	SINT
Read/Write Property	Read only
Set Function	-
Initial Value	1

Type	SINT
Value Range	1: It indicates that it is moving in the forward direction -1: It indicates that it is moving in the reverse direction
Related Chapters	VpSpeed (Solving speed)

5.1.4.6.2 Meaning description

It means the speed direction is calculated in real time by the motion instruction, which reflects the speed change direction of DPos.

5.1.4.7 Merge (Merging Function Mode)

5.1.4.7.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	HMC_SetMerge
Initial Value	0
Value Range	This value can only be 0, 1, 2, 3, or 4 0: Merge off, in which case each instruction block completes its own motion independently 1: Merge on, in which case the target speed at the end of the current instruction block is the Speed of the instruction block 2: Merge on, in which case the target speed at the end of the current instruction block is the Speed of the next instruction block 3: Merge on, in which case the target speed at the end of the current instruction block is the minimum value of the Speed of the instruction block and the Speed of the next instruction block. 4: Merge on, in which case the target speed at the end of the current instruction block is the maximum value of the Speed of the instruction block and the Speed of the next instruction block
Related Chapters	Motion look-ahead

5.1.4.7.2 Meaning description

It is used to set the speed merging function to determine whether to enable the speed planning strategy of look-ahead pre-processing. Merge (a merging function) connects a series of motion instructions in the axis motion control instruction cache to make the load motion speed continuous, saying goodbye to stop-and-go, which can significantly improve efficiency.

5.1.4.8 CreepSpeed (Origin Regression Creep Speed)

5.1.4.8.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	200
Value Range	Positive floating point number
Related Chapters	HMC_Home (Origin regression)

5.1.4.8.2 Meaning description

It means the creep speed, in user units/second.

It is used in the stage of origin regression to find the origin. In this stage, the system uses CreepSpeed as the target speed for motion calculation, and the user can modify CreepSpeed, which will take effect in real time.

This value can only be a positive number or 0. If it is set to a negative value, its absolute value is taken, and a warning is reported in Bit 2 of AxisStatus.

5.1.4.9 FastDecel (Fast Deceleration)

5.1.4.9.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	5,000,000
Value Range	Positive floating point number

5.1.4.9.2 Meaning description

It means the fast deceleration, in user units/(second*second), which is used for automatic emergency stop after soft or hard over-limit of a limited stroke axis. For details, please refer to the section "Limited stroke Axis Over-limit Processing".

This value can only be a positive number (If it is a negative value, its absolute value is taken; if it is 0, the parameter returns to the system default value, and; for invalid values, a warning is reported in Bit 2 of AxisStatus).

5.1.4.10 Jerk (Jerk)

5.1.4.10.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-

Type	LREAL
Initial Value	5,000,000,000
Value Range	Positive floating point number
Related Chapters	HMC_ClcJerk (Shock operation)

5.1.4.10.2 Meaning description

It means the jerk (a derivative of acceleration or deceleration), in user units/(second*second*second).

This value can only be positive. If it is a negative value, its absolute value is taken; if it is 0, the parameter returns to the system default value, and; for invalid values, a warning is reported in Bit 2 of AxisStatus.

5.1.4.11 JerkMode (Acceleration mode)

5.1.4.11.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	0: S-shaped motion disabled 1: S-shaped motion enabled
Related Chapters	-

5.1.4.11.2 Meaning description

It is used to enable/disable S-shaped motion (to enable or disable Jerk).

When Jerk is disabled, acceleration or deceleration is performed according to Accel and Decel, and the speed curve appears as a trapezoid; when Jerk is enabled, the acceleration (deceleration) and speed are changed according to Jerk as the acceleration jerk. Generally, the speed curve shows an "S" shape.

5.1.4.12 CmdStopMode (Instruction End Judgment Method)

5.1.4.12.1 Basic properties

Type	USINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	0 1
Related Chapters	-

5.1.4.12.2 Meaning description

The mode to determine whether the bus axis motion instruction has been executed.

(1) CmdStopMode=0

The criterion for the system to judge the end of the current instruction operation is: DPOS has reached the target position End Pos. That is, when this condition is met, the current instruction operation is considered to be completed, and the next instruction begins to be executed immediately.

(2) CmdStopMode= 1

The system determines that the following two conditions must be met at the end of the current instruction operation before executing the next instruction.

(a) DPOS has reached its target position

(b) It has been long enough for MPOS or DPOS to reach the target position (default 500 ms, which can also be modified through HMC_SetCmdStopMaxTime). When MPos reaches the target position, it means Abs(FE)<StopFETol. For example, if the system control cycle is 1 ms, 500 ms after DPOS reaches the target position, the system will consider that the instruction is completed even if MPOS does not reach the specified position.

5.1.4.13 MoveAbsMode (Direction Selection Method for Absolute Motion of Cycle Stroke Axis)

5.1.4.13.1 Basic properties

Type	**USINT**
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	0, 1, 2, 3, 4
Related Chapters	-

5.1.4.13.2 Meaning description

It is only valid for absolute motion instructions of the cycle stroke axis (RepMode=1 or RepMode=2), such as HMC_MoveAbs, HMC_MoveAbs2DEx, and HMC_MoveAbsND.

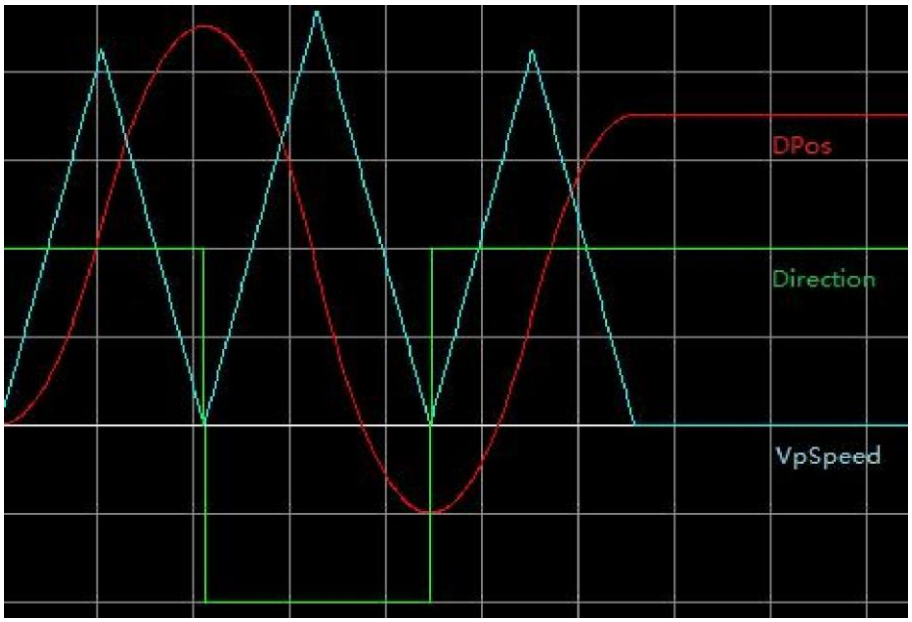
- MoveAbsMode=0: Confirm the direction within this cycle stroke
- MoveAbsMode= 1: The algorithm selects the direction with the shortest distance, and when the forward and reverse motions are of equal distances, select the forward direction
- MoveAbsMode=2: Move in the forward direction
- MoveAbsMode=3: Move in the reverse direction
- MoveAbsMode=4: Move in the current motion direction. That is, the Direction before MoveAbs is

the direction of this MoveAbs.

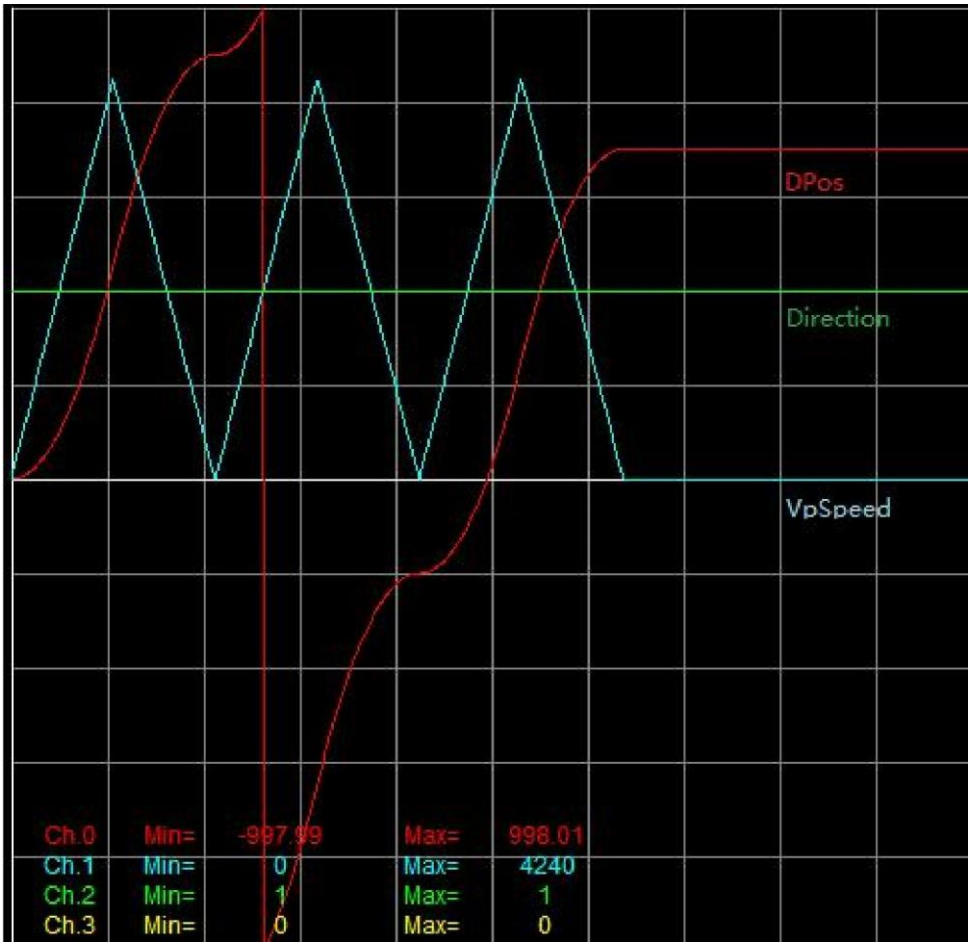
The following examples describe in detail the direction determination method for the above modes.

Write the following program, and then modify MoveAbsMode respectively to check the operation of axis 0.

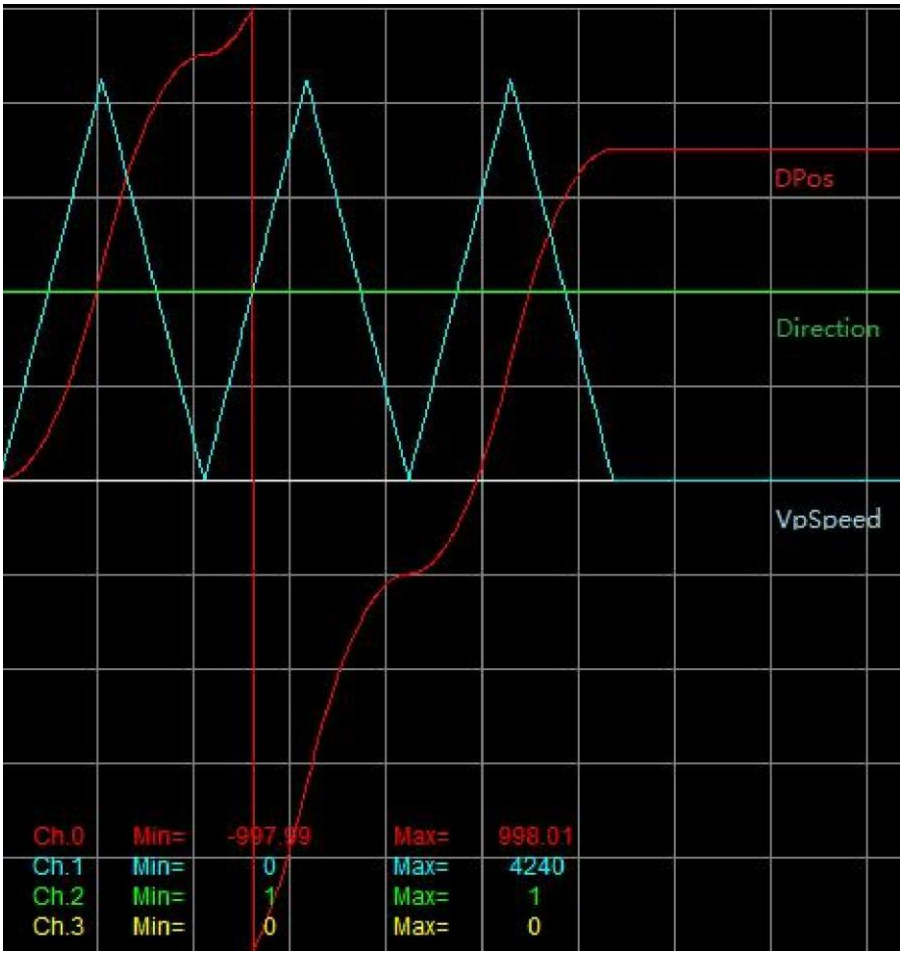
```
axis0.RepMode := 2;
axis0.RepDist := 1000;
hmc_defpos(0,0);
hmc_moveabs(0, 900);
hmc_moveabs(0, -200);
hmc_moveabs(0, 700);
```



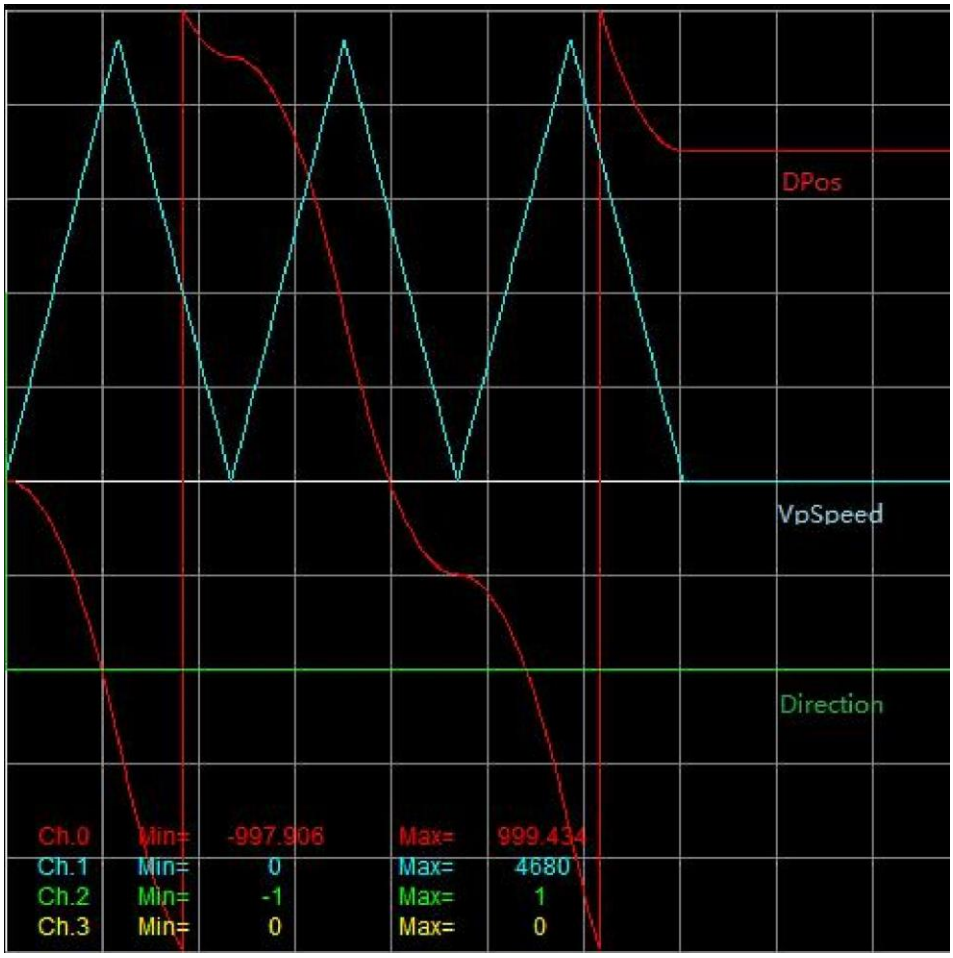
MoveAbsMode=0: Confirming direction within Cycle Stroke



For MoveAbsMode=1, Taking the Direction of the Shortest Distance



MoveAbsMode=2: Always Forward Direction



For MoveAbsMode=3, Always Reverse Direction

5.1.4.14 CancelMode (Instruction Stop Mode When Cancel)

5.1.4.14.1 Basic properties

Type	USINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	0, 1, 2
Related Chapters	CancelPos (Stop position when Cancel) HMC_Cancel (Canceling instruction)

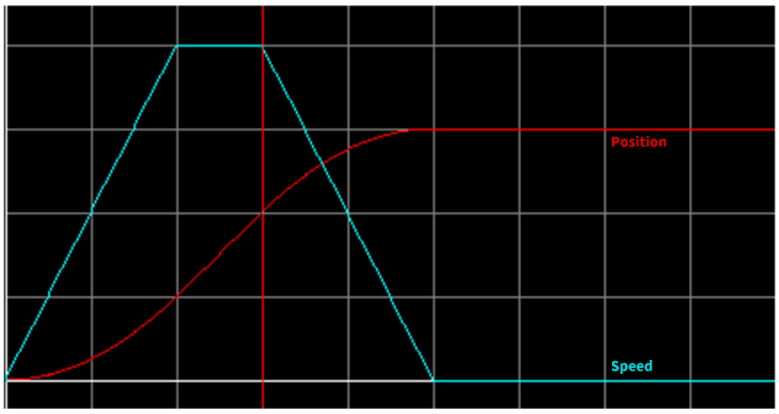
5.1.4.14.2 Meaning description

It indicates the instruction stop mode when Cancel.

During the instruction cancellation process, after receiving the Cancel instruction, the axis decelerates at the current speed until it stops. The specific stopping mode during this deceleration and stop process is specified by the axis parameter CancelMode. Details are as follows:

(1) For CancelMode=0, natural stop

It is the default value for the axis to stop naturally according to Decel. This mode supports all instruction types.

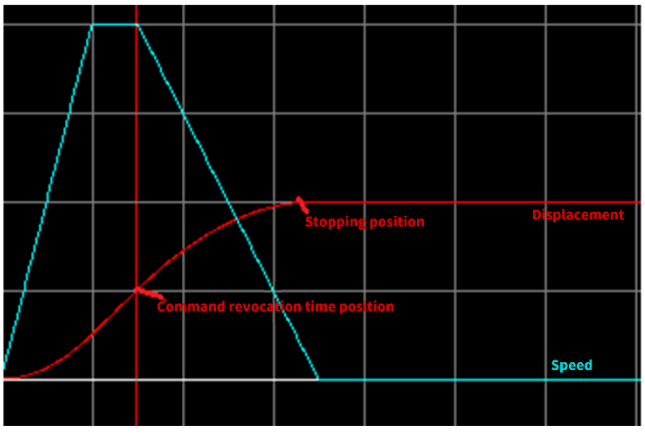


Instruction Stop Diagram

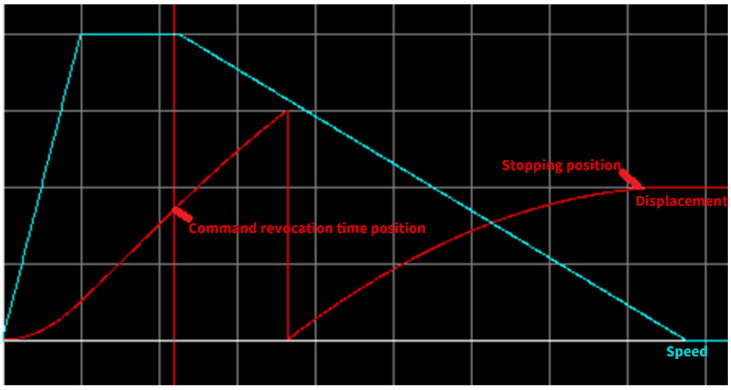
(2) For CancelMode=1, specifying the position to stop

The axis stops at the specified position at a speed not more than Decel, and the position is specified by the parameter CancelPos. This mode only takes effect for the cycle stroke mode (RepMode= 1 or 2) when the current instruction is one-way motion (HMC_Forward, HMC_Reverse). If it is a one-way motion instruction but is not in the cycle stroke mode, it is processed according to CancelMode=0.

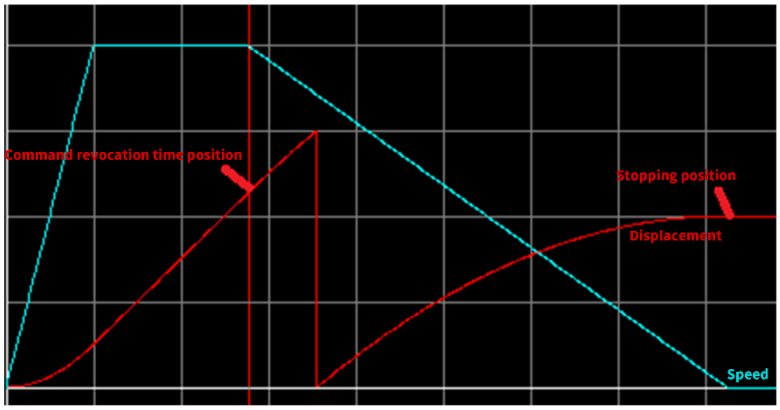
According to the relative relationship between the current position and CancelPos, there are three situations: 1. Current position < stop position, and the distance displacement difference between the two is enough for the current speed to reduce to 0 at deceleration; 2. Current position < stop position, but the distance displacement difference between the two is not enough for the current speed to reduce to 0 at deceleration, and; 3. Current position > stop position. The above three situations correspond to the following three figures respectively:



Instruction Stop Diagram



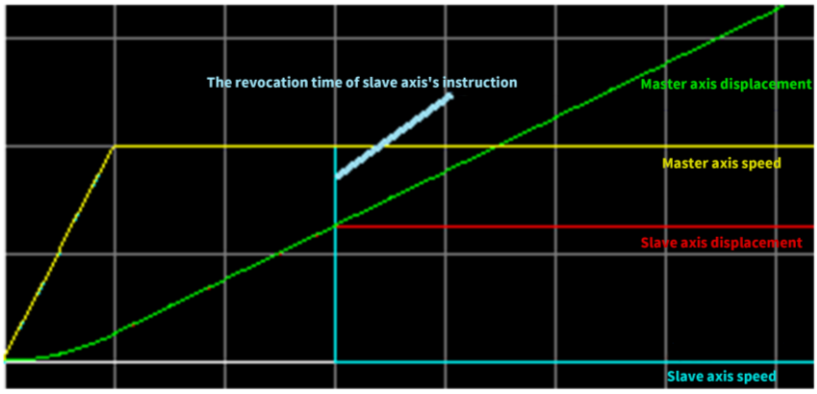
Instruction Stop Diagram



Instruction Stop Diagram

(3) For CancelMode=2, stopping immediately

The axis stops immediately and the speed is reduced directly from the current speed to 0. This mode is only effective for the axis that is executing the linkage instruction. If the status is not satisfied, it will be processed according to CancelMode=0 by default.



Instruction Stop Diagram

5.1.4.15 CancelPos (Stop Position When Cancel)

5.1.4.15.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	The value shall be between the upper and lower limits of the cycle stroke
Related Chapters	CancelMode (Instruction stop mode when Cancel)

5.1.4.15.2 Meaning description

When it is a cycle stroke axis with the instruction of HMC_Forward or HMC_Reverse, and CancelMode=1, this parameter is valid. This parameter indicates the position that the axis will reach when the Cancel instruction is completed. However, when CancelPos exceeds the stroke limit, it will be processed according to CancelMode=0.

5.1.5 Coefficient Factor Parameter (Gains)

5.1.5.1 Kp (PID Proportional Coefficient)

5.1.5.1.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	1
Value Range	Floating point number

5.1.5.1.2 Meaning description

When the control mode is the position closed-loop mode, it needs to go through the PID model operation, where Kp is the proportional coefficient of the PID model, which acts on the proportional term, and the proportional term acts on =Kp*Fe.

5.1.5.2 Ki (PID Integral Gain)

5.1.5.2.1 Basic properties

Type	LREAL
Read/Write property	Read/Write
Set Function	-
Initial Value	0
Value Range	Floating point number

5.1.5.2.2 Meaning description

When the control mode is the position closed-loop mode, it needs to go through the PID model operation, where Ki is the integral gain of the PID model, which acts on the integral term, and the integral term acts on =Ki*(∑(Fe*dt)).

5.1.5.3 Kd (Differential Gain of PID)

5.1.5.3.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Floating point number

5.1.5.3.2 Meaning description

When the control mode is the position closed-loop mode, it needs to go through the PID model operation, where Kd is the integral gain of the PID model, which acts on the differential term, and the differential term acts on =Kd*(d(Fe)/dt).

5.1.5.4 Kov (Measurement Speed Gain)

5.1.5.4.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Floating point number

5.1.5.4.2 Meaning description

If the control mode is the position closed-loop mode, the measured speed gain term will be superimposed on the PID calculation output. Measured speed gain action =Kov*MpSpeed.

5.1.5.5 Kvff (Speed Feedforward Gain)

5.1.5.5.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write

Type	LREAL
Set Function	-
Initial Value	0
Value Range	Floating point number

5.1.5.5.2 Meaning description

When the control mode is the position closed-loop mode, the superimposed speed feedforward gain term will be superimposed on the PID calculation output. Speed feedforward gain action = Kvff*((+1/- 1)*VpSpeed).

5.1.5.6 Kaff (Acceleration Feedforward Gain)

5.1.5.6.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Floating point number

5.1.5.6.2 Meaning description

It is the acceleration feedforward gain, with acceleration feedforward=Kaff*solved acceleration.

5.1.5.7 Ko (PID Output Amplification Factor)

5.1.5.7.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	1
Value Range	Floating point number

5.1.5.7.2 Meaning description

Servo output ratio, which is the amplification factor of the servo output value after PID calculation.

5.1.6 Limit Parameters (Limits)

5.1.6.1 FELimits (Follow-up Error Limit)

5.1.6.1.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	1.00E+100
Value Range	Positive double-precision floating point number (64 bits)
Related Chapters	-

5.1.6.1.2 Meaning description

It is used to set the normal range of FE (follow-up error). When Abs(FE)>FELimit, FE is considered to be over the limit. When FE exceeds the limit, the system will set bit0 in the axis parameter AxisStatus to 1 and turn off the servo enable. To shield the FE over-limit shutdown servo enable action, you can set bit0 of AxisErrMask to 0.

5.1.6.2 FsLimitMode/RsLimitMode (Instruction Cancellation Method during Rorward/reverse Limit)

5.1.6.2.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	0: Automatically cancel the current instruction after encountering the forward/reverse limit. 1: Manually cancel the current instruction after encountering the forward/reverse limit (not supported yet)
Related Chapters	-

5.1.6.2.2 Meaning description

It indicates the forward/reverse limit control mode, which is valid for both soft limit and hard limit. This parameter is only valid for axes with a limited stroke, but invalid for axes with a cycle stroke.

When a limited stroke axis exceeds the limit, an emergency stop action will be initiated. How the current instruction is processed is determined by the relevant parameters of the axis. For details, see "Limited Stroke Axis Over-Limit Processing" in the "Exception Handling" chapter.

0: Automatically cancel the current instruction after encountering the forward/reverse limit. It should be noted that for interpolation instructions, the entire interpolation instruction will be canceled no matter when the combined axis or the split axis encounters the limit.

1: Manually cancel the current instruction after encountering the forward/reverse limit. It should be noted that the current instruction will not be canceled before encountering the function HMC_Cancel. After the axis completes

the emergency stop, when the DPos value calculated by the original motion instruction aggravates the over-limit, the axis continues to stop, and when the DPos value calculated by the original motion instruction mitigates the over-limit, the axis will move according to the DPos value.

"aggravates the over-limit" here means: that when the forward soft/hard and soft limits are exceeded, the DPos value calculated by the current instruction becomes larger and larger, and when the reverse soft/hard and soft limits are exceeded, the DPos value calculated by the current instruction becomes smaller and smaller. Intuitively speaking, the over-limit is becoming more and more serious. "mitigate the over-limit" means exactly the opposite.

5.1.6.3 FHLimitIn/RHLimitIn (Forward /Reverse Hard Limit Channel Number)

5.1.6.3.1 Basic properties

Type	**UDINT**
Read/Write Property	Read only
Set Function	HMC_HwLimitConfig
Initial Value	-1
Value Range	Unused: 4294967295 Other values: DI channel number
Related Chapters	-

5.1.6.3.2 Meaning description

Set the DI channel number corresponding to the forward/reverse hard limit stroke switch.

It is meaningless during cycle stroke (RepMode = 1 or 2).

5.1.6.4 FSLimit/RSLimit (Forward/Reverse Soft Limit Boundary Value)

5.1.6.4.1 Basic properties

Type	**DINT**
Read/Write Property	Read/Write
Set Function	-
Initial Value	DACLimitLow: -1e100 DACLimitHigh: 1e100
Value Range	Double-precision floating-point number (64 bits). It needs to ensure RSLimit < FSLimit.
Related Chapters	-

5.1.6.4.2 Meaning description

It is used to set the forward/reverse soft limits of the limited stroke. It is meaningless during cycle stroke (RepMode = 1 or 2).

5.1.6.5 DACLimitHigh/DACLimitLow (DAC Output Upper/Lower Limit)

5.1.6.5.1 Basic properties

Type	DINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	DACLimitLow: -2147483648 DACLimitHigh: 2147483647
Value Range	Double-precision floating-point number (64 bits). It needs to ensure DACLimitHigh> DACLimitLow.
Related Chapters	-

5.1.6.5.2 Meaning description

It indicates the upper/lower limits of DACOut to ensure that the output of DACOut will not exceed the set range.

5.1.7 Origin and Hold Parameters (Home&Hold)

5.1.7.1 HomeState (Origin Regression Status)

5.1.7.1.1 Basic properties

Type	USINT
Read/Write property	Read only
Set function	-
Initial Value	0
Value range	The following description is related to the parameter HomeMode of the HMC_Home or HMC_Home instruction. 0: The search for the origin has not yet started 1: Quickly find the deceleration point state after the origin regression 2: Quick stop state after the deceleration point is found after the origin regression 3: Slowly find the home point state after the origin regression 4: Slowly find the Z-signal state after the origin regression 5: Slow stop state after the home point signal is found 6: Move to the home point signal position state 7: Find the limit switch quick stop state 8: When the limit switch is touched, quickly find the home point state in the reverse direction 9: Find the home DI at high speed and then decelerate to low speed 10: Stop state after the home point is found quickly in the reverse direction

	11: Origin regression completed 12: Origin regression instruction interrupted
Related chapters	HMC_Home (Origin regression) MC_Home (Origin Regression)

5.1.7.1.2 Meaning description

It indicates the origin regression state, which is used to indicate the state of HMC_Home and MC_Home during the origin regression process. The homing process goes through 0-12 states. The meaning of each state is shown in the superscript value range. The user can clearly understand the current origin regression situation by viewing this parameter.

5.1.7.2 HomeIn (DI Channel Used for Origin Regression)

5.1.7.2.1 Basic properties

Type	UDINT
Read/Write Property	Read only
Set Function	HMC_Home, HMC_HomeEx (Not Support)
Initial Value	-1
Value Range	4294967295 indicates that the origin regression function is not used, while other values indicate the DI channel number.
Related Chapters	HMC_Home (Origin regression)

5.1.7.2.2 Meaning description

It indicates the DI channel number used for origin regression. This parameter value is set when the HMC_Home and HMC_HomeEx (Not Support) instructions are issued.

5.1.7.3 HomeCapIn (High-speed Capture Channel Used for Origin Regression)

5.1.7.3.1 Basic properties

Type	INT
Read/Write Property	Read only
Set Function	HMC_HomeEx (Not Support)
Initial Value	-1
Value Range	Not Support

5.1.7.3.2 Meaning description

It is the high-speed capture channel number used by HMC_HomeEx, which is set when the HMC_HomeEx instruction is issued.

5.1.7.4 HoldSpeed (Hold Speed)

5.1.7.4.1 Basic properties

Type	LREAL
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Positive floating point number
Related Chapters	HMC_HoldConfig (Setting hold speed switch)

5.1.7.4.2 Meaning description

When the hold speed switch is triggered, the axis will accelerate or decelerate from the current speed to HoldSpeed. A common method is to set HoldSpeed=0. When the Hold trigger source (HoldIn) is 1, Speed is forced to 0, which can realize the "pause" function.

For a detailed description of the hold speed function, please refer to the section HMC_HoldConfig (Setting hold speed switch).

5.1.7.5 HoldIn (Hold Speed Switch DI Channel Number)

5.1.7.5.1 Basic properties

Type	UDINT
Read/Write Property	Read only
Set Function	HMC_HoldConfig
Initial Value	-1
Value Range	4294967295 indicates that the function is disabled, while other values indicate the DI channel number
Related Chapters	HMC_HoldConfig (Setting hold speed switch)

5.1.7.5.2 Meaning description

It is the trigger source DI channel number of the hold speed function.

5.1.8 Input Related Parameters (Input)

5.1.8.1 KeNumerator/KeDenominator (Numerator/Denominator of Encoder Input Ratio Ke)

5.1.8.1.1 Basic properties

Type	DINT
Read/Write Property	Read only
Set Function	HMC_SetKe

Type	DINT
Initial Value	1
Value Range	DINT integers: Neither the numerator nor the denominator is 0, but can have different signs
Related Chapters	HMC_SetKe (Setting Ke)

5.1.8.1.2 Meaning description

It indicates the numerator and denominator of Ke. The parameter Ke is only valid for axis types with actual position feedback (bus axes) and can be negative. This parameter has a similar function to Units. (Ke /Units) together constitute the "unit conversion factor" of the bus axis. Its unit is the "lines/user unit". Please refer to the position closed-loop diagram in the section "System Control Mode" for the action mode of Ke.

5.1.8.2 EncoderValue (Internal Encoder Value)

5.1.8.2.1 Basic properties

Type	DINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	-2147483648～+2147483647
Related Chapters	-

5.1.8.2.2 Meaning description

When the axis feeds back position information through the encoder, the axis parameter represents the internal encoder value, which is refreshed per servo cycle. The actual position value MPos is calculated incrementally based on this value.

5.1.8.3 FeedBackSrcMode (Axis Position Feedback Source Mode)

5.1.8.3.1 Basic properties

Type	USINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	0: The feedback input source is the actual physical connection input of the axis, such as bus input in 50, and 51 modes 1: The feedback input source is a user-defined input
Related Chapters	HMC_SetFeedBackSrcAddr (Setting user-defined feedback input address)

Type	USINT
	FeedBackSrcValue (Axis feedback self-defined input real-time value)

5.1.8.3.2 Meaning description

It is used to set the input source mode of the axis feedback value Mpos, which is only valid for axis types with actual input, such as bus axes.

When FeedBackSrcMode=0, the Mpos input source is the actual physical connection input of the axis. When FeedBack SrcMode=1, the MPos input source is the user-defined input, and the user-defined address is set through the HMC_SetFeedBackSrcAddr function.

5.1.8.4 FeedBackSrcValue (Axis Feedback Self-Defined Input Real-Time Value)

5.1.8.4.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	HMC_FeedBackSrcAddr
Initial Value	0
Value Range	Floating point number
Related Chapters	HMC_SetFeedBackSrcAddr (Setting user-defined feedback input address) FeedBackSrcMode (Axis position feedback source mode)

5.1.8.4.2 Meaning description

For an axis with actual feedback input, you can set FeedBackSrcMode=1 and call HMC_SetFeedBackSrcAddr to configure a parameter address as the feedback input source of the axis at this time. In other words, the Mpos of the axis at this time is calculated based on the parameters corresponding to this address. FeedBackSrcValue displays associated self-defined parameter values in real time.

5.1.8.5 FeedBackDataType (Axis Feedback Self-Defined Input Variable Data Type)

5.1.8.5.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	HMC_SetFeedBackSrcAddr
Initial Value	0
Value	USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, REAL=7, LREAL=8

Type	USINT
Range	
Related Chapters	HMC_SetFeedBackSrcAddr (Setting user-defined feedback input address) FeedBackSrcMode (Axis position feedback source mode)

5.1.8.5.2 Meaning description

FeedBackDataType indicates the data type of user-defined input variables. FeedBackSrcValue displays associated self-defined parameter values in real time.

5.1.9 Outputting Related Parameters (Output)

5.1.9.1 KsNumerator/KsDenominator (Numerator/denominator of stepper output ratio Ks)

5.1.9.1.1 Basic properties

Type	DINT
Read/Write Property	Read only
Set Function	HMC_SetKs
Initial Value	1
Value Range	DINT integers: Neither the numerator nor the denominator is 0, but with the same sign
Related Chapters	HMC_SetKs (Setting Ks)

5.1.9.1.2 Meaning description

Numerator/denominator of stepper output ratio Ks, which is set through HMC_SetKs. This parameter is only valid for EtherCAT_Position (50). Refer to HMC_SetKs(Setting Ks).

5.1.9.2 PulseNum (Number of Periodic Output Pulses of Stepper Axes)

5.1.9.2.1 Basic properties

Type	UDINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	0xffffffff
Related Chapters	-

5.1.9.2.2 Meaning description

Number of pulses output by the stepper axis per servo cycle.

5.1.9.3 ServoLoop (Closed-Loop Output Switch)

5.1.9.3.1 Basic properties

Type	USINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	0: It indicates open loop, and the analog output (DACOut) adopts the value given by the DAC parameter 1: It indicates closed loop, and the analog output (DACOut) is the result of PID operation
Related Chapters	-

5.1.9.3.2 Meaning description

It indicates the closed-loop output switch. This parameter is only valid for axis types in position closed-loop mode, such as EtherCAT bus speed mode 51. This switch is set to determine whether the position closed-loop takes effect:

(1) ServoLoop=0 (open loop)

The axis is in position open-loop mode. At this time, Mpos and MpSpeed come from the measured values of the encoder, and DPos tracks MPos. The output value DACOut adopts the value given by the DAC parameter, that is, open-loop mode.

(2) ServoLoop= 1 (closed loop)

The axis is in position closed-loop control mode. Dpos and VpSpeed are the real-time calculated values of the motion control instructions. Mpos and MpSpeed are the measured values from the encoder. By performing PID calculation on Dpos and MPos, the PID updates the calculation result to the output value DACOut to achieve the position closed-loop control effect.

It should be noted that when an exception occurs in the system, it will be displayed through the axis parameter AxisStatus. When the logical AND operation of AxisStatus and AxisErrMask is non-zero, the system will handle the exception. At this time, ServoLoop will be forcibly modified to 0 by the system.

5.1.9.4 DAC (Speed Mopen-Loop Output Value)

5.1.9.4.1 Basic properties

Type	DINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	-2, 147,483,648~2, 147,483,647
Related Chapters	-

5.1.9.4.2 Meaning description

Output value of DACOut when ServoLoop=0.

5.1.9.5 DACOut (Speed Mode Output Value)

5.1.9.5.1 Basic properties

Type	DINT
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	-2, 147,483,648~2, 147,483,647
Related Chapters	-

5.1.9.5.2 Meaning description

It indicates the speed output result of the current axis (in EtherCAT_Speed (51), the speed is directly sent to the servo driver through the bus).

5.1.9.6 PosOffset (DACOut Positive Dead Band Value)

5.1.9.6.1 Basic properties

Type	DINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Bus axis: -2,147,483,648~2,147,483,647
Related Chapters	-

5.1.9.6.2 Meaning description

Positive dead band value of servo bus axis output.

When the output value DACOut calculated by the servo bus axis is positive, the positive dead band value PosOffset is superimposed on this DACOut. When DACOut is 0, the output is fixed at (PosOffset+NegOffset)/2.

That is, after the dead band is set, the final output DACOut is always outside the voltage dead band range of the servo actuator, so as to prevent the actuator from being in the dead band for a long time and affecting the control effect.

For servo bus axes, the DACOut value range is -2,147,483,648~2,147,483,647. It needs to be calculated and set according to the actuator situation.

Application example

If there is a dead band in the bus axis actuator, and the corresponding voltage of the dead band is (-1, 2) V, you need to set NegOffset=-3278 and PosOffset=6553.

5.1.9.7 NegOffset (DACOut Negative Dead Band Value)

5.1.9.7.1 Basic properties

Type	DINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Bus axis: -2147483648~2147483647
Related Chapters	-

5.1.9.7.2 Meaning description

Negative dead band value of bus axis output.

When the output value DACOut calculated by the bus axis is negative, the negative dead band value NegOffset is superimposed on this DACOut. When DACOut is 0, the output is fixed at (PosOffset+NegOffset)/2. That is, after the dead band is set, the final output DACOut is always outside the voltage dead band range of the servo actuator, so as to prevent the actuator from being in the dead band for a long time and affecting the control effect.

5.1.9.8 ZeroWindow (DACOut Zero Gap Dead Band)

5.1.9.8.1 Basic properties

Type	DINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	Bus axis: -2147483648~2147483647
Related Chapters	-

5.1.9.8.2 Meaning description

DACOut zero gap dead band. When DACOut is located in the dead band window [-ZeroWindow,

ZeroWindow], it outputs the median value of NegOffset and PosOffset (i.e., (PosOffset+NegOffset)/2).

5.1.9.9 OutDstMode (DACOut Output Target Mode)

5.1.9.9.1 Basic properties

Type	USINT
Read/Write Property	Read/Write
Set Function	-
Initial Value	0
Value Range	0: The target of the DACOut output value calculated by the axis is the actual physical connection. 1: The target of the DACOut output value calculated by the axis is a user-defined variable.
Related Chapters	HMC_SetOutDstAddr (Setting user-defined output address)

5.1.9.9.2 Meaning description

For axis types in speed mode, such as EtherCAT bus speed mode 51, you can specify the final output target of the output value DACOut by setting OutDstMode.

(1) OutDstMode=0

The target of the DACOut output calculated by the axis is the actual physical connection. For example, the target output in mode 51 is the bus output channel.

(2) OutDstMode= 1

The target of the DACOut output value calculated by the axis is a user-defined variable, and this variable is displayed through OutDstValue. HMC_SetOutDstAddr is used to set user-defined variables.

At this time, the status of the physical output channel is as follows: for mode 51, for safety reasons, the bus driver speed will be set to 0 to stop the bus driver.

5.1.9.10 OutDstValue (DACOut Output Self-Defined Address Real-Time Value)

5.1.9.10.1 Basic properties

Type	LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	-

Type	LREAL
Related Chapters	HMC_SetOutDstAddr (Setting user-defined output address)

5.1.9.10.2 Meaning description

When OutDstMode=1, OutDstValue displays user-defined variable values in real time.

5.1.9.11 OutDstDataType (DACOut Output Self-Defined Variable Data Type)

5.1.9.11.1 Basic properties

Type	USINT
Read/Write Property	Read only
Set Function	HMC_ SetOutDstAddr
Initial Value	0
Value Range	USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, REAL=7, LREAL=8
Related Chapters	HMC_SetOutDstAddr (Setting user-defined output address)

5.1.9.11.2 Meaning description

When OutDstMode=1 is set, call HMC_SetOutDstAddr to configure the DACOut output of the axis to a user-defined variable. OutDstDataType represents the data type of the user-defined variable. OutDstValue displays user-defined variable values in real time.

5.1.10 Reserved Parameters

5.1.10.1 SystemValue*

5.1.10.1.1 Basic properties

Type	SystemValue0, SystemValue1: UDINT SystemValue2, SystemValue3: LREAL
Read/Write Property	Read only
Set Function	-
Initial Value	0
Value Range	-
Related Chapters	-

5.1.10.1.2 Meaning description

It is used for system debugging or recording system fault information. Currently, some reserved axis parameters with fixed axis numbers have been used, and their specific meanings are as follows:

Axis number	SystemValue0	SystemValue1	SystemValue2	SystemValue3
0	Algorithm solution time	Algorithm minor version number	-	-
1	Algorithm scheduling interval	Last system error code	-	-
2	Maximum algorithm solution time		-	-
3	Last user error code	Last user error message	-	-
4	Total execution time of motion control algorithm and protocol module	The cumulative number of times that the total execution time of the motion control algorithm and protocol module exceeds the servo cycle	-	-

5.2 System Startup Instructions

5.2.1 HMC_DriveEnable (Servo Enable)

5.2.1.1 Function

Control the on/off of the enable signals of all servo drivers connected to the controller.

(1) For bus-type servo drivers, the DriveEnable signal is sent through bus communication.

(2) When DriveEnable is disabled, each axis of the motion controller stops outputting.

5.2.1.2 Parameter

Parameter	Statement	Data Type	Description
bMode	Input	USINT	0: Disabled 1: Enabled
HMC_DriveEnable	Output	UDINT	0 Set successfully 0xFFFFFFFF function parameter error

5.2.1.3 Instruction type

It is a non-axis motion cache instruction.

Additional information

☐ This instruction controls the servo enable status of all drivers connected to the controller and is the servo enable master switch. For bus axis types, it also can implement servo sub-enable control, that is, to individually control the enable status of an axis. For details, see the bus-related instructions RTEX_DriveEnable and EtherCAT_DriveEnable.

☐ By clicking the enable button in the AutoThink menu bar, you can also switch the servo enable status.

☐ When some exception occurs on an axis, the bit corresponding to its AxisStatus will be set to 1. If the bit corresponding to AxisErrMask is also 1, the exception handler may automatically disable the DriveEnable signal.

5.2.1.4 Example

The following example uses the HMC_DriveEnable instruction to turn the servo on and off.

```
HMC_DriveEnable (1); (*Turn on servo enable*)
HMC_DriveEnable (0); (*Turn off servo enable*)
```

5.2.2 HMC_GetDriveEnableState (Getting Servo Enable Status)

5.2.2.1 Function

Get the servo total enable (DriveEnable) status.

5.2.2.2 Parameter

Parameter	Statement	Data Type	Description
HMC_GetDriveEnableState	Output	USINT	0: Disabled 1: Enabled

5.2.2.3 Instruction type

It is a non-axis motion cache instruction.

5.2.2.4 Additional Information

For bus axis types, you can also get the servo sub-enable status of an axis. For details, see EtherCAT_ GetDriveEnableState.

5.2.2.5 Example

The following example uses the HMC_GetDriveEnableState instruction to get the servo total enable (DriveEnable) status.

State:=HMC_GetDriveEnableState (); (*Get servo total enable status*)

5.3 Single Axis Motion Instructions

5.3.1 Origin Search and Position Definition

5.3.1.1 HMC_Home (Origin Regression)

5.3.1.1.1 Function

The device generally needs to find the mechanical zero point first before it can work normally after being powered on. This function determines the origin point for the axis by detecting the origin switch signal or the Z-phase signal of the servo motor encoder. The origin regression method is determined by the parameter ucHomeMode. The home switch signal is obtained through the DI channel of the controller DI module or the servo DI channel. The Z-phase signal is obtained from the servo driver by probe capture.

5.3.1.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucHomeMode	Input	USINT	It supports the following homing modes in CIA402 protocol: 1/2/3/5/7/11/17/18/19/21/23/27/33/34/35 Note: For details on the origin regression mode, refer to the MC_Home instruction in the PLCopen motion control instructions.
ucDiChl	Input	UDINT	DI channel number range used during origin regression: 0-2097151
HMC_Home	Output	UDINT	-

5.3.1.1.3 Instruction type

Axis motion cache instructions.

5.3.1.1.4 Precautions

(1) The Home mode related t to the Z signal only supports EtherCAT bus servo axes.

(2) The HMC_Home design uses a query mechanism to get the trigger signal. The query cycle is one servo cycle. Therefore, when the high level duration of the trigger signal is longer than or equal to one servo cycle, the trigger signal can be captured in each query cycle. If the high level duration of the trigger signal is shorter than one servo cycle, there is no guarantee that the origin signal can be captured in every query cycle, which will affect the normal operation of origin regression.

5.3.1.1.5 Example

For examples, see the example description of the PLCopen instruction MC_Home.

5.3.1.2 HMC_DefOrigin (Setting Origin)

5.3.1.2.1 Function

It is a moving coordinate system used to change the origin of a certain axis coordinate system and set the position in the original coordinate system calibrated by the parameter dMpos as the new coordinate system origin.

5.3.1.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis ID
dMpos	Input	LREAL	To be defined as the origin position
HMC_DefOrigin	Output	UDINT	Return value

5.3.1.2.3 Instruction type

It is a non-axis motion cache instruction.

5.3.1.2.4 Example

The following example uses the HMC_DefOrigin instruction to set the origin of the axis.

HMC_Def0rigin(Axis0.AxisID, 1000); (*Set the position where the axis Axis0 coordinate is 1000 as the new coordinate origin*)

5.3.1.3 HMC_DefPos (Setting Current Position)

5.3.1.3.1 Function

Moving coordinate system, with the current position defined as dMpos.

When this function is called when the axis is moving at high speed, deviations may occur when the controller is processing timing issues. At this time, to ensure that the origin position is accurately defined, the HMC_DefOrigin instruction can be used.

5.3.1.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dMpos	Input	LREAL	Specified value of current position
HMC_DefPos	Output	UDINT	Return value

5.3.1.3.3 Instruction type

It is a non-axis motion cache instruction.

5.3.1.3.4 Example

(1) Example 1

HMC_DefPos (Axis 0.AxisID, 0); (*Set the current coordinate position of Axis0 as the origin of coordinates*)

(2) Example 2

HMC_DefPos (Axis0.AxisID, 1000); (*Set the current coordinate position of Axis0 to 1000*)

5.3.2 One-way Motion

5.3.2.1 HMC_Forward (Forward Motion)

5.3.2.1.1 Function

It enables continuous forward motion, with the motion speed determined by the axis parameter Speed, and the acceleration and deceleration determined by the axis parameters Accel and Decel. During the motion process, you can modify such axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

5.3.2.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_Forward	Output	UDINT	Return value

5.3.2.1.3 Instruction type

Axis motion cache instructions.

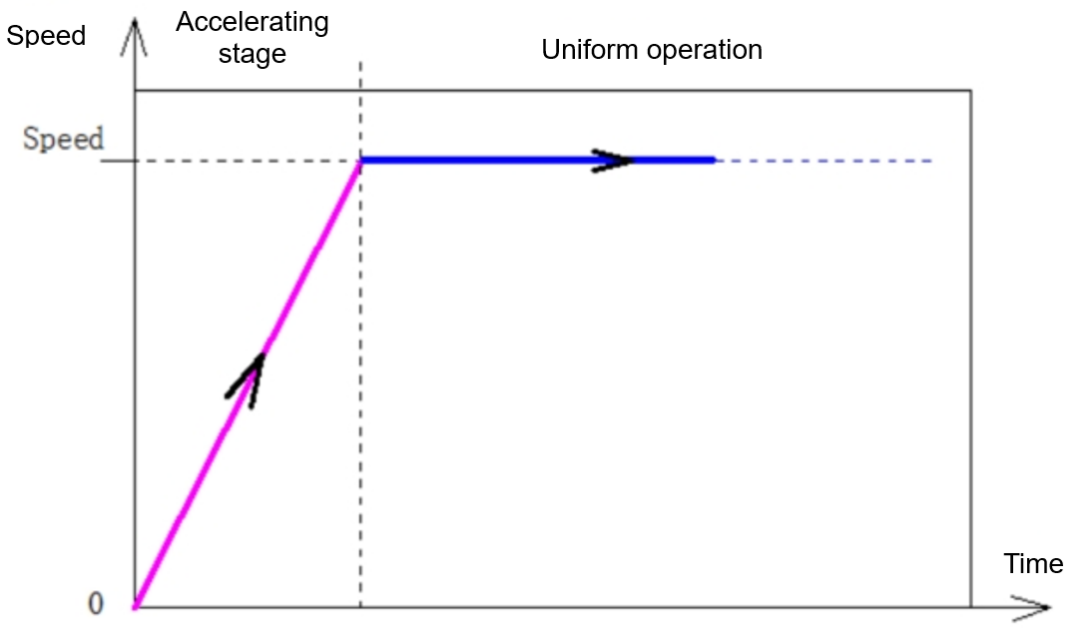
Additional information:

☐ If the instruction needs to be canceled to stop the axis during motion, you can use the HMC_Cancel instruction;

☐ If it encounters a forward limit during motion, it will decelerate and stop. Please refer to the handling of the limit.

5.3.2.1.4 Example

The following example uses the HMC_Forward instruction to implement the forward motion of the axis. The motion speed is determined by the axis parameter Speed, and the acceleration and deceleration are determined by the axis parameters Accel and Decel. During the motion process, you can modify such axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.



HMC_Forward

(1) Example 1: The example program is as follows (ST language), and the speed-time curve of the running process is shown in the figure above:

Axis 0.Speed:= 1000; (*Set motion speed*)

Axis 0.Accel:= 2000; (*Set the acceleration*)

Axis 0.Decel: = 2000; (*Set the deceleration*)

HMC_Forward(Axis0.AxisID); (*Send the HMC_Forward instruction to Axis0*)

(2) Example 2: Adjust speed and acceleration in real time, and cancel the stop

The example program is as follows (ST language):

Axis 0.Speed:= 2000; (*Set motion speed*)

Axis 0.Accel:= 1000; (*Set the acceleration*)

Axis 0.Decel: = 2000; (*Set the deceleration*)

HMC_Forward(Axis0.AxisID); (*Send the HMC_Forward instruction to Axis0*)

TimeDelay(500); (*Delay 500 milliseconds*)

Axis0.Accel :=2000; (*Change acceleration, valid in real time*)

TimeDelay(1500); (*Delay 1500 milliseconds*)

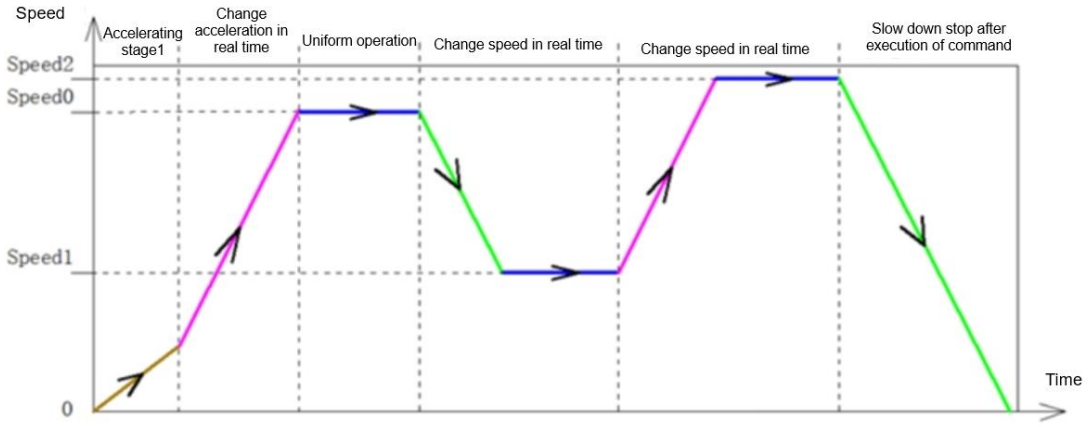
Axis0.Speed := 1000; (*Reduce speed, valid in real time*)

TimeDelay(1500); (*Delay 1500 milliseconds*)

Axis0.Speed := 2500; (*Increase speed, valid in real time*)

TimeDelay(1500); (*Delay 1500 milliseconds*)

HMC_Forward(Axis 0.AxisID, 1); (*Cancel the current HMC_Forward instruction of Axis 0*)



HMC_Forward Real-time Adjustment of Speed and Acceleration During Motion

5.3.2.2 HMC_Reverse (Reverse Motion)

5.3.2.2.1 Function

It performs continuous reverse motion, with the motion speed determined by the axis parameter Speed. During the motion process, you can modify such axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

5.3.2.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_Reverse	Output	UDINT	Return value

5.3.2.2.3 Instruction type

Axis motion cache instructions.

Additional information:

☐ If the instruction needs to be canceled to stop the axis during motion, you can use the HMC_Cancel instruction;

☐ If it encounters a reverse limit during motion, it will decelerate and stop. Please refer to the handling of the limit.

5.3.2.2.4 Example

Refer to HMC_Forward (Forward Motion).

5.3.2.3 HMC_Forward Ex (Advanced Forward Motion)

5.3.2.3.1 Function

When this instruction is executed, it needs to modify the axis parameter Speed= function parameter dSpeed first, and then the axis continues to move forward at the speed set by Speed. During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.

5.3.2.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dSpeed	Input	LREAL	Target speed to execute this motion
HMC_Forward Ex	Output	UDINT	Return value

5.3.2.3.3 Instruction type

Axis motion cache instructions.

Additional information:

☐ If the instruction needs to be canceled to stop the axis during motion, you can use the HMC_Cancel instruction;

☐ If it encounters a forward limit during motion, it will decelerate and stop. Please refer to the handling of the limit.

5.3.2.3.4 Example

The following example uses the HMC_Forward Ex instruction to implement the forward motion of the axis. When this instruction is executed, it needs to modify the axis parameter Speed= function parameter dSpeed first, and then the axis continues to move forward at the speed set by Speed. During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.

Example 1:

The example program is as follows (ST language):

Axis0.Speed:= 1000; (*Set motion speed*)

Axis0.Accel:= 2000; (*Set the acceleration*)

Axis0.Decel: = 2000; (*Set the deceleration*)

HMC_ForwardEx(Axis 0.AxisID); (*Send the HMC_Forward instruction to Axis 0)

Example 2: Modify speed, acceleration, and deceleration in real time

Refer to the example in HMC_Forward (Forward motion).

5.3.2.4 HMC_ReverseEx (Advanced Reverse Motion)

5.3.2.4.1 Function

When this instruction is executed, it first modifies the axis parameter Speed = function parameter dSpeed, and then continues to move in the reverse direction at the speed set by Speed. During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.

5.3.2.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dSpeed	Input	LREAL	Target speed to execute this motion
HMC_ReverseEx	Output	UDINT	Return value

5.3.2.4.3 Instruction type

Axis motion cache instructions.

Additional information:

☐ If the instruction needs to be canceled to stop the axis during motion, you can use the HMC_Cancel instruction;

☐ If it encounters a reverse limit during motion, it will decelerate and stop. Please refer to the handling of the limit.

5.3.2.4.4 Example

Refer to HMC_ForwardEx (Advanced forward motion).

5.3.3 Point Motion

5.3.3.1 HMC_Move (Relative Motion)

5.3.3.1.1 Function

Let an axis move at the preset speed according to the acceleration and deceleration parameters to the relative displacement relative to the position when this instruction starts to be executed (positive or negative). The motion speed is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the motion process, you can modify such axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

The HMC_Move instruction is the most typical motion control function in the HMC motion control system, which can control the independent motion of a single axis. The motion of Axes without interpolation or synchronization relationships can be independently controlled by simply executing HMC_Move.

The first parameter of HMC_Move is the axis number, and the second parameter is the displacement that the axis will move from the current position. The positive and negative values represent the direction of motion. The

displacement unit can be adjusted through the axis parameters Units, Ke, and Ks. For details, see the relevant axis parameters.

(1) In the case of uniform acceleration and uniform deceleration, if the set target speed (Speed) can be reached and there is a uniform speed section, the speed-time curve during the motion is divided into three sections (acceleration section, uniform speed section, deceleration section), which appears as a trapezoid; otherwise, it is divided into two sections (acceleration section, deceleration section), which appears as a triangle.

(2) During the execution of the instruction, for the limited stroke, the corresponding axis parameters have the following relationship:

Endpos = Dpos+Remain

For cycle mode, during the completion of the Move instruction, the relationship between Endpos, Remain, Dpos, and Mpos is detailed in RepMode and RepDist.

(3) In position open-loop mode (such as stepper axis), when Dpos=End pos and Remain=0 are satisfied, the instruction execution is completed. In position closed-loop mode (such as bus servo axis), to judge whether the execution is completed, it may also need to see whether Mpos is in place. For details, see the axis parameters CmdStopMode and StopFETol.

(4) When the Jerk mode is enabled, the maximum acceleration and deceleration executed by the instruction will be limited by the Jerk function. For details, see Jerk (Jerk).

(5) When the Merge function is turned off and the HMC_Move instruction is completed, the speed is 0. When the Merge function is turned on and the Merge validity conditions are met, multiple instructions can be connected together and executed continuously to improve efficiency. For details, see Merge (Merging function mode).

5.3.3.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
d Distance	Input	LREAL	Incremental displacement (positive or negative)
HMC_Move	Output	UDINT	Return value

5.3.3.1.3 Instruction type

Axis motion cache instructions.

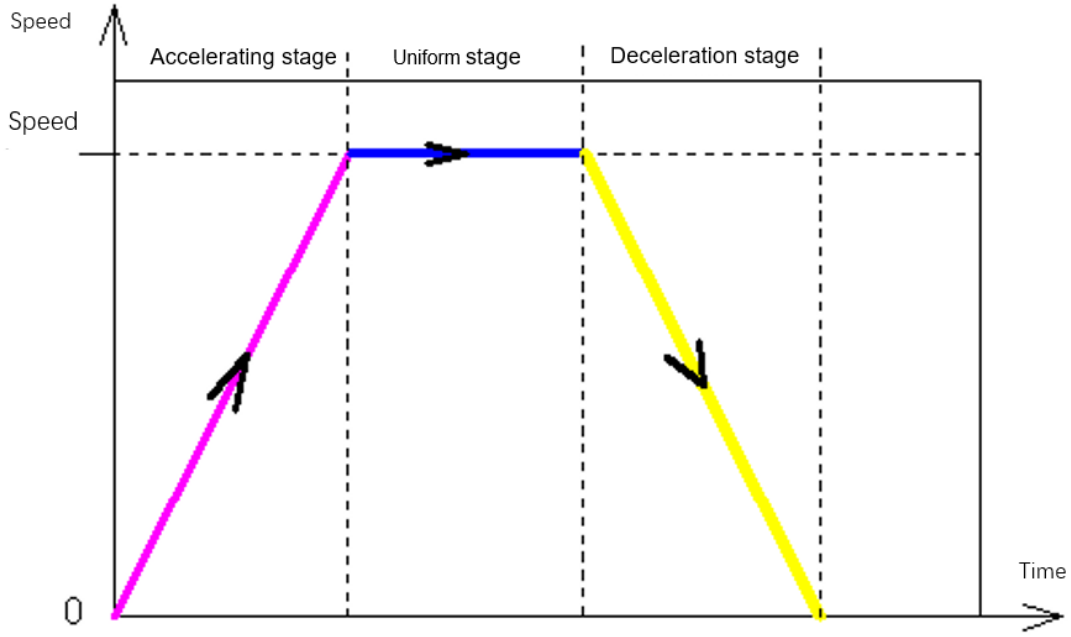
5.3.3.1.4 Example

The following example uses the HMC_Move instruction to implement the relative motion of the axis.

Example 1: Trapezoidal speed curve

When the set kinematic parameters satisfy the following relational expression, the speed curve during the instruction execution process is trapezoidal.

$$dDistance > \frac{Speed^2}{2}\left(\frac{1}{Accel}+\frac{1}{Decel}\right)$$



Trapezoidal Acceleration and Deceleration Curve

The example program is as follows (ST):

Axis0.Speed := 1000; (*Setting the speed of motion*)

Axis0.Accel := 2000; (*Setting the acceleration*)

Axis0.Decel := 2000; (*Setting the deceleration*)

HMC_Move(Axis0.AxisID，4000); (*Drop HMC_Move to Axis0 axis, the distance of incremental motion is 4000*)

Example 2: Triangular speed curve

When the set kinematic parameters satisfy the following relational expression, the speed curve during the instruction execution process is triangular.

$$dDistance \leq \frac{Speed^2}{2}\left(\frac{1}{Accel}+\frac{1}{Decel}\right)$$

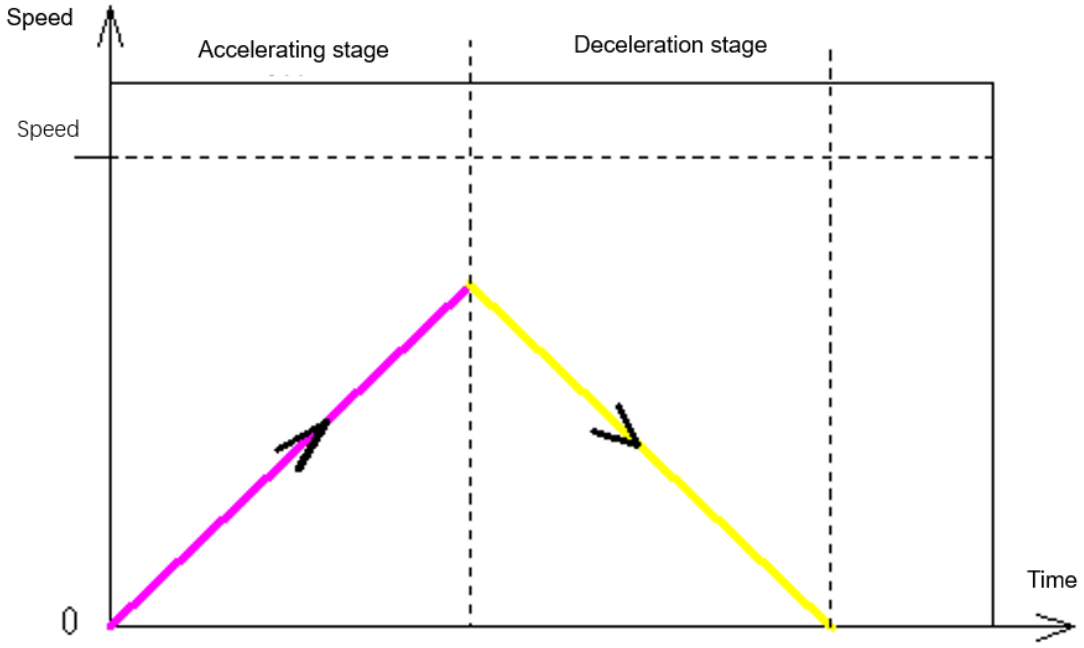
The example program is as follows (ST):

Axis0.Speed := 2000; (*Setting the speed of motion*)

Axis0.Accel := 1000; (*Setting the acceleration*)

Axis0.Decel := 1000; (*Setting the deceleration*)

HMC_Move(Axis0.AxisID，1000); (*Drop HMC_Move to Axis0 axis, the distance of incremental motion is 1000*)



Triangular Acceleration and Deceleration Curve

Example 3: Modify parameters in real time during motion

During the motion process, you can modify such axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

The example program is as follows (ST):

Axis0.Speed := 1000; (*Setting the speed of motion*)

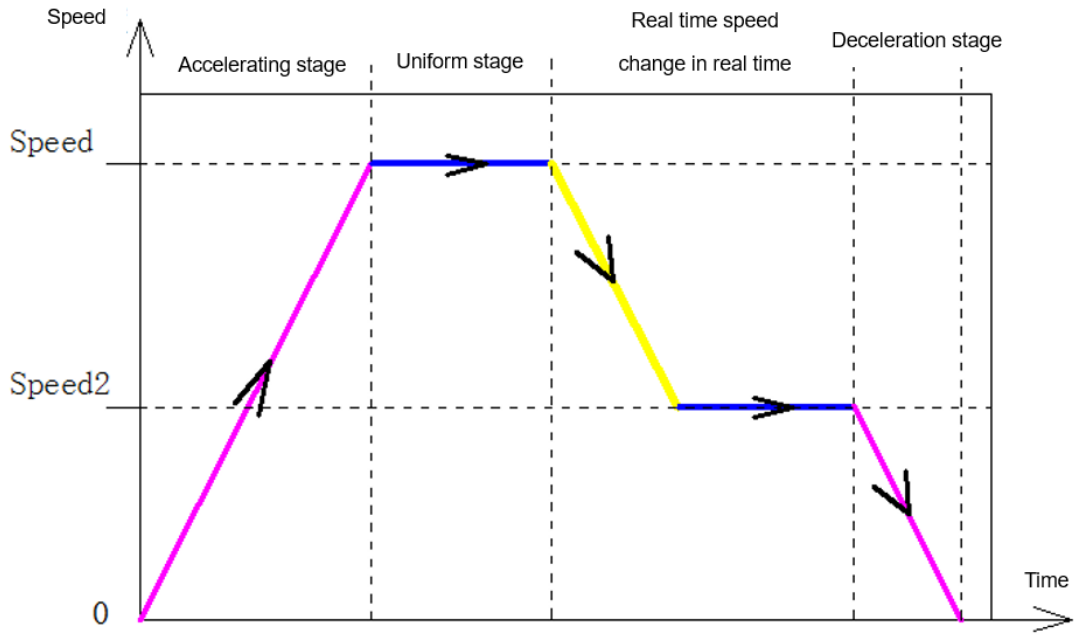
Axis0.Accel := 1000; (*Setting the acceleration*)

Axis0.Decel := 1000; (*Setting the deceleration*)

HMC_Move(Axis0.AxisID，3000); (*Drop HMC_Move to Axis0 axis*)

TimeDelay(2000); (*The time of delayed waiting is 2000s*)

Axis0.Speed := 500; (*Speed down to 500*)



Modifying Speed Parameters in Real Time During Motion

5.3.3.2 HMC_MoveAbs (Absolute Motion)

5.3.3.2.1 Function

Let an axis move to the coordinate position dDistance according to the preset acceleration, speed and acceleration parameters. The motion speed is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the motion process, you can modify such axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

5.3.3.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
d Distance	Input	LREAL	Incremental displacement (positive or negative)
HMC_Move	Output	UDINT	Return value

5.3.3.2.3 Instruction type

Axis motion cache instructions.

5.3.3.2.4 Example

The following example uses the HMC_MoveAbs instruction to implement the absolute motion of the axis.

Example 1: Make axis 0 to move for incremental displacements of 1000, 3000, -1000, and 2000 in sequence. The program implemented using HMC_Move and HMC_MoveAbs is as follows (ST language):

HMC_Move implemented program:

Axis0.Speed := 1000; (*Setting the speed of motion*)

Axis0.Accel := 2000; (*Setting the acceleration*)

Axis0.Decel := 2000; (*Setting the deceleration*)

HMC_Move(Axis0.AxisID，1000); (*Drop HMC_Move to Axis0 axis, incremental motion distance is 1000*)

HMC_Move(Axis0.AxisID，3000); (*Drop HMC_Move to Axis0 axis, incremental motion distance is 3000*)

HMC_Move(Axis0.AxisID，-1000); (*Drop HMC_Move to Axis0 axis, incremental motion distance is -1000*)

HMC_Move(Axis0.AxisID，2000); (*Drop HMC_Move to Axis0 axis, incremental motion distance is 2000*)

HMC_MoveAbs implemented program (assuming the starting point is 0):

Axis0.Speed := 1000; (*Setting the speed of motion*)

Axis0.Accel := 2000; (*Setting the acceleration*)

Axis0.Decel := 2000; (*Setting the deceleration*)

HMC_MoveAbs (Axis0.AxisID，0 + 1000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 1000*)

HMC_ MoveAbs (Axis0.AxisID，0 + 1000 + 3000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 4000*)

HMC_ MoveAbs (Axis0.AxisID，0 + 1000 + 3000 + (-1000)); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 3000*)

HMC_ MoveAbs (Axis0.AxisID，0 + 1000 + 3000 + (-1000) + 2000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 5000*)

Example 2: Make axis 0 to move to the absolute coordinates 1000, 3000, -1000, and 2000 in sequence. The program implemented using HMC_Move and HMC_MoveAbs is as follows (ST language):

HMC_Move implemented program (assuming the starting point is 0):

Axis0.Speed := 1000; (*Setting the speed of motion*)

Axis0.Accel := 2000; (*Setting the acceleration*)

Axis0.Decel := 2000; (*Setting the deceleration*)

HMC_Move(Axis0.AxisID，1000 - 0); (*Drop HMC_Move to Axis0 axis, incremental motion distance is 1000*)

HMC_Move(Axis0.AxisID，3000 - 1000); (*Drop HMC_Move to Axis0 axis, incremental motion distance is 2000*)

HMC_Move(Axis0.AxisID，- 1000 - 3000); (*Drop HMC_Move to Axis0 axis, incremental motion distance is -1000*)

HMC_Move(Axis0.AxisID，2000 – (-1000)); (*Drop HMC_Move to Axis0 axis, incremental motion distance is 2000*)

HMC_MoveAbs implemented program:

Axis0.Speed := 1000; (*Setting the speed of motion*)

Axis0.Accel := 2000; (*Setting the acceleration*)

Axis0.Decel := 2000; (*Setting the deceleration*)

HMC_MoveAbs (Axis0.AxisID，1000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 1000*)

HMC_ MoveAbs (Axis0.AxisID，3000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 3000*)

HMC_ MoveAbs (Axis0.AxisID，-1000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position -1000*)

HMC_ MoveAbs (Axis0.AxisID，2000); (*Drop HMC_MoveAbs to Axis0 axis, moving to absolute position 2000*)

It can be seen from the above examples that it is more suitable to use HMC_Move in incremental coordinates, and it is more convenient to use HMC_MoveAbs in absolute coordinates.

5.3.3.3 HMC_MoveEx (Advanced Relative Motion)

5.3.3.3.1 Function

When this instruction is executed, first modify the axis parameter Speed = function parameter dSpeed, and then move at the Speed to the relative displacement relative to the position when this instruction starts to be executed (positive or negative). During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.

5.3.3.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
d Distance	Input	LREAL	Incremental displacement (positive or negative)
dSpeed	Input	LREAL	Target speed to execute this motion
HMC_MoveEx	Output	UDINT	Return value

5.3.3.3.3 Instruction type

Axis motion cache instructions.

5.3.3.3.4 Example

The following example uses the HMC_MoveEx instruction to implement advanced relative motion of the axis. When this instruction is executed, it needs to modify the axis parameter Speed = function parameter dSpeed first, and then the axis moves to the relative displacement (either positive or negative) relative to the position where this instruction starts to execute at a speed set by Speed. During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time. For details, please refer to the function description section in HMC_Move (Relative motion).

Axis0.Speed := 1000; (*Setting the speed of motion*)

Axis0.Accel := 2000; (*Setting the acceleration*)

Axis0.Decel := 2000; (*Setting the deceleration*)

HMC_MoveEx(Axis0.AxisID，10000,2000); (*Drop HMC_MoveEx to Axis0, will running at a maximum speed of 2000 (instead of 1000)*)

5.3.3.4 HMC_MoveAbsEx (Advanced Absolute Motion)

5.3.3.4.1 Function

Axis parameter Speed= function parameter dSpeed, allowing an axis to move to the absolute position according to the target speed set by this instruction. During the motion process, correct such axis parameters as Speed, Accel, and Decel in real time to change the motion process, which is effective in real time.

5.3.3.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
d Distance	Input	LREAL	Incremental displacement
dSpeed	Input	LREAL	Target speed to execute this motion
HMC_MoveAbsEx	Output	UDINT	Return value

5.3.3.4.3 Instruction type

Axis motion cache instructions.

5.3.3.4.4 Example

Refer to HMC_MoveAbs(Absolute motion).

5.3.4 Instruction Correction

5.3.4.1 HMC_Cancel (Canceling Instruction)

5.3.4.1.1 Function

It is used to cancel the instructions in the motion cache of the specified axis. It supports three modes: canceling the current instruction, canceling the next instruction, and canceling all instructions for the axis.

(1) Cancel response modes for different instruction types

(a) Single-axis instruction

Cancel the current or all instructions for this axis.

(b) Interpolating combined axis

If the interpolation combined axis instruction is canceled, the interpolation instruction of all axes related to the interpolation instruction will be canceled, including combined axes and the split axes, regardless of whether the instruction is the current instruction in each interpolating split axis. For example, when the current motion instruction cache status of each axis is as follows, to cancel the current instruction for axis a,

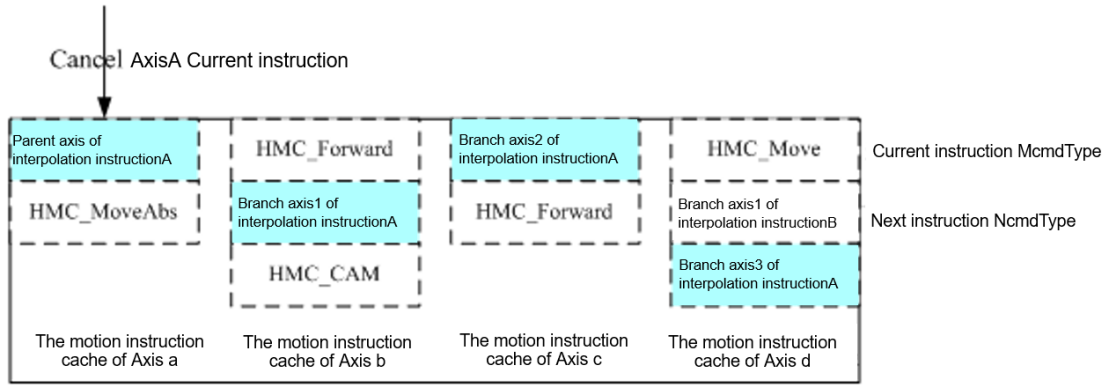


Diagram of Motion Instruction Cache Status

after the Cancel instruction is called, axis a responds immediately, regardless of whether the instruction of its slave axis is the current instruction. After the Cancel instruction is executed, the motion instruction cache status of each axis is as follows:

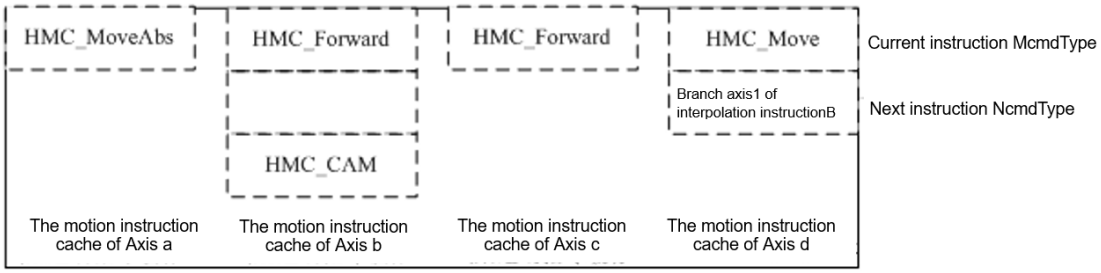


Diagram of Motion Instruction Cache Status

Note that after the Cancel interpolation instruction is executed, the instruction cache of the combined axis and the split axis will be deleted directly, that is, it will be empty. However, the subsequent instruction will not be adjusted to move up at this time, and, instead, it will wait for the execution of this instruction. If it is judged to be

empty, the subsequent instruction will be executed directly, which has no impact on actual execution efficiency.

(c) Interpolating split axis

The Cancel instruction is invalid for interpolating split axes.

(d) Linkage master axis

When the Cancel instruction is executed against the linkage instruction master axis, it only takes effect on the master axis, but has no effect on the slave axis.

(e) Linkage slave axis

When the Cancel instruction is executed against the linkage instruction split axis, it only takes effect on the split axis, but has no effect on the master axis.

(2) How the axis stops when the instruction is canceled

During the instruction cancellation process, after receiving the Cancel instruction, the axis decelerates at the current speed until it stops. However, the specific stopping method during the deceleration and stop process, such as actual deceleration and final stop position, can be specified by the axis parameters CancelMode and CancelPos. Details are as follows:

(a) For CancelMode=0, natural stop

It is the default value for the axis to stop naturally according to Decel. This mode supports all instruction types.

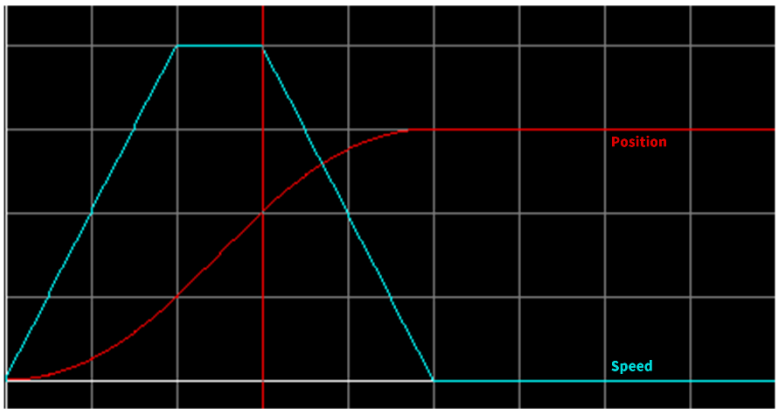


Diagram of Motion Trajectory

(b) For CancelMode=1, specifying the position to stop

The axis stops at the specified position at a speed not more than Decel, and the position is specified by the parameter CancelPos. This mode only takes effect for the cycle stroke mode (RepMode= 1 or 2) when the current instruction is one-way motion (HMC_Forward, HMC_Reverse). If it is a one-way motion instruction but is not in the cycle stroke mode, it is processed according to CancelMode=0.

According to the relative relationship between the current position and CancelPos, there are three situations: 1. Current position < stop position, and the distance displacement difference between the two is enough for the current speed to reduce to 0 at deceleration; 2. Current position < stop position, but the distance displacement difference

between the two is not enough for the current speed to reduce to 0 at deceleration, and; 3. Current position > stop position. The above three situations correspond to the following three figures respectively:

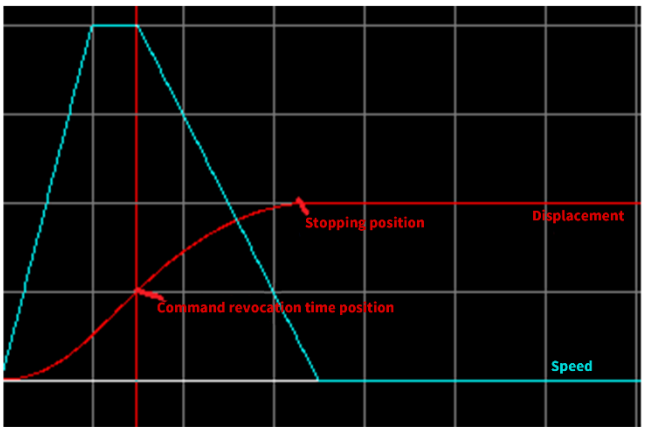


Diagram of Motion Trajectory

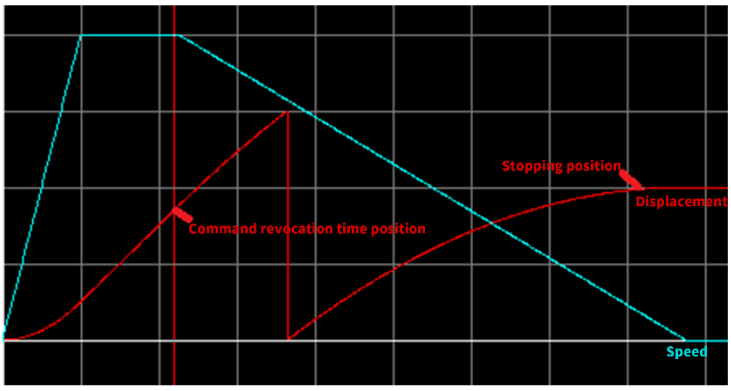


Diagram of Motion Trajectory

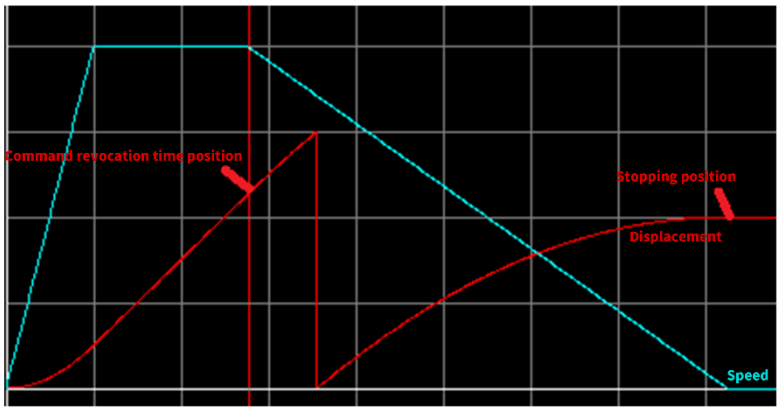


Diagram of Motion Trajectory

(c) For CancelMode=2, stopping immediately

The axis stops immediately and the speed is reduced directly from the current speed to 0. This mode is only effective for the axis that is executing the linkage instruction. If the status is not satisfied, it will be processed according to CancelMode=0 by default.

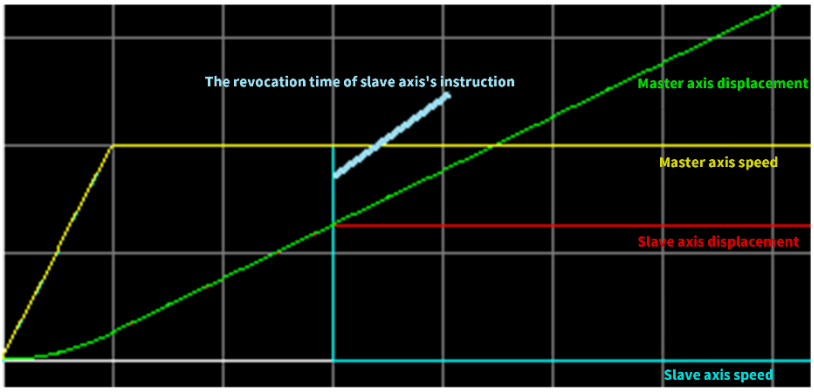


Diagram of Motion Trajectory

5.3.4.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucCancelMode	Input	USINT	0: Cancel the next instruction 1: Cancel the current instruction 2: Cancel all instructions for this axis
HMC_Cancel	Output	UDINT	Return value

5.3.4.1.3 Instruction type

It is a non-axis motion cache instruction.

5.3.4.1.4 Example

The following example uses the HMC_Cancel instruction to achieve that axis 0 performs one-way forward motion for 10 seconds, and then decelerates to a stop with Decel.

```
Axis0.Speed := 100000;
Axis0.Accel :=100000;
Axis0.Decel :=100000;
Axis0.CancelMode:=0;
HMC_Forward(0);
TimeDelay (10000);
HMC_Cancel(0, 1);
```

5.3.4.2 HMC_MoveModify (Target Position Change)

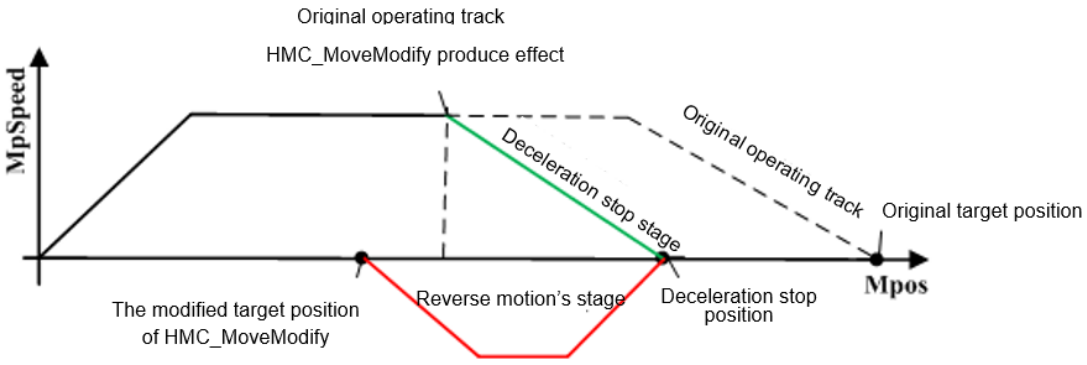
5.3.4.2.1 Function

Modify the target position of the HMC_Move/HMC_MoveAbs instruction in real time, and the modified target position is dDpos.

When the HMC_MoveModify is executed, if there is no instruction in the current axis instruction queue, the MoveAbs instruction will be directly sent; if there is an instruction but the currently executed instruction is not Move or MoveAbs, this modification request will be ignored.

When HMC_MoveModify is executed, there are the following processing methods:

(1) If the axis moves away from the end position specified by HMC_MoveModify, the axis will decelerate until it stops, and then move in the reverse direction to the position set by HMC_MoveModify.



Motion Diagram

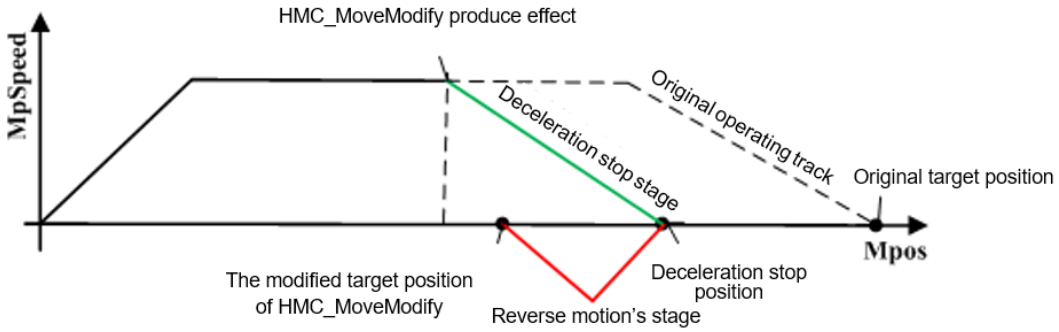
Sample program (ST):

p1 := HMC_Move(0, 10000);

p2:= TimeDelay(15); (*At this point, the location value of axis0.dpos has exceeded 5000 and continues to be away from 5000*)

p3:= HMC_MoveModify(0, 5000)

(2) If the axis moves towards the end position specified by HMC_MoveModify, but there isn't sufficient deceleration distance, the current axis will decelerate and stop, and then move in the reverse direction to the position set by HMC_MoveModify.



Motion Diagram

Sample program (ST):

p1 := HMC_Move(0, 10000);

p2 := TimeDelay(10); (*At this point, the position value of axis0.dpos is nearing 6000 and there is not enough

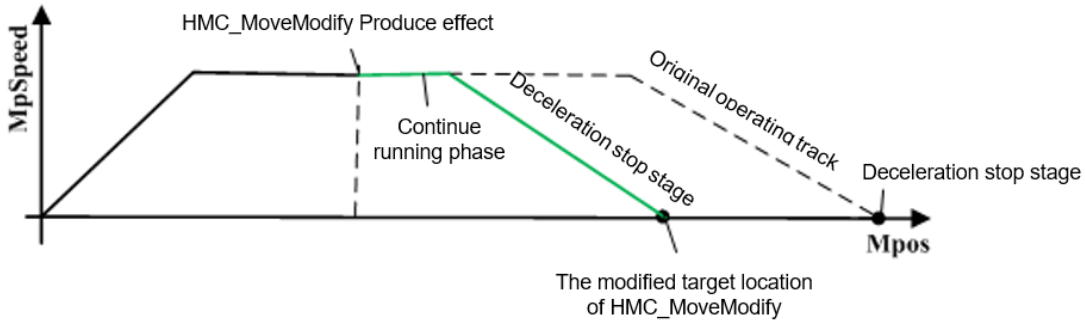
deceleration distance to stop it*)

p3 := HMC_MoveModify(0, 6100)

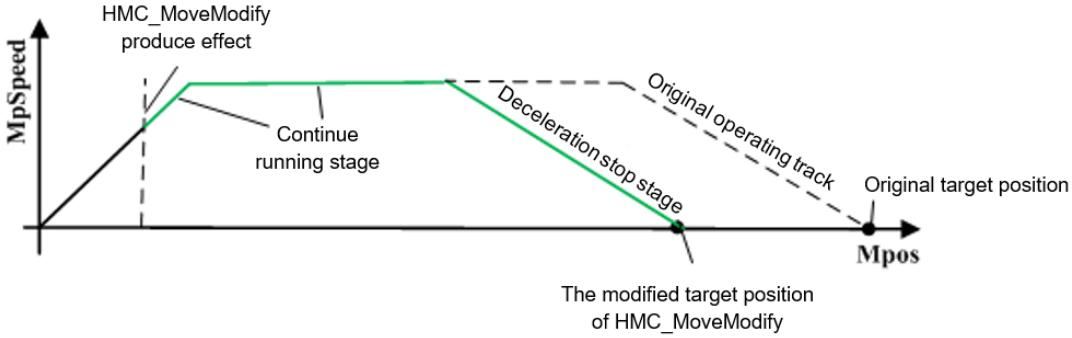
(3) If the axis moves towards the end position specified by HMC_MoveModify, and there is sufficient deceleration distance, the current axis will continue to move to the deceleration point, and then decelerate and stop at the position set by HMC_MoveModify.

Due to the difference in the execution timing of HMC_MoveModifyEx and the difference in the modified end position, the motion process of the "continue running stage" is different, and the uniform speed stage does not necessarily exist. Below are several typical diagrams of motion processes:

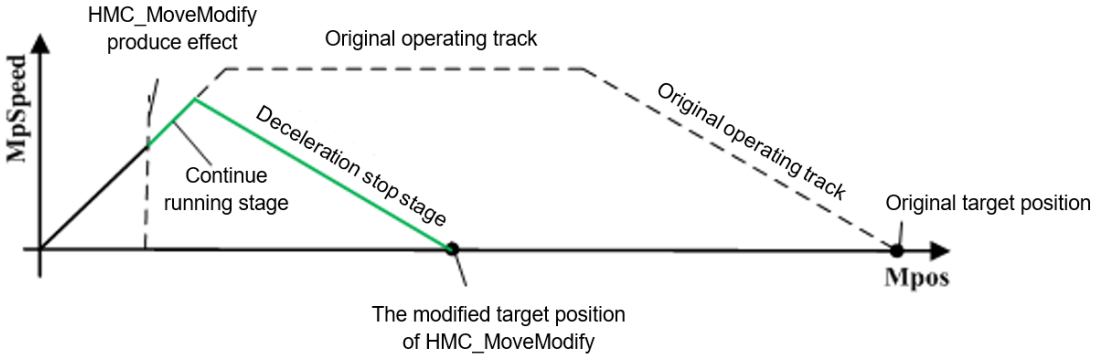
(a) The end position specified by HMC_MoveModify is before the original target position.



Set Position Before Original Target Position

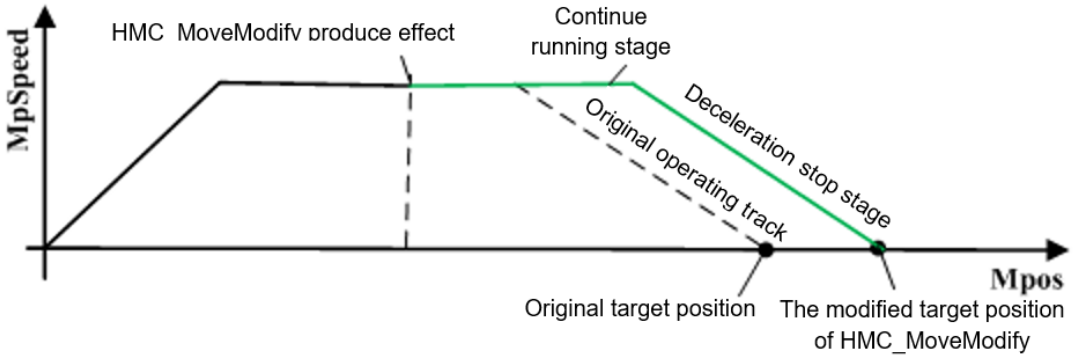


Set Position Before Original Target Position (HMC_MoveModify is executed during the acceleration stage, and there is a uniform speed section)

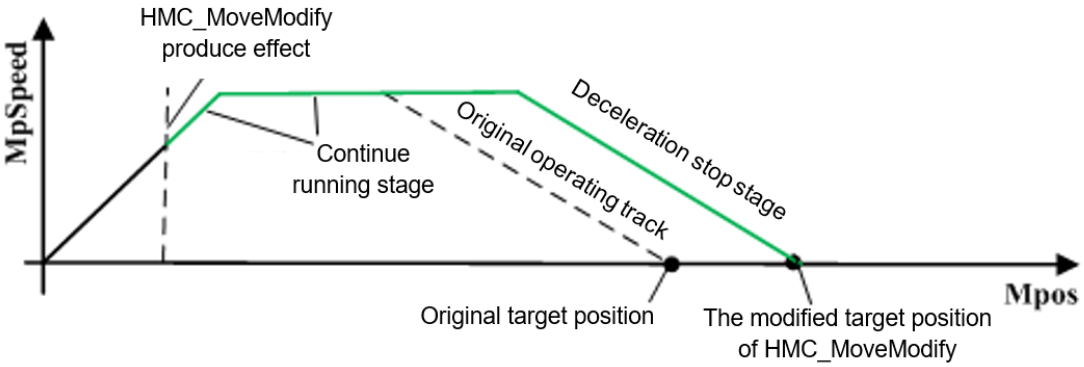


Set Position Before Original Target Position (HMC_MoveModify is executed during the acceleration stage, with no constant speed section)

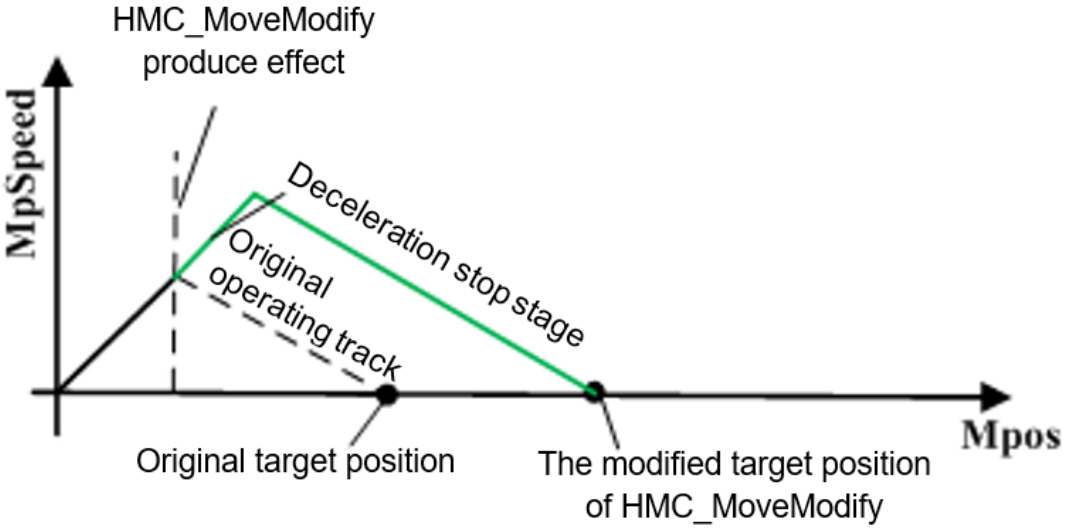
(b) The end position specified by HMC_MoveModify is after the original target position.



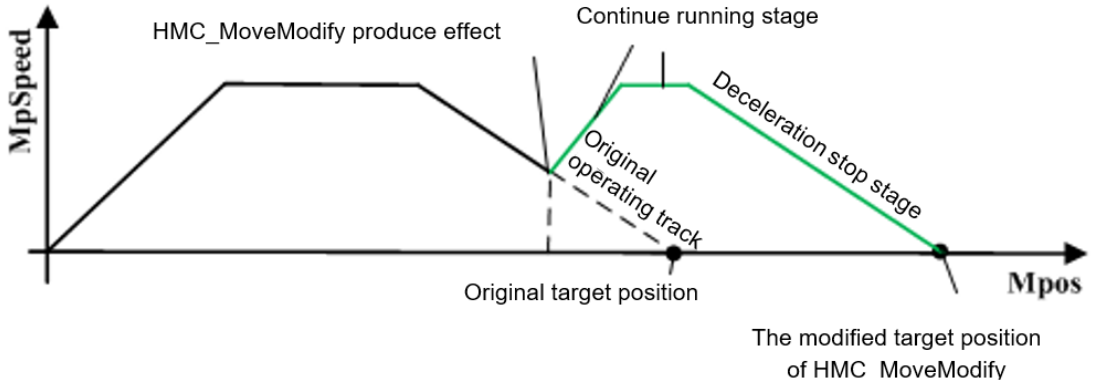
Set Position After Original Target Position (HMC_MoveModify is executed during the constant speed stage)



Set Position After Original Target Position (HMC_MoveModify is executed during the acceleration stage, and there is a constant speed section)



Set Position Before Original Target Position (HMC_MoveModify is executed during the acceleration stage, with no constant speed section)



Set Position After Original Target Position (HMC_MoveModify is executed during the deceleration stage)

Sample program (ST):

p1 := HMC_Move(0, 10000);

p2 := TimeDelay(2); (*The position value axis0.dpos at this time is far less than 5000, and there is enough remaining deceleration distance*)

p3 := HMC_MoveModify(0, 5000)

5.3.4.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dDpos	Input	LREAL	Modify the displacement of the current instruction
HMC_MoveModify	Output	UDINT	Return value

5.3.4.2.3 Instruction type

It is a non-axis motion cache instruction.

■ Additional information:

☐ If Merge is currently turned on, Merge will be automatically turned off when the HMC_MoveModify instruction is sent.

5.3.4.2.4 Example

Satellite signal receiving vehicle.

When measuring and controlling the operating status of the rocket during field launch and receiving satellite signals, the vehicle-mounted satellite receiver must make real-time adjustments to the horizontal direction and up-and-down pitch direction of the receiving antenna based on the data sent from the upper computer system to accurately track and control the status of the target. Then the HMC_MoveModify function in our controller can

meet this requirement very well. The corrected target position can be continuously loaded during the execution of this instruction to complete the follow-up tracking task.

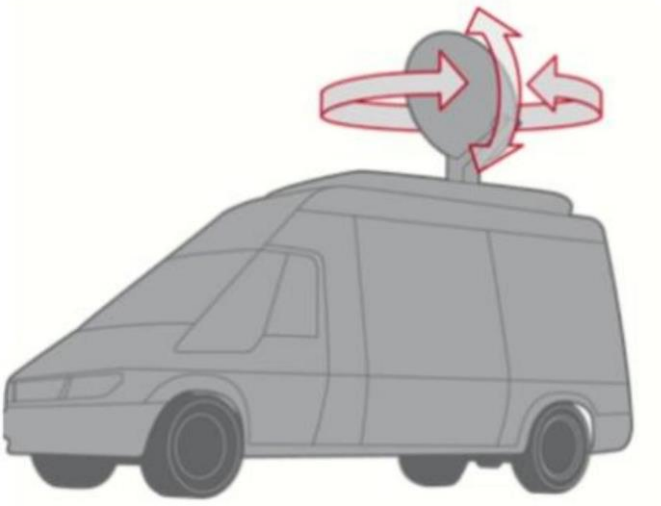


Diagram of Satellite Signal Receiving Vehicle

Assume that the horizontal correction position sent by the upper computer system is g_rHorizontalAdjPos, and the vertical pitch correction position is g_rVerticalAdjPos. Assume that the instruction pulse for turning 360 °in the horizontal direction is X, and the number of instruction pulses for turning 360 °in the vertical direction is Y. The specific program is as follows:

```
(***Parameter setting***)
Axis0UnitReval := HMC_SetUnits(0, X/3600); (*Set the AXIS0 unit to 0.1 degrees*)
Axis1UnitReval := HMC_SetUnits(1, Y/3600); (*Set the AXIS1 unit to 0.1 degrees*)
AxisEnableReval1 := HMC DriveEnable (1);
axis0. Speed := 100;
axis0. Accel := 1000;
axis0. Decel := 1000;
axis1. Speed := 100;
axis1. Accel := 1000;
axis1. Decel := 1000;
(****Subsequent program calculation, cycle scanning****)
WHILE true DO
IF bStart=1 THEN (*Device running*)
(*Correct horizontal direction*)
Axis0ModifyReva1 :=HMC_MoveModify(0, g_rHorizonta1AdjPos);
```

```
(*Correct vertical direction*)
Axis1ModifyReva1 :=HMC_MoveModify(1, g_rVertica1AdjPos);
END_IF
IF bStop=1 THEN (*Device stop*)
AxisOStopReVa1 := HMC_Cance1 (0, 2);
AxisIStopReVa1 := HMC_Cance1(1, 2);
RESULT := HMC_MoveAbs(0, 0);
RESULT := HMC_MoveAbs(1, 0);
RESULT := HMC_WaitAxisIdle(0);
RESULT := HMC_WaitAxisIdle(1)
END_IF
END_WHILE
```

5.3.4.3 HMC_MoveModifyEx (Advanced Target Position Change)

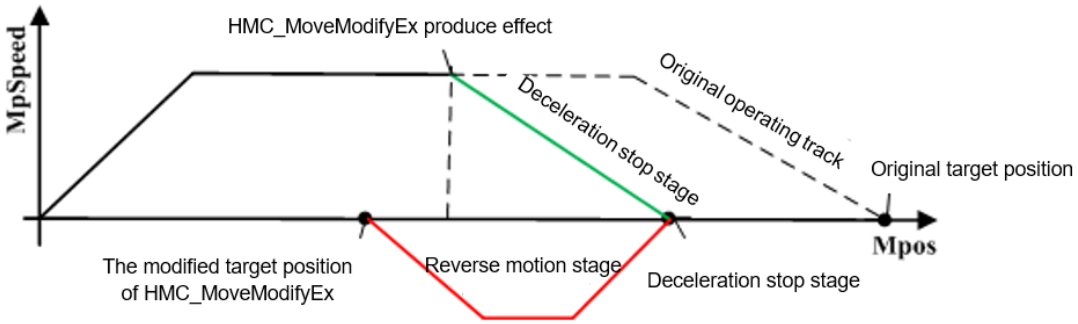
5.3.4.3.1 Function

It is used to modify the target position of the HMC_Move/HMC_MoveAbs instruction in real time. The modification method is determined by ucModifyMode: when ucModifyMode=0, this instruction is the same as HMC_MoveModify, which is the absolute position modification mode, and the modified target position is dDpos; when ucModifyMode=1, it is the incremental position modification mode, and the modified target position is the current dPos + dDpos.

When HMC_MoveModifyEx is executed, if there is no instruction in the current axis instruction queue, the HMC_MoveAbs instruction is snet when ucModifyMode=0, and the HMC_Move instruction is sent when ucModifyMode=1; if there is an instruction in the current axis instruction queue, and the instruction being executed is Move or MoveAbs, then the displacement of the current instruction will be modified; if there is an instruction in the current axis and the instruction being executed is not Move or MoveAbs, this modification request will be ignored.

The following example uses the HMC_MoveModifyEx instruction to modify the advanced position target. When HMC_MoveModifyEx is executed, there are the following processing methods:

(1) If the axis moves away from the end position specified by HMC_MoveModifyEx, the axis will decelerate until it stops, and then move in the reverse direction to the position set by HMC_MoveModifyEx. The diagram of the motion process of executing HMC_MoveModify in the uniform speed stage is as follows.



Motion Diagram

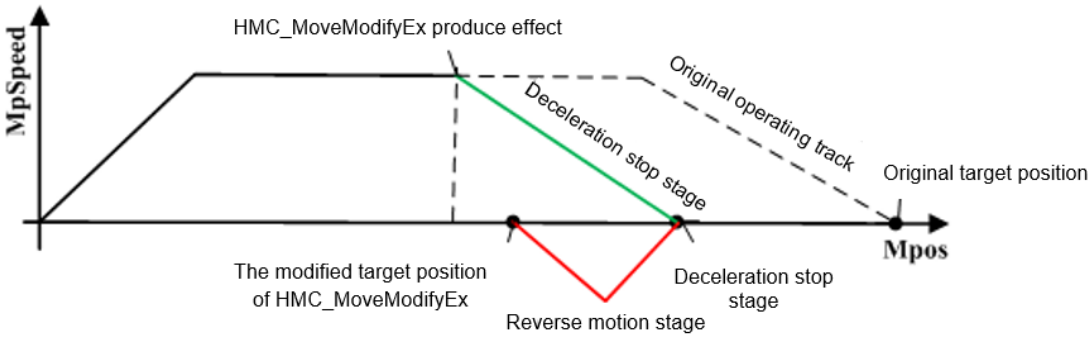
Sample program (ST):

p1 := HMC_Move(0, -10000);

p2 := TimeDelay(10); (*At this point, the position value has exceeded 5000 and continues to be far away from 5000*)

p3 := HMC_MoveModifyEx(0, 5000, 1);

(2) If the axis moves towards the end position specified by HMC_MoveModifyEx, but there isn't sufficient deceleration distance, the current axis will decelerate and stop, and then move in the reverse direction to the position set by HMC_MoveModifyEx. The diagram of the motion process of executing HMC_MoveModify in the uniform speed stage is as follows.



Motion Diagram

Sample program (ST):

p1 := HMC_Move(0, 10000);

p2 := TimeDelay(10);

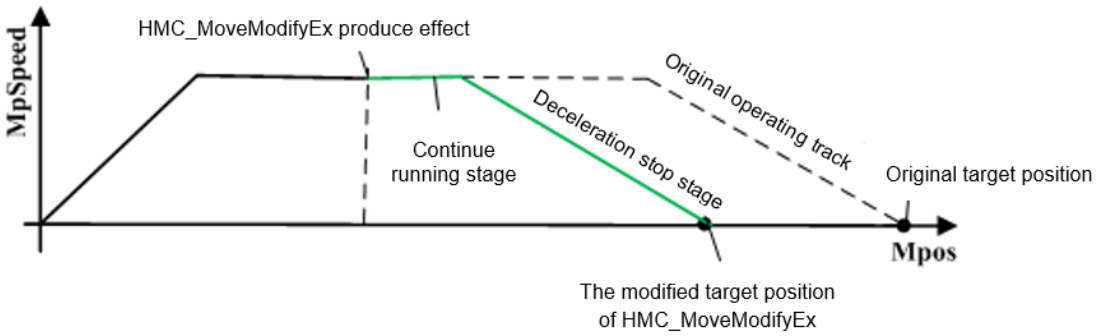
p3 := HMC_ MoveModifyEx(0,50,1); (*Incremental position value is too small to slow down*)

(3) If the axis moves towards the end position specified by HMC_MoveModifyEx, and there is sufficient deceleration distance, the current axis will continue to move to the deceleration point, and then decelerate and stop at the position set by HMC_MoveModifyEx.

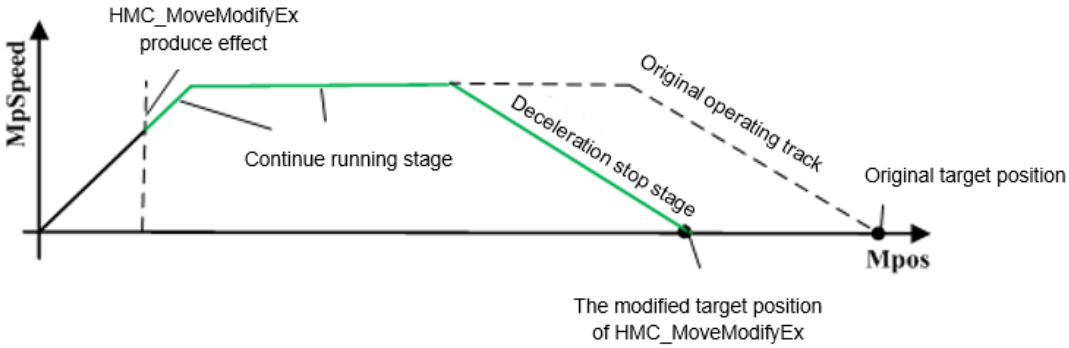
(4) Due to the difference in the execution timing of HMC_MoveModifyEx and the difference in the modified end position, the motion process of the "continue running stage" is different, and the uniform speed stage does not

necessarily exist. Below are several typical diagrams of motion processes:

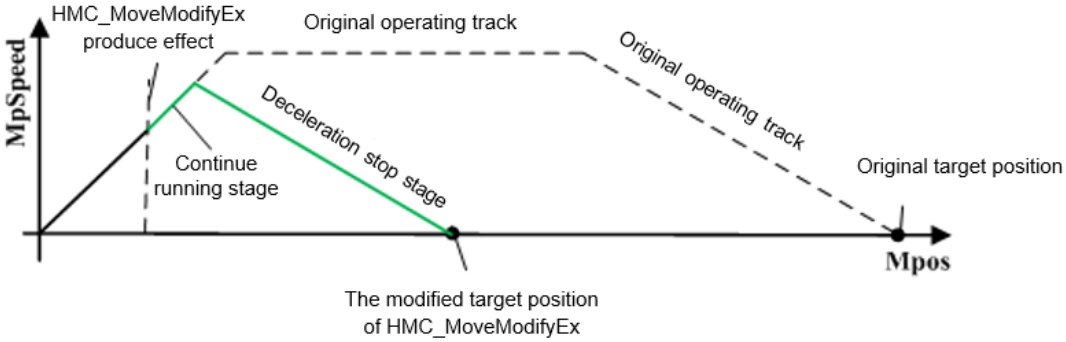
(5) The end position specified by HMC_MoveModifyEx is before the original target position.



Set Position Before Original Target Position (HMC_MoveModifyEx is executed during the constant speed stage)

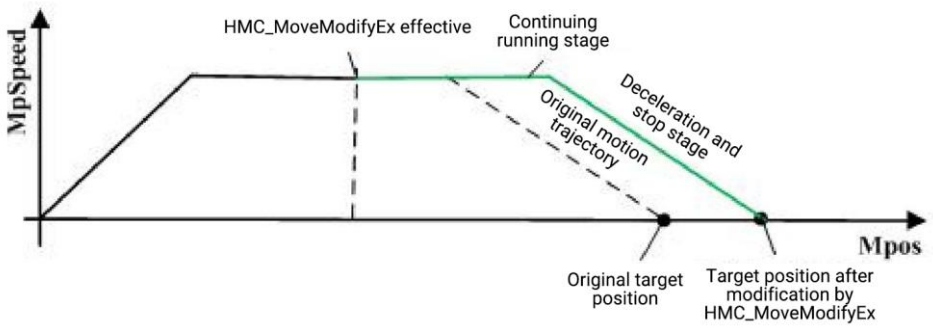


Set Position Before Original Target Position (HMC_MoveModifyEx is executed during the acceleration stage, and there is a constant speed section)

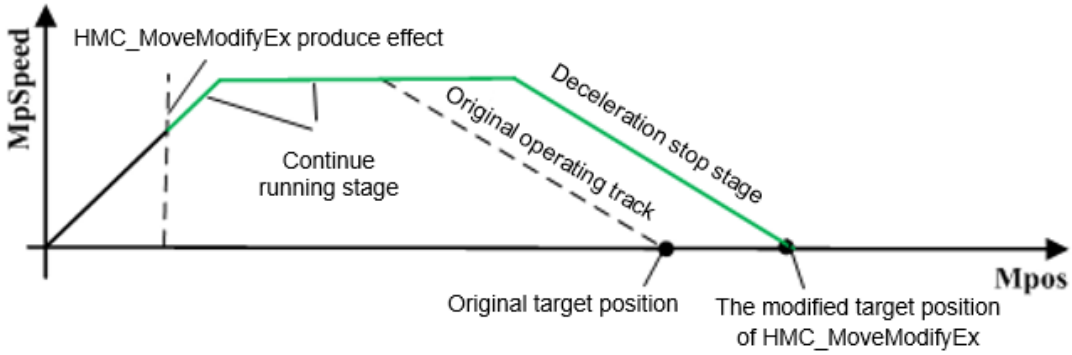


Set Position Before Original Target Position (HMC_MoveModifyEx is executed during the acceleration stage, with no constant speed section)

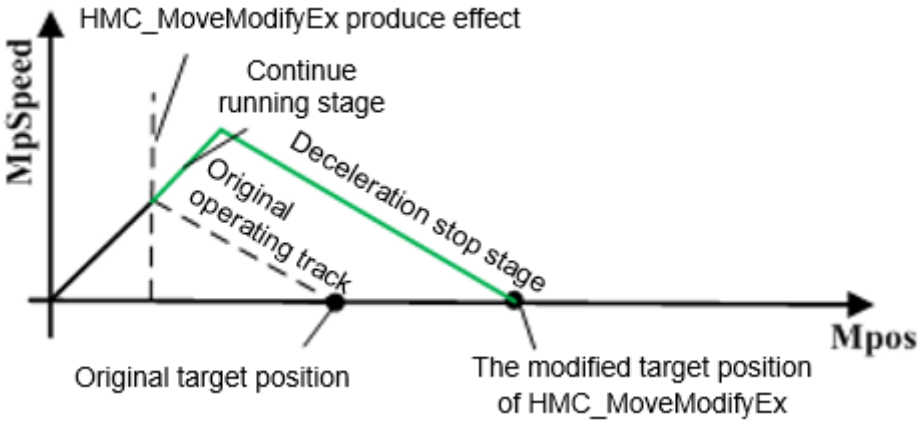
(b) The end position specified by HMC_MoveModifyEx is after the original target position.



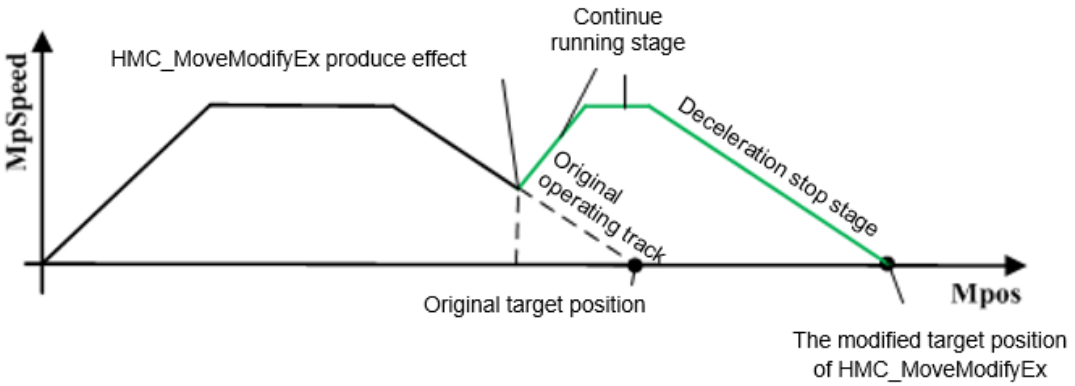
Set Position After Original Target Position (HMC_MoveModifyEx is executed during the constant speed stage)



Set Position After Original Target Position (HMC_MoveModifyEx is executed during the acceleration stage, and there is a constant speed section)



Set Position Before Original Target Position (HMC_MoveModifyEx is executed during the acceleration stage, with no constant speed section)



Set Position After Original Target Position (HMC_MoveModifyEx is executed during the deceleration stage)

Sample program (ST):

p1: =HMC_Move(0, 10000);

p2: = TimeDelay(2);

p3: =HMC_MoveModifyEx(0, 5000, 1); (*The incremental position is 5000 at this time, and there is enough time to slow down*)

5.3.4.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dDpos	Input	LREAL	Modify the displacement of the current instruction
ucModifyMode	Input	USINT	0: Absolute position modification mode 1: Incremental position modification mode
HMC_MoveModify	Output	UDINT	Return value

5.3.4.3.3 Instruction type

It is a non-axis motion cache instruction.

Additional information:

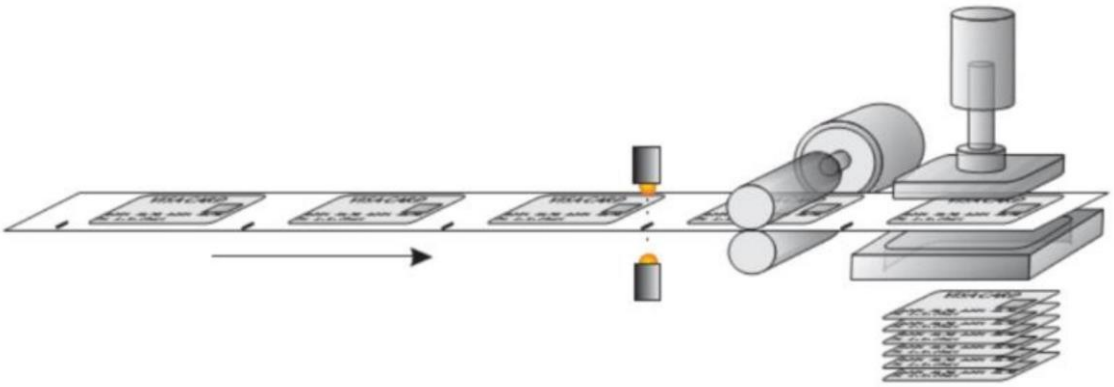
☐ If Merge is currently turned on, Merge will be automatically turned off when the HMC_MoveModifyEx instruction is sent.

5.3.4.3.4 Example

During the operation of the packaging machinery, the mechanical vibration of the conveyor may cause material shift, change in transmission speed, stretch of packaging materials, and there are cumulative errors in the positioning of the color marks on the packaging materials, which may have a great impact on the sealing and cutting quality.

By using the high-speed capture function, together with the correction function for target position changes, compensation is performed in each color mark positioning cycle, which can effectively suppress the cumulative error and control the error within the allowable range. Examples are as follows.

The pinch roller transports a printed card, and a color mark block is printed on the card during printing (the black square in the two patterns in the figure below). It is required to die-cut the card, with the die-cutting position exactly in the center of the card, and there must be no cumulative error. Implementation principle: We install a color mark sensor on the left side of the pinch roller to detect the arrival of the color mark block. When the color mark block arrives, we use the high-speed capture function to capture the precise position of the color mark block and for the distance traveled each time, we use the position of the color mark block as a reference to offset and walk an appropriate distance. Since they are printed from the same printing plate, the positions of color marks and patterns on the card are fixed. In this way, the distance traveled each time is based on the position of the color mark, eliminating accumulated errors. The color mark sensor is connected to the fast DI channel 0, and high-speed capture channel 0 is used to capture the color mark position.



Transmission Diagram

We assume that the distance between the two card patterns is 300 mm, and the position to be die-cut is offset from the color mark (the position can be adjusted). The specific program is as follows:

```
Start_MPOS := 0.0;

End_MPOS := 300.00;

Offset := 30.0;

Axis0TypeReval :=HMC_SetAxisType (Axis0.AxisID, 50); (*Set axis type*)

AxisOUnitReval := HMC_SetUnits(Axis0.AxisID, 1); (*Set user units*)

AxisEnableReval := HMC_DriveEnable(1); (*Turn on servo enable*)

axis0. Speed := 100;

axis0. Acce1 :=1000;

axis0. Dece1 := 1000;
```

WHILE true DO (*Subsequent program calculation, cycle scanning*)

IF bStart = 1 THEN (*Press Run button*)

Axis0DefReval := HMC_Def0rigin(Axis0.AxisID, Axis0.MPos); (*Clear current position*)

Config_result := HMC_CaptureConfig(0, 0, 1, 0, 0, 1, Start_MPOS, End_MPOS); (*Configure high-speed capture*)

Axis0moveReval :=HMC_Move (0, 300); (*Move for one card pattern distance*)

(*Read the status of high-speed capture channel 0 and wait for the high-speed capture to complete. When the color mark sensor has a signal, high-speed capture will be completed, and the precise position of the color mark sensor can be obtained.)

WHILE (HMC_CaptureGetState (0)<>3) DO

Reg_pos := HMC_CaptureGetMpos (0);

END_WHILE

Axis0ModifyReval := HMC_MoveModifyEx (0, offset,1);

(*By capturing the precise position at high speed, the position of the color mark sensor is used as a reference to correct the displacement to ensure that the center of the card can be accurately positioned. For example, if the captured position is 280 and the Offset is 30, then the compensated absolute target position is 310, which compensates for an error of 10*)

Idlereval: = HMC_WaitAxisIdle(0); (*Wait for MoveModify to complete compensation before proceeding to the next cycle*)

(****Die-cut card (omitted)****)

END_IF

END_WHILE

Conveyor trajectory:

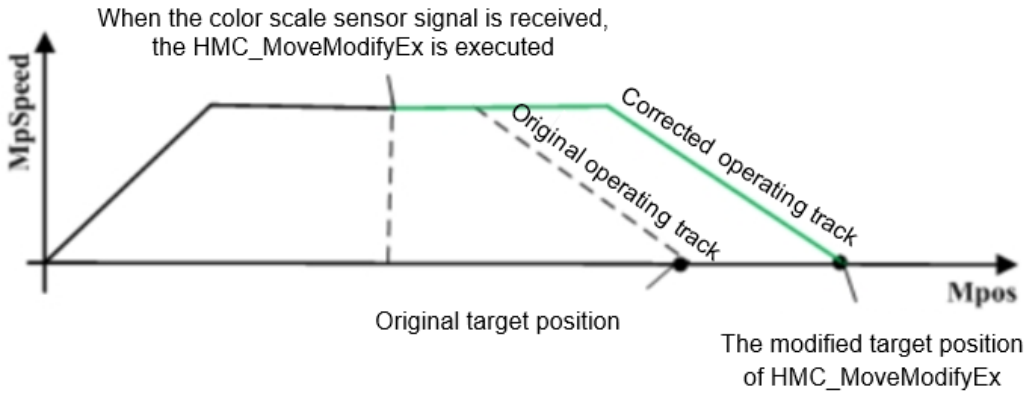


Diagram of Conveyor Trajectory

5.4 Multi-Axis Motion Instructions

5.4.1 Linear Interpolation

5.4.1.1 Two-Axis Linear Interpolation

5.4.1.1.1 HMC_Move2D (Two-axis linear interpolation)

5.4.1.1.1.1 Function

Interpolation function where the motion profile on the two-dimensional plane is a straight line. Taking the execution time of this instruction as the starting point and the relative position of the two split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target position of this instruction is an incremental motion based on the end position of the last motion, that is, each step is an incremental mode motion.

The target position of the interpolation combined axis is, based on the position at the start of the instruction execution, a position after adding the length of the hypotenuse of the right-angled triangle with the incremental position of the two split axes as the right-angled side. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the combined axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

The target position of the interpolating split axis is, based on the position at the start of the instruction execution, a position that is incremented by the relative motion distance specified by the axis. The running speed, acceleration, and deceleration of the interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the split axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

5.4.1.1.1.2 Parameter

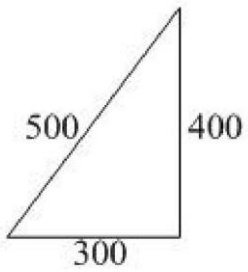
Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
dDistanceX	Input	LREAL	Motion distance of axis X
dDistanceY	Input	LERAL	Motion distance of axis Y
HMC_Move2D	Output	UDINT	Return value

5.4.1.1.1.3 Instruction type

Axis motion cache instructions.

5.4.1.1.1.4 Example

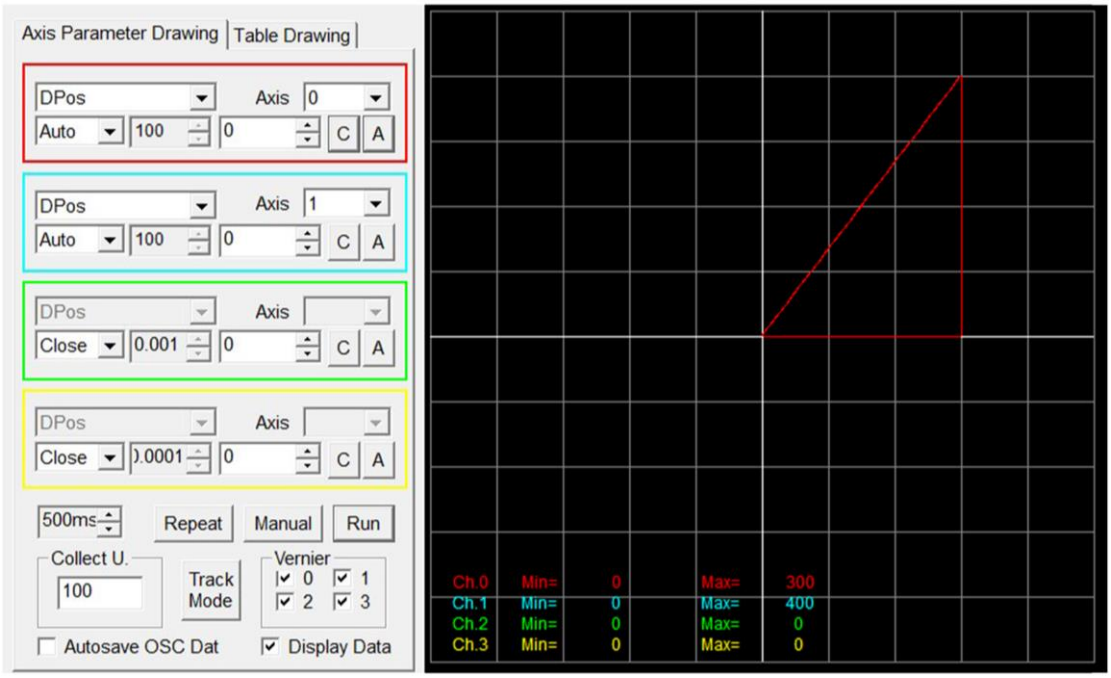
The following example uses the HMC_Move2D instruction to implement the interpolation function where the motion profile on the two-dimensional plane is a straight line. Take a material whose plane cutting outline is a triangle as an example, as shown below:



Triangle Profile

```
(*Set the axis types of the three axes used separately*)
HMC_SetAxisType(10,0);
HMC_SetAxisType(0,50);
HMC_SetAxisType(1,50);
 HMC_Move2D(10,0,1,300,0); (*The X-axis moves 300 positively, the Y-axis is at rest. The first edge of right angle*)
HMC_Move2D(10,0,1,0,400); (*The X-axis is at rest, the Y-axis moves 400 positively. The second edge of right angle*)
HMC_Move2D(10,0,1,-300,-400);  (*The X-axis moves -300, the Y-axis moves -400. Bevel*)
```

The actual running trajectory observed through the AT oscilloscope is as follows:



Triangle Profile Trajectory Realized By Two-axis Linear Interpolation

5.4.1.1.2 HMC_Move2DEx (Advanced two-axis linear interpolation)

5.4.1.1.2.1 Function

Interpolation function where the motion profile on the two-dimensional plane is a straight line. Taking the

execution time of this instruction as the starting point and the relative position of the two split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target position of this instruction is an incremental motion based on the end position of the last motion, that is, each step is an incremental mode motion.

5.4.1.1.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
dDistanceX	Input	LREAL	Motion distance of axis X
dDistanceX	Input	LERAL	Motion distance of axis Y
dSpeed	Input	LREAL	Combined axis motion speed
HMC_Move2DEx	Output	UDINT	Return value

5.4.1.1.2.3 Instruction type

Axis motion cache instruction

5.4.1.1.2.4 Example

The following example uses the HMC_Move2DEx instruction to implement the advanced two-axis linear interpolation. Before this instruction is executed, the axis parameter Speed= function parameter dSpeed of the combined axis is automatically set, and the combined axis calculates using this speed as the target speed. For details, please refer to the function description section of HMC_Move2D (Two-axis linear interpolation).

```
Axis10.Speed := 1000;  (*Set the speed of motion*)

HMC_Move2DEx(10，0，1，30000, 40000,2000); (*Delivery HMC_Move2DEx, coaxial axis10 will run at maximum speed 2000 (not 1000)*)
```

5.4.1.1.3 HMC_MoveAbs2D (Two-axis absolute linear interpolation)

5.4.1.1.3.1 Function

Interpolation function where the motion profile on the two-dimensional plane is a straight line. Taking the execution time of this instruction as the starting point and the absolute position of the two split axes set by the instruction input parameters as the endpoint, the line connecting the starting point and the endpoint is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

5.4.1.1.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
dDposX	Input	LREAL	Motion distance of axis X
dDposY	Input	LERAL	Motion distance of axis Y
dSpeed	Input	LREAL	Combined axis motion speed
HMC_Move2DEx	Output	UDINT	Return value

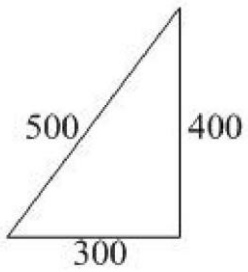
5.4.1.1.3.3 Instruction type

Axis motion cache instruction

5.4.1.1.3.4 Example

The following example uses the HMC_MoveAbs2D instruction to implement the two-axis absolute linear interpolation. Taking the execution time of this instruction as the starting point and the absolute position of the two split axes set by the instruction input parameters as the endpoint, the line connecting the starting point and the endpoint is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

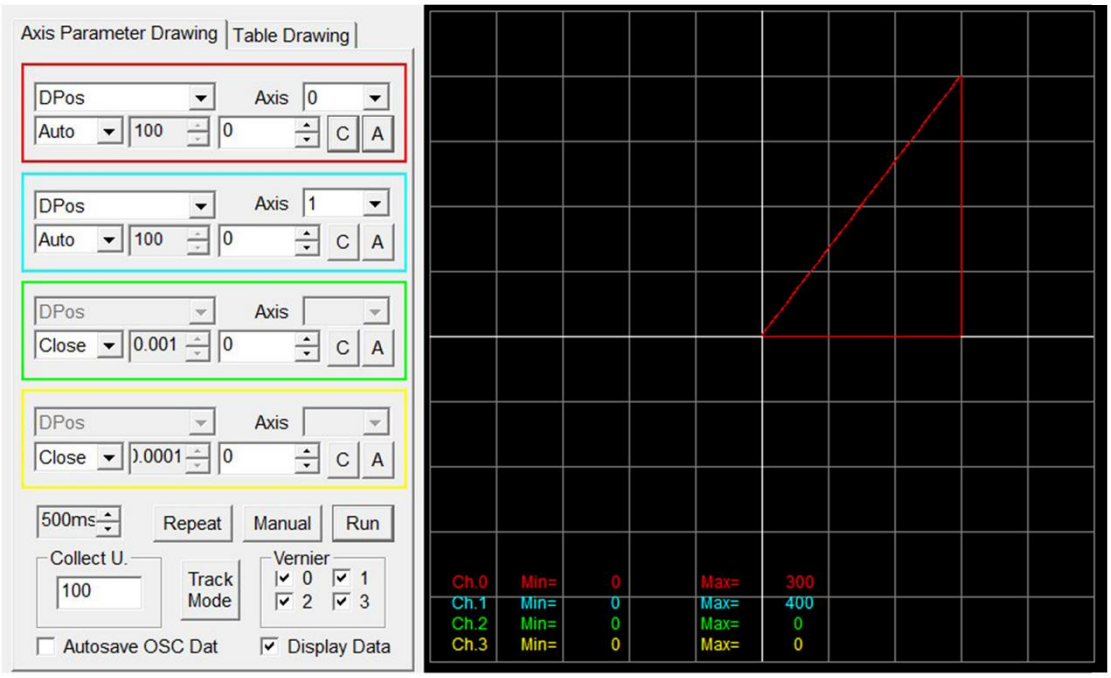
Take a material whose plane cutting outline is a triangle as an example, as shown below:



Triangle Profile

```
(*Set the axis types of the three axes used separately*)
HMC_SetAxisType(10,0);
HMC_SetAxisType(0,50);
HMC_SetAxisType(1,50);
HMC_Move2D(10,0,1,300,0);     (*The X-axis and Y-axis move to (300,0). The first edge of right angle*)
HMC_Move2D(10,0,1,300,400); (*The X-axis and Y-axis move to (300,400). The second edge of right angle*)
HMC_Move2D(10,0,1,0,0);       (*The X-axis and Y-axis move to (0,0). Bevel*)
```

The actual running trajectory observed through the AT oscilloscope is as follows:



Triangle Profile Trajectory Realized By Two-axis Linear Interpolation

5.4.1.1.4 HMC_MoveAbs2DEx (Advanced two-axis absolute linear interpolation)

5.4.1.1.4.1 Function

Interpolation function where the motion profile on the two-dimensional plane is a straight line. Taking the execution time of this instruction as the starting point and the absolute position of the two split axes set by the instruction input parameters as the endpoint, the line connecting the starting point and the endpoint is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

5.4.1.1.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
dDposX	Input	LREAL	Motion distance of axis X
dDposY	Input	LERAL	Motion distance of axis Y
dSpeed	Input	LREAL	Combined axis motion speed
HMC_Move2DEx	Output	UDINT	Return value

5.4.1.1.4.3 Instruction type

Axis motion cache instruction

5.4.1.1.4.4 Example

The following example uses the HMC_MoveAbs2DEx instruction to implement the advanced two-axis absolute linear interpolation. Before this instruction is executed, the axis parameter Speed= function parameter dSpeed of the combined axis is automatically set, and the combined axis calculates using this speed as the target speed. For details, please refer to the function description section of HMC_MoveAbs2D(2Two-axis linear interpolation.)

```
Axis10.Speed := 1000; (*Set the speed of motion*)
HMC_MoveAbs2DEx(10，0，1，30000, 40000,2000);(*Delivery HMC_MoveAbs2DEx, coaxial axis10 will run at maximum speed 2000 (not 1000)*)
```

5.4.1.2 Three-Axis Linear Interpolation

5.4.1.2.1 HMC_Move3D (Three-axis linear interpolation)

5.4.1.2.1.1 Function

Interpolation function in which the motion contour is a straight line in a three-dimensional space. Taking the execution time of this instruction as the starting point and the relative position of the three split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target position of this instruction is an incremental motion based on the end position of the last motion, that is, each step is an incremental mode motion.

The target position of the interpolation combined axis is, based on the position at the start of the instruction execution, a position that is added with the length of the line from the starting point to the end point. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the combined axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

5.4.1.2.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
ucAxisX	INPUT	USINT	Axis number of interpolation Axis X
ucAxisY	INPUT	USINT	Axis number of interpolation axis Y
ucAxisZ	INPUT	USINT	Axis number of interpolation Axis Z
dDistanceX	INPUT	LREAL	Motion distance of axis X
dDistanceY	INPUT	LREAL	Motion distance of axis Y

Parameter	Statement	Data Type	Description
dDistanceZ	INPUT	LREAL	Motion distance of axis Z
HMC_Move3D	OUTPUT	UDINT	Axis instruction ID

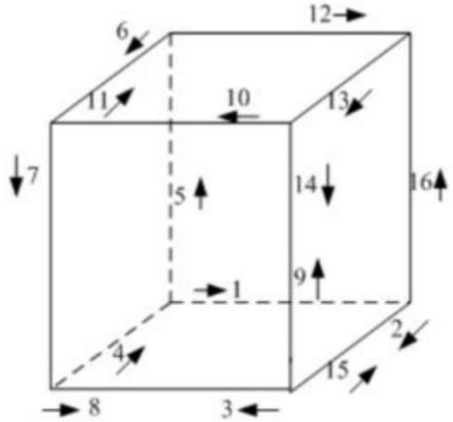
5.4.1.2.1.3 Instruction type

Axis motion cache instructions.

5.4.1.2.1.4 Example

Application example:

Draw a cube with a side length of 1000 in space, allow the same side to be drawn repeatedly, draw the sides in the following order, with the numbers representing the trajectory order and the arrows representing the trajectory direction, and finally draw the cube.



Cube Diagram

Write the program as follows:

(*The starting position is the origin*)

HMC_Move3D (5, 0, 1, 2, 1000, 0, 0); (*Trajectory 1*)

HMC_Move3D (5, 0, 1, 2, 0, 1000, 0); (*Trajectory 2*)

HMC_Move3D (5, 0, 1, 2, -1000, 0, 0); (*Trajectory 3*)

HMC_Move3D (5, 0, 1, 2, 0, -1000, 0); (*Trajectory 4*)

HMC_Move3D (5, 0, 1, 2, 0, 0, 1000); (*Trajectory 5*)

HMC_Move3D (5, 0, 1, 2, 0, 1000, 0); (*Trajectory 6*)

HMC_Move3D (5, 0, 1, 2, 0, 0, -1000); (*Trajectory 7*)

HMC_Move3D (5, 0, 1, 2, 1000, 0, 0); (*Trajectory 8*)

HMC_Move3D (5, 0, 1, 2, 0, 0, 1000); (*Trajectory 9*)

HMC_Move3D (5, 0, 1, 2, -1000, 0, 0); (*Trajectory 10*)

HMC_Move3D (5, 0, 1, 2, 0, -1000, 0); (*Trajectory 11*)

HMC_Move3D (5, 0, 1, 2, 1000, 0, 0); (*Trajectory 12*)

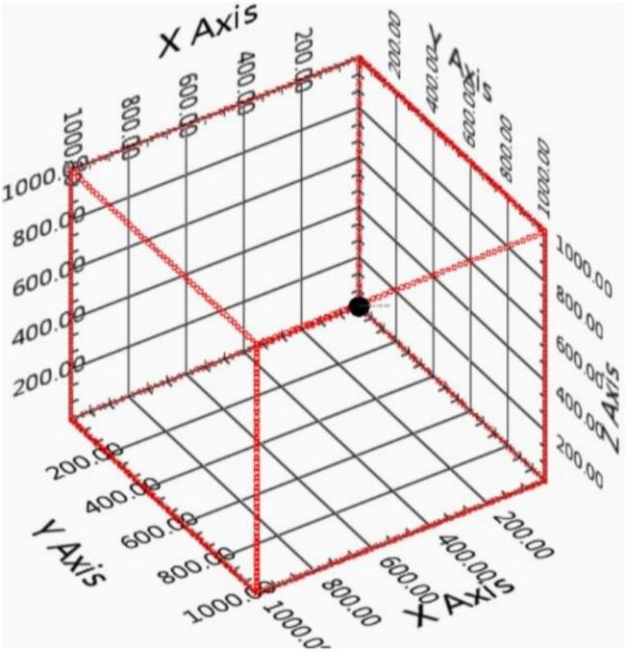
HMC_Move3D (5, 0, 1, 2, 0, 1000, 0); (*Trajectory 13*)

HMC_Move3D (5, 0, 1, 2, 0, 0, -1000); (*Trajectory 14*)

HMC_Move3D (5, 0, 1, 2, 0, -1000, 0); (*Trajectory 15*)

HMC_Move3D (5, 0, 1, 2, 0, 0, 1000); (*Trajectory 16*)

Run the above program. Since the AT oscilloscope currently does not support drawing 3D trajectories, we need to record the Dpos values of the three axes on the oscilloscope, then export the recorded data, and import it in the three-dimensional graphics tool:



3D Diagram

5.4.1.2.2 HMC_Move3DEx (Advanced 3-axis linear interpolation)

5.4.1.2.2.1 Function

Interpolation function in which the motion contour is a straight line in a three-dimensional space. Taking the execution time of this instruction as the starting point and the relative position of the three split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target position of this instruction is an incremental motion based on the end position of the last motion, that is, each step is an incremental mode motion.

This instruction can set the axis parameter Speed= function parameter dSpeed, and the combined axis performs calculation with this speed as the target speed.

5.4.1.2.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
ucAxisX	INPUT	USINT	Axis number of interpolation Axis X
ucAxisY	INPUT	USINT	Axis number of interpolation axis Y
ucAxisZ	INPUT	USINT	Axis number of interpolation Axis Z
dDistanceX	INPUT	LREAL	Motion distance of axis X
dDistanceY	INPUT	LREAL	Motion distance of axis Y
dDistanceZ	INPUT	LREAL	Motion distance of axis Z
dSpeed	INPUT	LREAL	Combined axis target speed
HMC_Move3DEx	OUTPUT	UDINT	Axis instruction ID

5.4.1.2.2.3 Instruction type

Axis motion cache instruction

5.4.1.2.2.4 Example

Axis10.Speed := 1000; (*Set the speed of motion*)

HMC_Move3DEx(10，0，1，2，30000, 40000,5000，2000); (*Drop HMC_Move3DEx, axis 10 will run at a maximum speed of 2000 (instead of 1000)*)

5.4.1.2.3 HMC_MoveAbs3D (Three-axisabsolute linear interpolation)

5.4.1.2.3.1 Function

It is an interpolation function in which the motion contour is a straight line on a three-dimensional plane. Taking the execution time of this instruction as the starting point and the relative position of the three split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

5.4.1.2.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
ucAxisX	INPUT	USINT	Axis number of interpolation Axis X
ucAxisY	INPUT	USINT	Axis number of interpolation axis Y

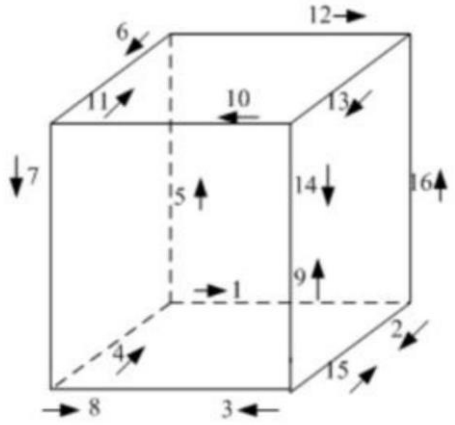
Parameter	Statement	Data Type	Description
ucAxisZ	INPUT	USINT	Axis number of interpolation Axis Z
dDposX	INPUT	LREAL	Axis X target position
dDposY	INPUT	LREAL	Axis Y target position
dDposZ	INPUT	LREAL	Axis Z target position
HMC_MoveAbs3D	OUTPUT	UDINT	Axis instruction ID

5.4.1.2.3.3 Instruction type

Axis motion cache instruction

5.4.1.2.3.4 Example

Draw a cube with a side length of 1000 in space, allow the same side to be drawn repeatedly, draw the sides in the following order, with the numbers representing the trajectory order and the arrows representing the trajectory direction, and finally draw the cube.



Cube Diagram

Write the program as follows:

(*The starting position is the origin*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 0, 0); (*Trajectory 1*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 1000, 0); (*Trajectory 2*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 1000, 0); (*Trajectory 3*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 0, 0); (*Trajectory 4*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 0, 1000); (*Trajectory 5*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 1000, 1000); (*Trajectory 6*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 1000, 0); (*Trajectory 7*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 1000, 0); (*Trajectory 8*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 1000, 1000); (*Trajectory 9*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 1000, 1000); (*Trajectory 10*)

HMC_MoveAbs3D (5, 0, 1, 2, 0, 0, 1000); (*Trajectory 11*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 0, 1000); (*Trajectory 12*)

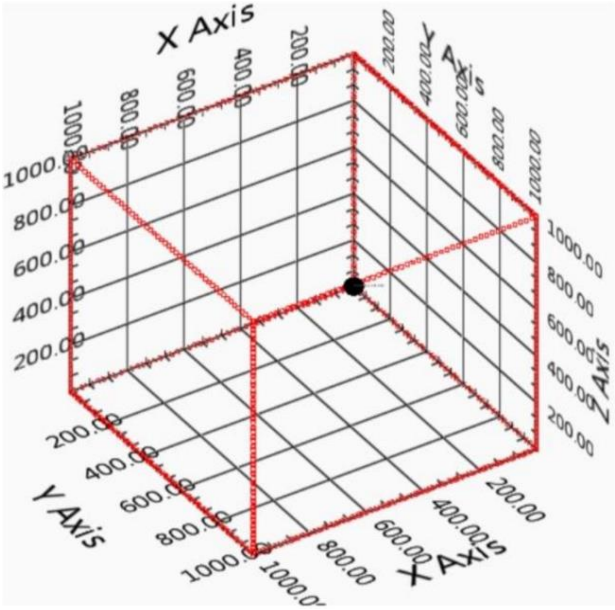
HMC_MoveAbs3D (5, 0, 1, 2, 1000, 1000, 1000); (*Trajectory 13*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 1000, 0); (*Trajectory 14*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 0, 0); (*Trajectory 15*)

HMC_MoveAbs3D (5, 0, 1, 2, 1000, 0, 1000); (*Trajectory 16*)

Run the above program. Since the AT oscilloscope currently does not support drawing 3D trajectories, we need to record the Dpos values of the three axes on the oscilloscope, then export the recorded data, and import it in the three-dimensional graphics tool:



3D Diagram

5.4.1.2.4 HMC_MoveAbs3DEx (Advanced 3-axis absolute linear interpolation)

5.4.1.2.4.1 Function

It is an interpolation function in which the motion contour is a straight line on a three-dimensional plane. Taking the execution time of this instruction as the starting point and the relative position of the three split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

This instruction can set the axis parameter Speed= function parameter dSpeed, and the combined axis performs calculation with this speed as the target speed.

5.4.1.2.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
ucAxisX	INPUT	USINT	Axis number of interpolation Axis X
ucAxisY	INPUT	USINT	Axis number of interpolation axis Y
ucAxisZ	INPUT	USINT	Axis number of interpolation Axis Z
dDposX	INPUT	LREAL	Axis X target position
dDposY	INPUT	LREAL	Axis Y target position
dDposZ	INPUT	LREAL	Axis Z target position
dSpeed	INPUT	LREAL	Combined axis target speed
HMC_MoveAbs3D	OUTPUT	UDINT	Axis instruction ID

5.4.1.2.4.3 Instruction type

Axis motion cache instruction

5.4.1.2.4.4 Example

Refer to the example in HMC_MoveAbs3D(Three-axis absolute linear interpolation).

5.4.1.3 Multi-Axis Linear Interpolation

5.4.1.3.1 HMC_MoveND (N-axis linear interpolation)

5.4.1.3.1.1 Function

Interpolation function in which the motion contour is straight in any N-dimensional coordinate system. Taking the execution time of this instruction as the starting point and the relative position of the linear interpolating split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target position of this instruction is an incremental motion based on the end position of the last motion, that is, each step is an incremental mode motion.

The target position of the interpolation combined axis is, based on the position at the start of the instruction execution, a position that is added with the length of the line from the starting point to the end point. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately.

The target position of the interpolating split axis is, based on the position at the start of the instruction execution, a position that is incremented by the relative motion distance specified by the axis. The operation speed, acceleration, and deceleration of the interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

The split axes of N-axis interpolation instructions are divided into two types: linear interpolation split axes and independent interpolation split axes. The linear interpolating split axis determines the actual trajectory of the combined interpolation motion. The independent interpolating split axis has nothing to do with the combined motion trajectory. The independent interpolating split axis can realize synchronous motion with the combined axis or the split axis. That is, at the end of the instruction, when the combined axis and the linear interpolating split axis complete the instruction motion distance, the independent interpolating split axis also completes the motion distance at the same time.

The motion distance of each linear interpolating split axis determines the motion distance of the combined axis. The motion distance of the combined axis is equal to the square root of the sum of the squares of the motion distances of each linear interpolating split axis. The speed of the linear interpolating split axis is determined by the ratio of the combined axis speed, the motion distance of the linear interpolating split axis, and the motion distance of the combined axis.

The motion distance of the independent interpolating split axis does not affect the motion distance of the combined axis, so it is called an independent axis. It can realize synchronous motion with the combined axis or interpolation linear axis. The speed of the independent axis is determined by the ratio of the combined axis speed, the motion distance of the independent interpolating split axis and the motion distance of the combined axis.

The relationship between the combined motion and the split motion in N-axis linear interpolation is described in detail as follows:

The arrays pAxis and pDistance represent the split axis number and the motion distance of each split axis respectively. LN is the number of linear interpolation split axes (ucLineNum is recorded as LN), and SN is the number of independent interpolation split axes (ucSelfNum is recorded as SN). That is, LN+SN is the total number of split axes, so pAxis and pDistance point to the number of elements of the array respectively, which shall satisfy: $N \geq LN+SN$.

The motion distance of the combined motion axis ucAxisID is D and the speed is Speed;

The motion distance of LN linear interpolation axes is L(0), …, L(LN- 1), and the speed is: Speed L(0), …, Speed L(LN- 1);

The motion distance of SN independent interpolation axes is S(0), ..., S(SN-1), and the speed is: SpeedS(0), ..., SpeedS(SN-1);

The motion distance of the combined axis is determined by the motion distance of the linear interpolating split axis, detailed as follows:

$$D^2 = L(0)^2 + L(1)^2 + \ldots + L(LN-1)^2$$

The relationship between the linear interpolating split axis speed and the combined axis speed is as follows:

$$SpeedL(i) = \frac{L(i)}{D} \times Speed \quad (0 \le i \le LN-1)$$

The relationship between the independent interpolating split axis speed and the combined axis speed is as follows:

$$SpeedS(j) = \frac{S(j)}{D} \times Speed \quad (0 \le j \le SN-1)$$

This instruction does not support the Merge function.

5.4.1.3.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
pAxis	INPUT	POINTER TO USINT	First address of the interpolation split motion axis number array
pDistance	INPUT	POINTER TO LREAL	First address of the interpolation split motion axis motion distance
ucLineNum	INPUT	USINT	Number of linear interpolation axes. The first ucLineNum elements of pAixs are the axis numbers of the linear interpolation split axes. The first ucLineNum elements of pDistance are the motion distances of the linear interpolation split axes. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucLineNum >= 1.
ucSelfNum	INPUT	USINT	Number of independent interpolation axes. The ucSelfNum elements of pAixs after ucLineNum elements are the axis numbers of the independent interpolation split axes, and the ucSelfNum elements of pDistance after ucLineNum elements are the motion distance of the independent interpolating split axis. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucSelfNum>=0.
HMC_MoveND	OUTPUT	UDINT	Axis instruction ID

5.4.1.3.1.3 Instruction type

Axis motion cache instruction

5.4.1.3.1.4 Example

```
FOR i:=0 TO 4 DO
   AxisArry[i] := i;
END_FOR      （* The first 5 elements of the AxisArry array are the axis numbers of the interpolation partition, ranging from 0 to 4 in sequence*）

DistanceArry[0] := 10000;
DistanceArry[1] := 20000;
DistanceArry[2] := 30000;
DistanceArry[3] := 180;
DistanceArry[4] := 120;

HMC_MoveND (15 ,ADR(AxisArry), ADR(DistanceArry), 3, 2);      （* The first three sub axes are linear interpolation sub axes with axis numbers 0, 1, and 2. The next two sub axes are independent interpolation sub axes with axis numbers 3 and 4*）
```

5.4.1.3.2 HMC_MoveNDEx (Advanced N-axis linear interpolation)

5.4.1.3.2.1 Function

Interpolation function in which the motion contour is straight in any N-dimensional coordinate system. Taking the execution time of this instruction as the starting point and the relative position of the linear interpolating split axes set by the instruction input parameters (the coordinate position with the starting point of the instruction as the coordinate origin) as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target position of this instruction is an incremental motion based on the end position of the last motion, that is, each step is an incremental mode motion.

This instruction can set the axis parameter Speed= function parameter dSpeed, and the combined axis performs calculation with this speed as the target speed.

5.4.1.3.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
pAxis	INPUT	POINTER TO USINT	First address of the interpolation split motion axis number array
pDistance	INPUT	POINTER TO LREAL	First address of the interpolation split motion axis motion distance
ucLineNum	INPUT	USINT	Number of linear interpolation axes. The first ucLineNum elements of pAixs are the axis numbers of the linear interpolation split axes. The first ucLineNum elements of pDistance are the motion distances of the linear interpolation split axes. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucLineNum >= 1.
ucSelfNum	INPUT	USINT	Number of independent interpolation axes. The ucSelfNum elements of pAixs after

Parameter	Statement	Data Type	Description
			ucLineNum elements are the axis numbers of the independent interpolation split axes, and the ucSelfNum elements of pDistance after ucLineNum elements are the motion distance of the independent interpolating split axis. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucSelfNum>=0.
dSpeed	INPUT	LREAL	Target speed of combined axis ucAxisID
HMC_MoveND	OUTPUT	UDINT	Axis instruction ID

5.4.1.3.2.3 Instruction type

Axis motion cache instruction

5.4.1.3.2.4 Example

```
FOR i:=0 TO 4 DO
   AxisArry[i] := i;
END_FOR    （* The first 5 elements of the AxisArry array are the axis numbers of the interpolation partition, ranging from 0 to 4 in sequence*）

DistanceArry[0] := 10000;
DistanceArry[1] := 20000;
DistanceArry[2] := 30000;
DistanceArry[3] := 180;
DistanceArry[4] := 120;

HMC_MoveNDEX (15 ,ADR(AxisArry), ADR(DistanceArry), 3, 2,1000);
```

5.4.1.3.3 HMC_MoveAbsND (N-axisabsolute linear interpolation)

5.4.1.3.3.1 Function

Interpolation function in which the motion contour is straight in any N-dimensional coordinate system. Taking the absolute position of the linear interpolating split axis set by the instruction input parameters as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

5.4.1.3.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
pAxis	INPUT	POINTER TO USINT	First address of the interpolation split motion axis number array
pDpos	INPUT	POINTER TO LREAL	First address of the interpolation split motion axis target position
ucLineNum	INPUT	USINT	Number of linear interpolation axes. The first ucLineNum elements of pAixs are the axis

Parameter	Statement	Data Type	Description
			numbers of the linear interpolation split axes. The first ucLineNum elements of pDistance are the motion distances of the linear interpolation split axes. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucLineNum >= 1.
ucSelfNum	INPUT	USINT	Number of independent interpolation axes. The ucSelfNum elements of pAixs after ucLineNum elements are the axis numbers of the independent interpolation split axes, and the ucSelfNum elements of pDistance after ucLineNum elements are the motion distance of the independent interpolating split axis. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucSelfNum>=0.
HMC_MoveND	OUTPUT	UDINT	Axis instruction ID

5.4.1.3.3.3 Instruction type

Axis motion cache instruction

5.4.1.3.3.4 Example

Refer to HMC_MoveND (N-axis linear interpolation).

5.4.1.3.4 HMC_MoveAbsNDEx (Advanced N-axis absolute linear interpolation)

5.4.1.3.4.1 Function

Interpolation function in which the motion contour is straight in any N-dimensional coordinate system. Taking the absolute position of the linear interpolating split axis set by the instruction input parameters as the end point, the line connecting the starting point and the end point is the motion trajectory of the instruction. The target positions of this instruction are all absolute, that is, the position relative to the axis origin.

5.4.1.3.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Interpolation combined motion axis number
pAxis	INPUT	POINTER TO USINT	First address of the interpolation split motion axis number array
pDpos	INPUT	POINTER TO LREAL	First address of the interpolation split motion axis target position
ucLineNum	INPUT	USINT	Number of linear interpolation axes. The first ucLineNum elements of pAixs are the axis numbers of the linear interpolation split axes. The first ucLineNum elements of pDistance are

Parameter	Statement	Data Type	Description
			the motion distances of the linear interpolation split axes. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucLineNum >= 1.
ucSelfNum	INPUT	USINT	Number of independent interpolation axes. The ucSelfNum elements of pAixs after ucLineNum elements are the axis numbers of the independent interpolation split axes, and the ucSelfNum elements of pDistance after ucLineNum elements are the motion distance of the independent interpolating split axis. From front to back, they correspond to the axis numbers specified by pAxis, with the value range: ucSelfNum>=0.
dSpeed	INPUT	LREAL	Target speed of combined axis ucAxisID
HMC_MoveND	OUTPUT	UDINT	Axis instruction ID

5.4.1.3.4.3 Instruction type

Axis motion cache instruction

5.4.1.3.4.4 Example

Refer to HMC_MoveND (N-axis linear interpolation).

5.4.2 Arc Interpolation

5.4.2.1 Planar Arc Interpolation

5.4.2.1.1 HMC_MoveCircle (Arc interpolation)

5.4.2.1.1.1 Function

It is an interpolation function in which the motion contour is an arc on a two-dimensional plane.

The target position of the interpolation combined axis is, based on the position at the start of the instruction execution, a position that is added with the arc length of the instruction trajectory. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the combined axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately. The operation speed, acceleration, and deceleration of the interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

5.4.2.1.1.2 Parameter

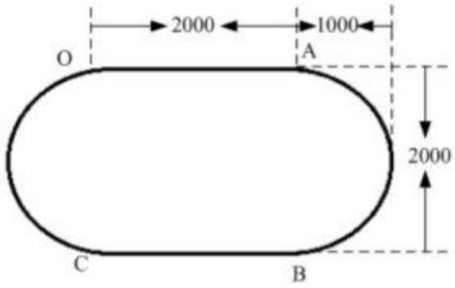
Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
ucMode (O is the current point)	Input	USINT	0: A is the end point, and B is an arc at a certain point in the middle; when O, B, and A are three points on a line and B is in the middle, it is treated as a straight line 1: A is the end point, B is the center of the arc, and it moves in the counterclockwise direction 3: A is the end point, B is the center of the arc, and it moves in the clockwise direction
dAx	Input	LREAL	The position of point A on the X-axis - the position of point O on the X-axis (X coordinate of A when O is regarded as the origin of the coordinate system)
dAy	Input	LREAL	The position of point A on the Y-axis - the position of point O on the Y-axis (Y coordinate of A when O is regarded as the origin of the coordinate system)
dBx	Input	LREAL	The position of point B on the X-axis - the position of point O on the X-axis (X coordinate of B when O is regarded as the origin of the coordinate system)
dBy	Input	LREAL	The position of point B on the Y-axis - the position of point O on the Y-axis (Y coordinate of B when O is regarded as the origin of the coordinate system)
HMC_MoveCircle	Output	UDINT	Axis instruction ID

5.4.2.1.1.3 Instruction type

Axis motion cache instructions.

5.4.2.1.1.4 Example

Cut the following workpiece.



Workpiece Size Drawing

Taking point O in the above figure as the starting point, write the program as follows:

HMC_Move2D (6, 0, 1, 2000, 0);

HMC_MoveCircle(6, 0, 1, 3, 0, -2000, 0, -1000); (*The current point A is the starting point, and both the end point and the center of the circle have point A as the origin point*)

HMC_Move2D(6, 0, 1, -2000, 0);

HMC_MoveCircle(6, 0, 1, 3, 0, 2000, 0, 1000); (*The current point C is the starting point, and both the end point and the center of the circle have point C as the origin point*)

Run the program and get the trajectory on the AT oscilloscope as follows:

5.4.2.1.2 HMC_MoveCircleEx (Advanced arc interpolation)

5.4.2.1.2.1 Function

It is an interpolation function in which the motion contour is an arc on a two-dimensional plane.

The target position of the interpolation combined axis is, based on the position at the start of the instruction

execution, a position that is added with the arc length of the instruction trajectory. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately. The operation speed, acceleration, and deceleration of the interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

This instruction can set the axis parameter Speed= function parameter dSpeed, and the combined axis performs calculation with this speed as the target speed.

5.4.2.1.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
ucMode (O is the current point)	Input	USINT	0: A is the end point, and B is an arc at a certain point in the middle; when O, B, and A are three points on a line and B is in the middle, it is treated as a straight line 1: A is the end point, B is the center of the arc, and it moves in the counterclockwise direction 3: A is the end point, B is the center of the arc, and it moves in the clockwise direction
dAx	Input	LREAL	The position of point A on the X-axis - the position of point O on the X-axis (X coordinate of A when O is regarded as the origin of the coordinate system)
dAy	Input	LREAL	The position of point A on the Y-axis - the position of point O on the Y-axis (Y coordinate of A when O is regarded as the origin of the coordinate system)
dBx	Input	LREAL	The position of point B on the X-axis - the position of point O on the X-axis (X coordinate of B when O is regarded as the origin of the coordinate system)
dBy	Input	LREAL	The position of point B on the Y-axis - the position of point O on the Y-axis (Y coordinate of B when O is regarded as the origin of the coordinate system)

Parameter	Statement	Data Type	Description
dSpeed	Input	LREAL	Combined axis target speed
HMC_MoveCircle	Output	UDINT	Axis instruction ID

5.4.2.1.2.3 Instruction type

Axis motion cache instructions.

5.4.2.1.2.4 Example

Refer to HMC_MoveCircle (Arc interpolation).

5.4.2.2 Spherical Interpolation

5.4.2.2.1 HMC_MoveSpherical (Spherical arc interpolation)

5.4.2.2.1.1 Function

It is an interpolation function in which the motion contour is a spherical arc in a three-dimensional space. A spherical arc is an arc on a certain plane in space, or an arc on a space circle where the spherical surface intersects a certain plane.

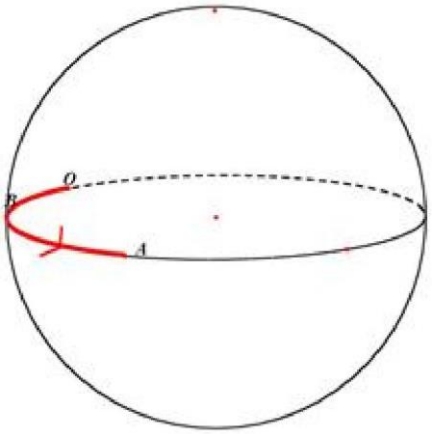
The target position of the interpolation combined axis is, based on the position at the start of the instruction execution, a position that is added with the spherical arc length of the instruction trajectory. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately. The operation speed, acceleration, and deceleration of the interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

This instruction supports four modes:

(1) ucMode=0

Taking the instruction start time as the starting point O, B as the middle point, and A as the end point, these three points determine a spherical arc. The coordinates of points A and B are based on point O as the origin of the coordinates. If points O, B, and A are on a straight line and point B is the middle point, it will be processed as a straight line.

The following is a spherical arc in space determined by three points O, B, and A. The motion direction is O->B->A. The red curve is the motion trajectory of the spherical arc, and the arrow indicates the direction.

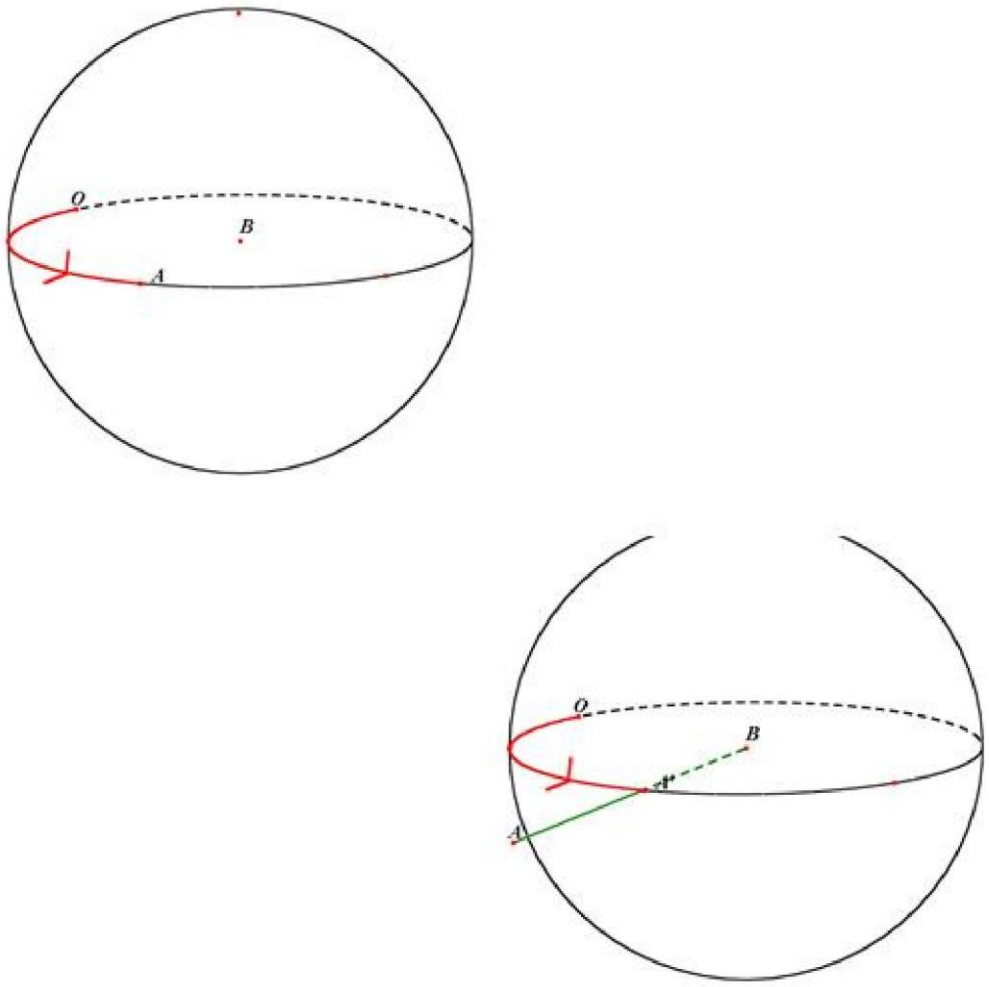


Motion Trajectory

(2) ucMode= 1

Taking the start time of the instruction as the starting point O, B as the center of the space arc, and A as the end point, determine a space arc based on the shortest distance from O to A as the motion direction.

The figure below is a spherical arc in space with O as the starting point, B as the center of the spherical arc, and A as the end point. When point A is not on the spherical arc, the intersection between line AB and the spherical arc is the end point. The red curve is the motion trajectory of the spherical arc, and the arrow indicates the direction.

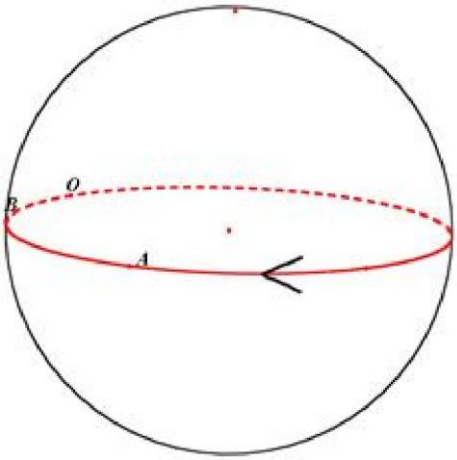


Motion Trajectory

(3) ucMode=2

Taking the instruction start time as the starting point O, and A and B as the intermediate points, with the direction O->A->B, complete an arc motion.

The following is the spherical arc in space determined by three points O, B, and A. The motion direction is O->A->B. The red curve is the motion trajectory of the complete spherical arc determined by the three points, and the arrow indicates the direction.

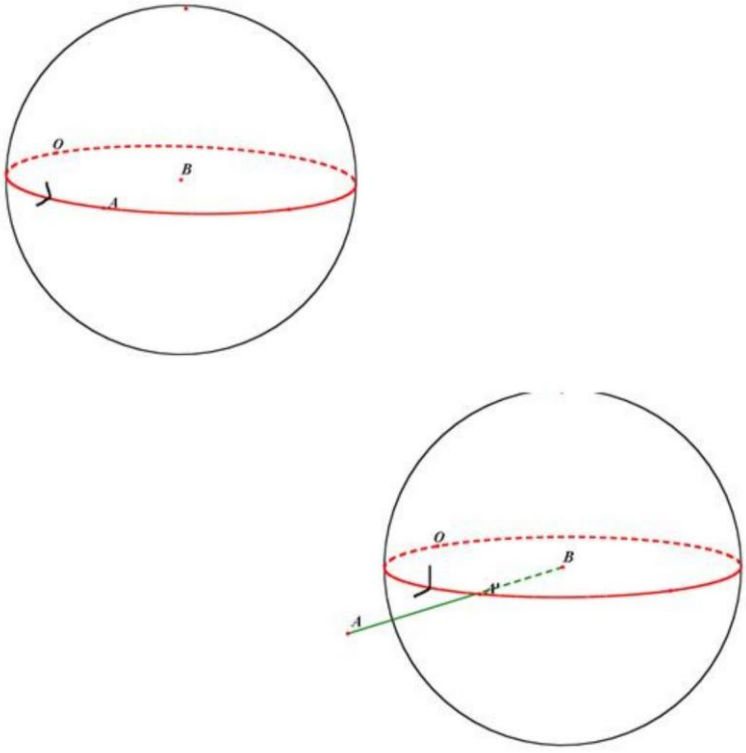


Motion Trajectory

(4) ucMode=3

Taking the start time of the instruction as the starting point O, B as the center of the space arc, and A as the intermediate point, to finish the complete arc based on the shortest distance from O to A as the motion direction.

The figure below is a spherical arc in space with O as the starting point, B as the center point of the spherical arc, and A as the intermediate point, which moves for the shortest distance from O to A, namely, in the counterclockwise direction. The red curve is the motion trajectory of the spherical arc, and the arrow indicates the direction.



Motion Trajectory

5.4.2.2.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
ucAxisZ	Input	USINT	Axis number of interpolation Axis Z
ucMode (O is the current point)	Input	USINT	0: A is the end point, and B is an arc at a certain point in the middle; when O, B, and A are three points on a line and B is the intermediate point, it is treated as a straight line 1: With A as the end point and B as an arc at the center of the space arc, the system determines the direction with the following method: making the distance from O to A shortest (the system automatically handles special cases such as equal distance). Note: The three points O, B and A cannot be colinear during user settings. 2: With A and B as intermediate points, move for a complete arc in the direction ->A->B 3: With A as an intermediate point and B as the center of the space arc to finish the complete arc; the system determines the direction with the following method: making the distance from O to A shortest (the system automatically handles special cases such as equal distance). Note: The three points O, B and A cannot be colinear during user settings.
dAx	Input	LREAL	The position of point A on the X-axis - the position of point O on the X-axis (X coordinate of A when O is regarded as the origin of the coordinate system)
dAy	Input	LREAL	The position of point A on the Y-axis - the position of point O on the Y-axis (Y coordinate of A when O is regarded as the origin of the coordinate system)
dAZ	Input	LREAL	The position of point A on the Z-axis - the position of point O on the Z-axis (Z coordinate of A when O is regarded as the origin of the coordinate system)
dBx	Input	LREAL	The position of point B on the X-axis - the position of point O on the X-axis (X coordinate of B when O is regarded as the origin of the coordinate system)
dBy	Input	LREAL	The position of point B on the Y-axis - the position of point O on the Y-axis (Y coordinate of B when O is regarded as the origin of

Parameter	Statement	Data Type	Description
			the coordinate system)
dBZ	Input	LREAL	The position of point B on the Z-axis - the position of point O on the Z-axis (Z coordinate of B when O is regarded as the origin of the coordinate system)
HMC_MoveSpherical	Output	UDINT	Axis instruction ID

5.4.2.2.1.3 Instruction type

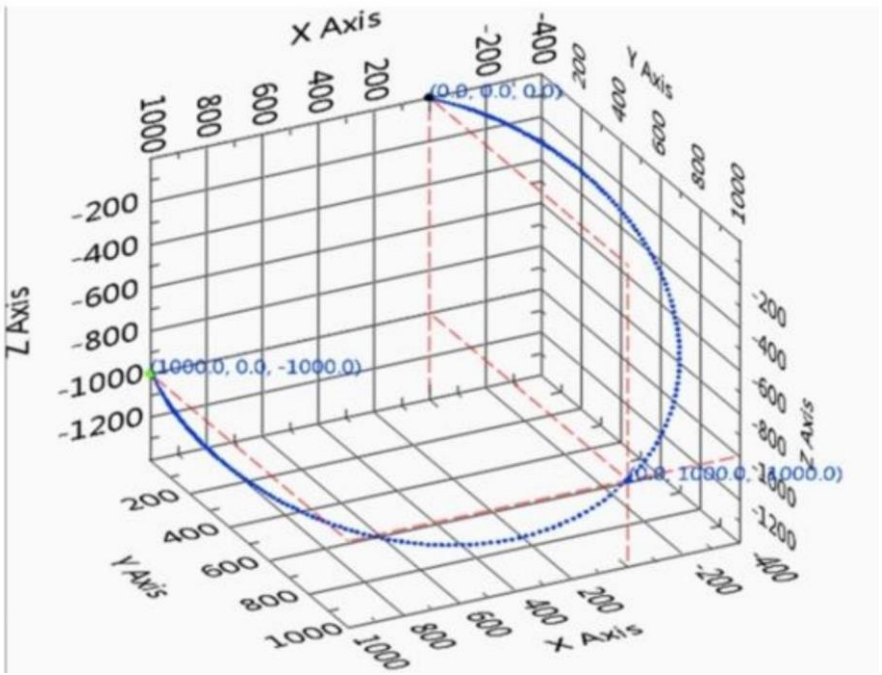
Non-blocking instructions, axis motion cache instructions.

5.4.2.2.1.4 Example

Taking (0, 0, 0) as the starting point, (0, 1000, - 1000) as the intermediate point, and (1000, 0, - 1000) as the end points, using mode 0, write a program as follows:

HMC_MoveSpherical (0, 1,2,3,0, 1000,0,- 1000,0, 1000,- 1000);

The AT oscilloscope records the Dpos of each split axis, exports the data, and reads the data through third-party software to draw the three-dimensional graph as follows:



3D Diagram

5.4.2.2.2 HMC_MoveSphericalEx (Advanced spherical arc interpolation)

5.4.2.2.2.1 Function

It is an interpolation function in which the motion contour is a spherical arc in a three-dimensional space. A spherical arc is an arc on a certain plane in space, or an arc on a space circle where the spherical surface intersects a certain plane.

The target position of the interpolation combined axis is, based on the position at the start of the instruction execution, a position that is added with the spherical arc length of the instruction trajectory. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately. The operation speed, acceleration, and deceleration of the interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

This instruction can set the axis parameter Speed= function parameter dSpeed, and the combined axis performs calculation with this speed as the target speed.

5.4.2.2.2.2 Parameter

Parameter	**Statement**	**Data Type**	**Description**
ucAxisID	Input	USINT	Interpolation combined motion axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
ucAxisZ	Input	USINT	Axis number of interpolation Axis Z
ucMode (O is the current point)	Input	USINT	0: A is the end point, and B is an arc at a certain point in the middle; when O, B, and A are three points on a line and B is the intermediate point, it is treated as a straight line 1: With A as the end point and B as an arc at the center of the space arc, the system determines the direction with the following method: making the distance from O to A shortest (the system automatically handles special cases such as equal distance). Note: The three points O, B and A cannot be colinear during user settings. 2: With A and B as intermediate points, move for a complete arc in the direction O->A->B 3: With A as an intermediate point and B as the center of the space arc to finish the complete arc; the system determines the direction with the following method: making the distance from O to A shortest (the system automatically handles special cases such as equal distance). Note: The three points O, B and A cannot be colinear during user settings.
dAx	Input	LREAL	The position of point A on the X-axis - the position of point O on the X-axis (X coordinate of A when O is regarded as the origin of the coordinate system)
dAy	Input	LREAL	The position of point A on the Y-axis - the position of point O on the Y-axis

Parameter	Statement	Data Type	Description
			(Y coordinate of A when O is regarded as the origin of the coordinate system)
dAZ	Input	LREAL	The position of point A on the Z-axis - the position of point O on the Z-axis (Z coordinate of A when O is regarded as the origin of the coordinate system)
dBx	Input	LREAL	The position of point B on the X-axis - the position of point O on the X-axis (X coordinate of B when O is regarded as the origin of the coordinate system)
dBy	Input	LREAL	The position of point B on the Y-axis - the position of point O on the Y-axis (Y coordinate of B when O is regarded as the origin of the coordinate system)
dBZ	Input	LREAL	The position of point B on the Z-axis - the position of point O on the Z-axis (Z coordinate of B when O is regarded as the origin of the coordinate system)
dSpeed	Input	LREAL	Combined axis target speed
HMC_MoveSphericalEx	Output	UDINT	Axis instruction ID

5.4.2.2.2.3 Instruction type

Non-blocking instructions, axis motion cache instructions

5.4.2.2.2.4 Example

Refer to HMC_MoveSpherical (Spherical arc interpolation).

5.4.3 Helical Interpolation

5.4.3.1 HMC_MoveHelical (Helical Interpolation)

5.4.3.1.1 Function

It indicates a helical interpolation motion. ucAxisID is the combined motion axis. Under the coordinate frame O-XYZ, axes ucAxisX and ucAxisY perform planar arc interpolation motion, axis ucAxisZ performs Z-phase linear motion, and the three axes work together to complete the cylindrical helix.

The target position of the interpolation combined axis is calculated in different ways according to different modes. The combined motion speed of interpolation motion is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately. The operation speed, acceleration, and deceleration of the

interpolation motion split axis have nothing to do with the speed parameters Speed, Accel, and Decel set for the axis. The motion speed of the split axis is associated and calculated in real time by the combined axis speed parameter VpSpeed.

This instruction supports six modes:

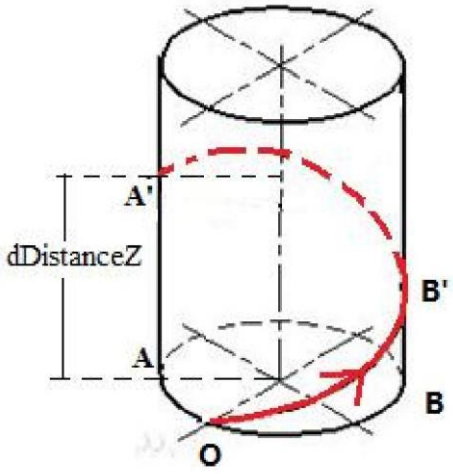
(1) ucMode=0

A planar arc has A as the end point and B as the intermediate point. The combined axis performs the combined motion through axes X, Y and Z. The target position of the combined axis is to increment a hypotenuse length with the length of the planar arc and the length of the Z-axis motion position as the right-angle sides based on the position where the instruction starts. The relationship in position and speed of the combined axis and the split axis is as follows:

$$dDistance\_AxisID^2 = dCircleLength^2 + dDistance\_AxisZ^2$$

$$\frac{dSpeed\_AxisZ}{dSpeed\_AxisID} = \frac{dDistance\_AxisZ}{dDistance\_AxisID}$$

The figure below is the motion diagram in this mode, where the red curve is the spatial trajectory curve of the linked three axes.

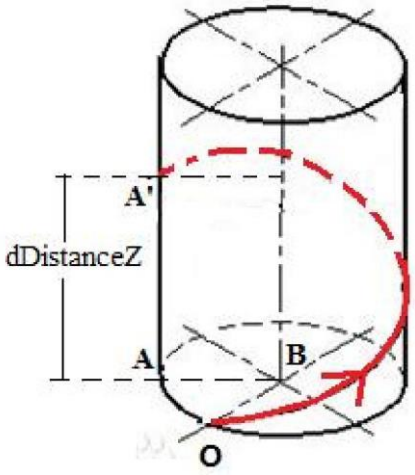


Motion Trajectory

(2) ucMode= 1

The planar arc has B as the center and A as the end point, and moves in the counterclockwise direction. The combined axis performs the combined motion through axes X, Y and Z. The relationship in position and speed between the combined axis and the split axis is consistent with mode 0.

The figure below is the motion diagram in this mode, where the red curve is the spatial trajectory curve of the linked three axes.

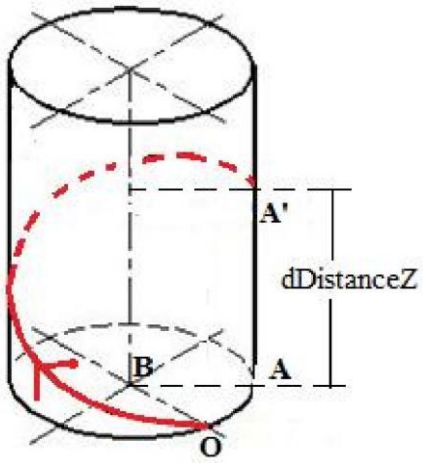


Motion Trajectory

(3) ucMode=3

The planar arc has B as the center and A as the end point, and moves in a clockwise direction. The combined axis performs the combined motion through axes X, Y and Z. The relationship in position and speed between the combined axis and the split axis is consistent with mode 0.

The figure below is the motion diagram in this mode, where the red curve is the spatial trajectory curve of the linked three axes.



Motion Trajectory

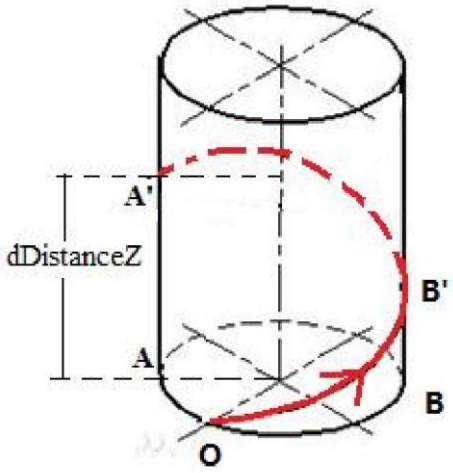
(4) ucMode=4

A planar arc has A as the end point and B as the intermediate point. The combined axis performs the combined motion through axes A and Y. The target position of the combined axis is to increment the planar arc length based on the position where the instruction starts. The relationship in position and speed of the combined axis and the split axis is as follows:

$$dDistance\_AxisID = dCircleLength$$

$$\frac{dSpeed\_AxisZ}{dSpeed\_AxisID} = \frac{dDistance\_AxisZ}{dDistance\_AxisID}$$

The figure below is the motion diagram in this mode, where the red curve is the spatial trajectory curve of the linked three axes.

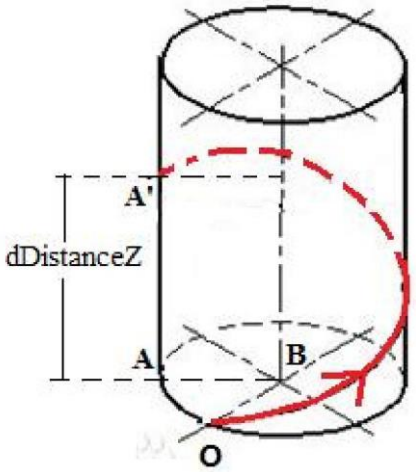


Motion Trajectory

(5) ucMode=5

The planar arc has B as the center and A as the end point, and moves in a counterclockwise direction. The combined axis performs the combined motion through axes X and Y. The relationship in position and speed between the combined axis and the split axis is consistent with mode 4.

The figure below is the motion diagram in this mode, where the red curve is the spatial trajectory curve of the linked three axes.

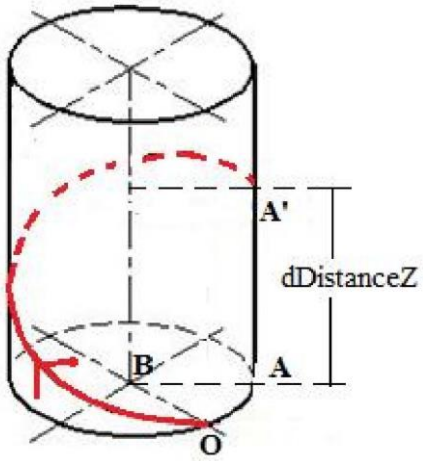


Motion Trajectory

(6) ucMode=7

The planar arc has B as the center and A as the end point, and moves in a clockwise direction. The combined axis performs the combined motion through axes X and Y. The relationship in position and speed between the combined axis and the split axis is consistent with mode 4.

The figure below is the motion diagram in this mode, where the red curve is the spatial trajectory curve of the linked three axes.



Motion Trajectory

5.4.3.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
ucAxisZ	Input	USINT	Axis number of interpolation Axis Z
ucMode (O is the current point)	Input	USINT	0: A is the end point, and B is an arc at a certain point in the middle; the speed dSpeed of ucAxisID is the speed of the combined motion of the three axes. Note: The three points O, B and A cannot be colinear during user settings. 1: A is the end point, B is the center of the arc, and it moves in the counterclockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the three axes. 3: A is the end point, B is the center of the arc, and it moves in the clockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the three axes. 4: A is the end point, and B is an arc at a certain point in the middle; the speed dSpeed of ucAxisID is the speed of the combined motion of the axes X and Y. Note: The three points O, B and A cannot be colinear during user settings. 5: A is the end point, B is the center of the arc, and it moves in the counterclockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the axes X

Parameter	Statement	Data Type	Description
			and Y 7: A is the end point, B is the center of the arc, and it moves in the clockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the axes X and Y
dAx	Input	LREAL	The position of point A on the X-axis - the position of point O on the X-axis (X coordinate of A when O is regarded as the origin of the coordinate system)
dAy	Input	LREAL	The position of point A on the Y-axis - the position of point O on the Y-axis (Y coordinate of A when O is regarded as the origin of the coordinate system)
dBx	Input	LREAL	The position of point B on the X-axis - the position of point O on the X-axis (X coordinate of B when O is regarded as the origin of the coordinate system)
dBy	Input	LREAL	The position of point B on the Y-axis - the position of point O on the Y-axis (Y coordinate of B when O is regarded as the origin of the coordinate system)
dDistanceZ	Input	LREAL	Z-axis displacement
HMC_MoveHelical	Output	UDINT	Axis instruction ID

5.4.3.1.3 Instruction type

Non-blocking instructions, axis motion cache instructions

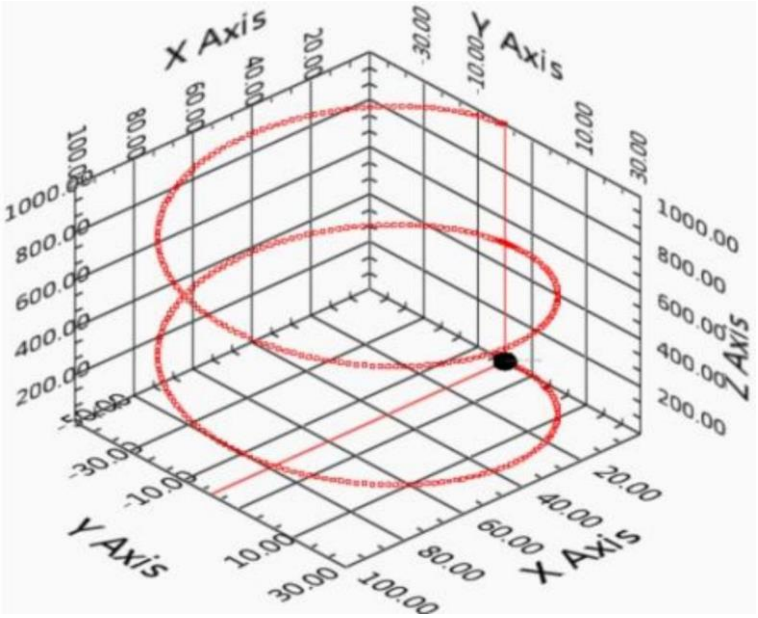
5.4.3.1.4 Example

The thread milling motion trajectory is a helix, which can be realized through the three-axis linkage of CNC machine tools. The workpiece performs arc interpolation of the X and Y axes on the horizontal plane and linear motion of the Z-axis in the vertical direction. It completes a full circular motion in the XOY plane, that is, after rotating 360 degrees along the threaded hole, it rises by one pitch in the Z-axis direction. Threads can be milled using helical interpolation.

For example, process a thread with a diameter of 100 mm, a pitch of 500 mm, and a height of 1000 mm.

```
FOR i:=1 TO 2 BY 1 DO
   HMC_MoveHelical (0, 1, 2, 3, 7, 0, 0, 50, 0, 500);
END_FOR
```

The AT oscilloscope records the Dpos of each split axis, exports the data, and reads the data through third-party software to draw the three-dimensional graph:



3D Diagram

5.4.3.2 HMC_MoveHelicalEx (Advanced Helical Interpolation)

5.4.3.2.1 Function

It indicates a helical interpolation motion. ucAxisID is the combined motion axis. Under the coordinate frame O-XYZ, axes ucAxisX and ucAxisY perform planar arc interpolation motion, axis ucAxisZ performs Z-phase linear motion, and the three axes work together to complete the cylindrical helix.

This instruction can set the axis parameter Speed= function parameter dSpeed, and the combined axis performs calculation with this speed as the target speed.

5.4.3.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
ucAxisX	Input	USINT	Axis number of interpolation Axis X
ucAxisY	Input	USINT	Axis number of interpolation axis Y
ucAxisZ	Input	USINT	Axis number of interpolation Axis Z
ucMode (O is the current point)	Input	USINT	0: A is the end point, and B is an arc at a certain point in the middle; the speed dSpeed of ucAxisID is the speed of the combined motion of the three axes. Note: The three points O, B and A cannot be colinear during user settings. 1: A is the end point, B is the center of the arc, and it moves in the counterclockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the three axes. 3: A is the end point, B is the center of the arc, and it moves in the clockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the three axes.

Parameter	Statement	Data Type	Description
			4: A is the end point, and B is an arc at a certain point in the middle; the speed dSpeed of ucAxisID is the speed of the combined motion of the axes X and Y. Note: The three points O, B and A cannot be colinear during user settings. 5: A is the end point, B is the center of the arc, and it moves in the counterclockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the axes X and Y 7: A is the end point, B is the center of the arc, and it moves in the clockwise direction; the speed dSpeed of ucAxisID is the speed of the combined motion of the axes X and Y
dAx	Input	LREAL	The position of point A on the X-axis - the position of point O on the X-axis (X coordinate of A when O is regarded as the origin of the coordinate system)
dAy	Input	LREAL	The position of point A on the Y-axis - the position of point O on the Y-axis (Y coordinate of A when O is regarded as the origin of the coordinate system)
dBx	Input	LREAL	The position of point B on the X-axis - the position of point O on the X-axis (X coordinate of B when O is regarded as the origin of the coordinate system)
dBy	Input	LREAL	The position of point B on the Y-axis - the position of point O on the Y-axis (Y coordinate of B when O is regarded as the origin of the coordinate system)
dDistanceZ	Input	LREAL	Z-axis displacement
dSpeed	Input	LREAL	Target speed of combined axis ucAxisID
HMC_MoveHelical	Output	UDINT	Axis instruction ID

5.4.3.2.3 Instruction type

Non-blocking instructions, axis motion cache instructions

5.4.3.2.4 Example

Refer to the example in HMC_MoveHelical (Helical interpolation).

5.4.4 Spline Interpolation

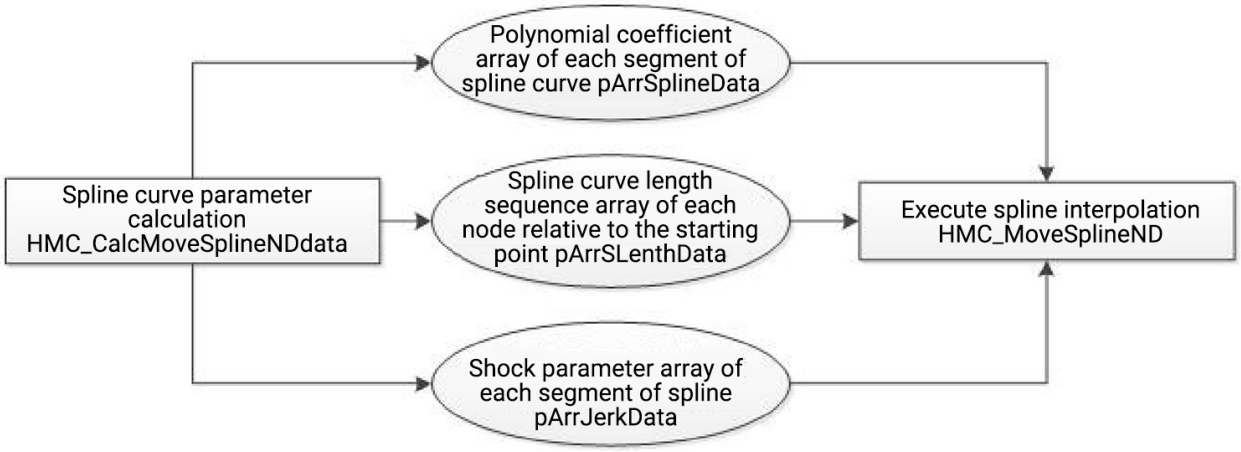
5.4.4.1 HMC_MoveSplineND (Spline Interpolation)

5.4.4.1.1 Function

Taking the current point as the starting point, perform N-axis spline interpolation motion (relative

coordinates/incremental coordinates).

This instruction uses the HMC_CalcMoveSplineNDdata function to calculate the relevant parameters of the spline curve (pArrSplineData, pArrSplineData, pArrSlenthData). It implements the N-axis spline interpolation function by specifying the combined axis number ucAxisID, each interpolating split axis number array pAxis, and the spline curve node coordinate array pArrPos.



Spline interpolation function

The N-axis spline interpolation instruction is most commonly used to implement 2D or 3D spline interpolation. The parameters can be set as follows:

(1) Two-axis spline interpolation: follow axis ucFollowAxisNum=0, endpoint axis ucEndPointAxisNum=2;

(2) Three-axis spline interpolation: follow axis ucFollowAxisNum=0, endpoint axis ucEndPointAxisNum=3.

When the above two-axis or three-axis spline interpolation is performed, the interpolation combined axis moves according to the trajectory of the endpoint axis combined motion, and the combined speed of the endpoint axis speed is approximately equal to the interpolation combined axis speed VpSpeed. By setting the combined axis parameter Speed, the interpolation profile speed can be controlled.

The axis numbers of the endpoint axis (EndPointAxis) and the follow axis (FollowAxis) are placed in the pAxis array, with the endpoint axis before the follow axis.

The coordinates in the pArrPos array consist of the coordinate components of each node on the endpoint axis and the follow axis. The number of endpoint axes and follow axes is determined by ucEndPointAxisNum and ucFollowAxisNum. The coordinate arrangement order in the array is first the coordinates of each endpoint axis, and then the coordinates of each follow axis. The array can be defined as a two-dimensional array so that the array element pArrPos[i,j] represents the coordinate component of the ith axis on the jth node.

The difference between endpoint axes and follow axes (if only conventional two-axis/three-axis spline interpolation is used, the number of follow axes is 0, and this section can be ignored) is similar to the linear axes and independent axes of N-axis linear interpolation (HMC_MoveND). When the spline interpolation motion is performed, the interpolation combined axis follows the trajectory of the combined motion of each endpoint axis and is not affected by the follow axis. The combined speed of each endpoint axis speed is approximately equal to the

interpolation combined axis speed VpSpeed. The average speed of each follow axis is approximately (total displacement of follow axis/total displacement of combined axis)*combined axis VpSpeed. The instantaneous speed is related to the instantaneous slope of the spline curve at that point. The number of endpoint axes can only be 2 or 3, while the number of follow axes can be 0. The total number of axes cannot exceed the maximum number of axes in the system (64).

The interpolation combined motion speed is set by the corresponding axis parameter Speed, and the acceleration and deceleration are set by the axis parameters Accel and Decel respectively. During the interpolation motion process, you can modify such combined axis parameters as Speed, Accel, and Decel in real time, and the modification takes effect immediately. The operation speed, acceleration and deceleration of the interpolation motion split axis have nothing to do with the axis's own Speed, Accel, and Decel, and are calculated by the real-time correlation of the combined axis.

5.4.4.1.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
pAxis	Input	POINTER TO USINT	First address of the interpolation split motion axis number array. The arrangement order is the endpoint axis first and then the follow axis.
ucEndPointAxisNum	Input	USINT	Number of endpoint axes. The speed of the endpoint axis combined motion is approximately the interpolation combined axis Speed. Value range: 2 or 3
ucFollowAxisNum	Input	USINT	Number of follow axes.
pArrPos	Input	POINTER TO LREAL	Spline curve node coordinates, first address of 2-dimensional array pArrPos[length1][length2]. The length of the first dimension of the array, length1, is the number of coordinate axes (dimensions), and the length of the second dimension, length2, is the number of spline nodes. There is the following relationship: length1 = n length2 = uiSplineNum where, n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineNum	Input	UDINT	Number of sample points in pArrPos, 1~1000.
pArrSplineData	Input	POINTER TO LREAL	First address of the array, which is the polynomial coefficient parameter of each segment of the spline curve calculated in HMC_CalcMoveSplineNDdata.
pArrSlenthData	Input	POINTER TO LREAL	First address of the array, which is the spline curve length sequence of each node relative to the starting point calculated in HMC_CalcMoveSplineNDdata.
pArrJerkData	Input	POINTER TO LREAL	First address of the array, which is the shock parameter of each segment of the spline curve calculated in HMC_CalcMoveSplineNDdata.

Parameters	Statement	Data Type	Description
HMC_MoveSplineND	Output	UDINT	Axis instruction ID

5.4.4.1.3 Instruction type

Axis motion cache instructions.

Additional information:

(1) pArrJerkData is the maximum shock calculation parameter corresponding to each spline function calculated in HMC_CalcMoveSplineNDdata. During the interpolation motion, the real-time shock data is calculated based on pArrJerkData and the real-time combined axis speed VpSpeed. If it is greater than the limit value of the combined axis parameter Jerk, the combined speed will be reduced to ensure that the shock does not exceed the limit during the motion.

(2) Near the node with a sharp turning point, the actual interpolation speed may be slightly larger than the set combined axis interpolation speed Speed, but the above processing measures can ensure that the shock here will not exceed the Jerk value set by the user.

5.4.4.1.4 Example

Example 1: Two-axis spline interpolation

See example 1 in HMC_CalcMoveSplineNDdata

Example 2: Three-axis spline interpolation

See example 2 in HMC_CalcMoveSplineNDdata

Example 3: Picking and placing materials while avoiding obstacles

The three-axis Cartesian coordinate manipulator needs to transport materials from position A to position B, and then return from B to A, and this process is repeated continuously. Since there are some obstacles between A and B, it is necessary to plan a suitable path to avoid the obstacles.

If the conventional three-axis linear interpolation is used, the path planning is as shown in the red curve in the figure below. It can be seen from the figure that at each turning point 1, 2, 3, and 4, it is necessary to decelerate to 0, and then start the next motion, which greatly reduces the handling efficiency.

If you use spline interpolation, take point A as the current starting point and move to point B, and take nodes 1', 2', 3', 4', 5', and 6' on the path, the spline interpolation trajectory is as follows. Since the direction change of the spline interpolation trajectory is smooth and will not decelerate to 0 at the turning point like linear interpolation, it can greatly improve the handling efficiency and the smoothness of motion.

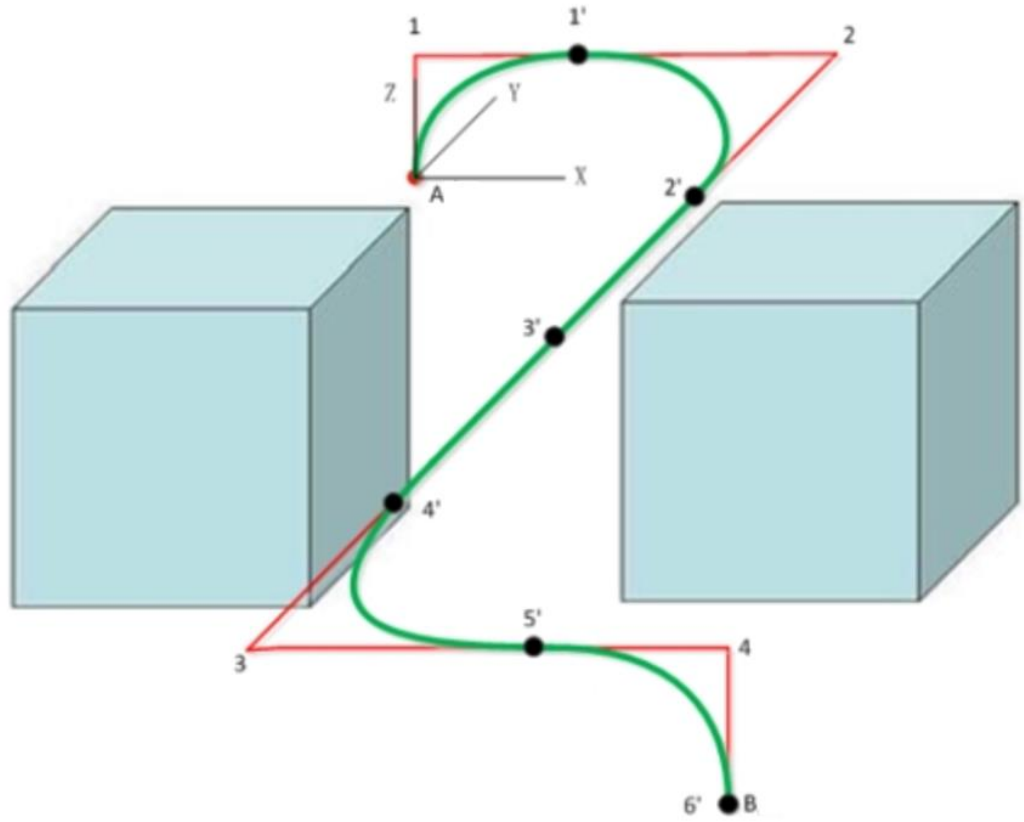


Diagram of Spline Interpolation

Let point A be the origin of the coordinates, and the coordinates of point B be (3000,-4000,0).

If three-axis spline interpolation is used, the coordinates of the stop points 1, 2, 3, and 4 pass from point A to point B and point B are (0,0,500), (1500,0,500), (1500, -4000,500), (3000,-4000,500), and (3000,-4000,0).

The three-axis linear interpolation program from point A to point B is as follows:

(****First define a node array with a size of pArrPos[3][5], and then assign a value to the target position array****)

pArrPos[0][0] := 0; (*Coordinate component of the first target point on the X-axis*)

pArrPos[0][1] := 1500; (*Coordinate component of the second target point on axis 4 (X-axis)*)

pArrPos[0][2] := 1500; (*Coordinate component of the third target point on axis 4 (X-axis)*)

pArrPos[0][3] := 3000; (*Coordinate component of the fourth target point on axis 4 (X-axis)*)

pArrPos[0][4] := 3000; (*Coordinate component of the fifth target point on axis 4 (X-axis)*)

pArrPos[1][0] := 0; (*Coordinate component of the first target point on axis 5 (Y-axis)*)

pArrPos[1][1] := 0; (*Coordinate component of the second target point on axis 5 (Y-axis)*)

pArrPos[1][2] := -4000; (*Coordinate component of the third target point on axis 5 (Y-axis)*)

pArrPos[1][3] := -4000; (*Coordinate component of the fourth target point on axis 5 (Y-axis)*)

pArrPos[1][4] := -4000; (*Coordinate component of the fifth target point on axis 5 (Y-axis)*)

pArrPos[2][0] := 500; (*Coordinate component of the first target point on axis 6 (Z-axis)*)

pArrPos[2][1] := 500; (*Coordinate component of the second target point on axis 6 (Z-axis)*)

pArrPos[2][2] := 500; (*Coordinate component of the third target point on axis 6 (Z-axis)*)

pArrPos[2][3] := 500; (*Coordinate component of the fourth target point on axis 6 (Z-axis)*)

pArrPos[2][4] := 0; (*Coordinate component of the fifth target point on axis 6 (Z-axis)*)

(*Define the split axis number array pArrAxis[3]*)

pArrAxis[0] := 4; (*X-axis number*)

pArrAxis[1] := 5; (*Y-axis number*)

pArrAxis[2] := 6; (*Z-axis number*)

(*Define the combined axis number AxisNo_3d *)

AxisNo_3d := 7;

(*Execute three-axis linear interpolation instruction*)

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][0], pArrPos[1][0], pArrPos[2][0]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][1], pArrPos[1][1], pArrPos[2][1]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][2], pArrPos[1][2], pArrPos[2][2]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][3], pArrPos[1][3], pArrPos[2][3]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][4], pArrPos[1][4], pArrPos[2][4]);

If spline interpolation is used, move from point A to point B, take nodes 1', 2', 3', 4', 5', and 6' on the path, and store them in the node position array pArrPos. The coordinates of 1', 2', 3', 4', 5', and 6' are (750,0,500), (1500,-1000,500), (1500,-2000,500), (1500,-3000) ,500), (2250,-4000,500), and (3000,-4000,0) respectively.

The spline interpolation program from point A to point B is as follows:

(****First define a node array with a size of pArrPos[3][6], and then assign a value to the node array****)

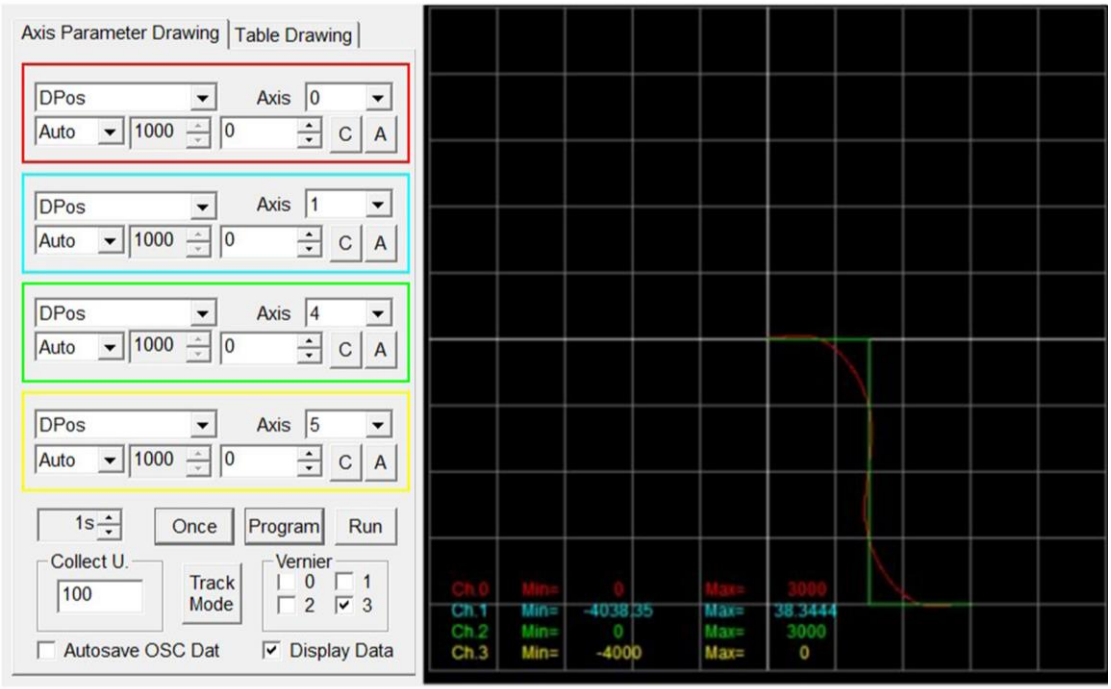
pArrPos[0][0] := 750; (*Coordinate component of the first target point on axis 0 (X-axis)*)

pArrPos[0][1] := 1500; (*Coordinate component of the second node on axis 0 (X-axis)*)

pArrPos[0][2] := 1500; (*Coordinate component of the third node point on axis 0 (X-axis)*)

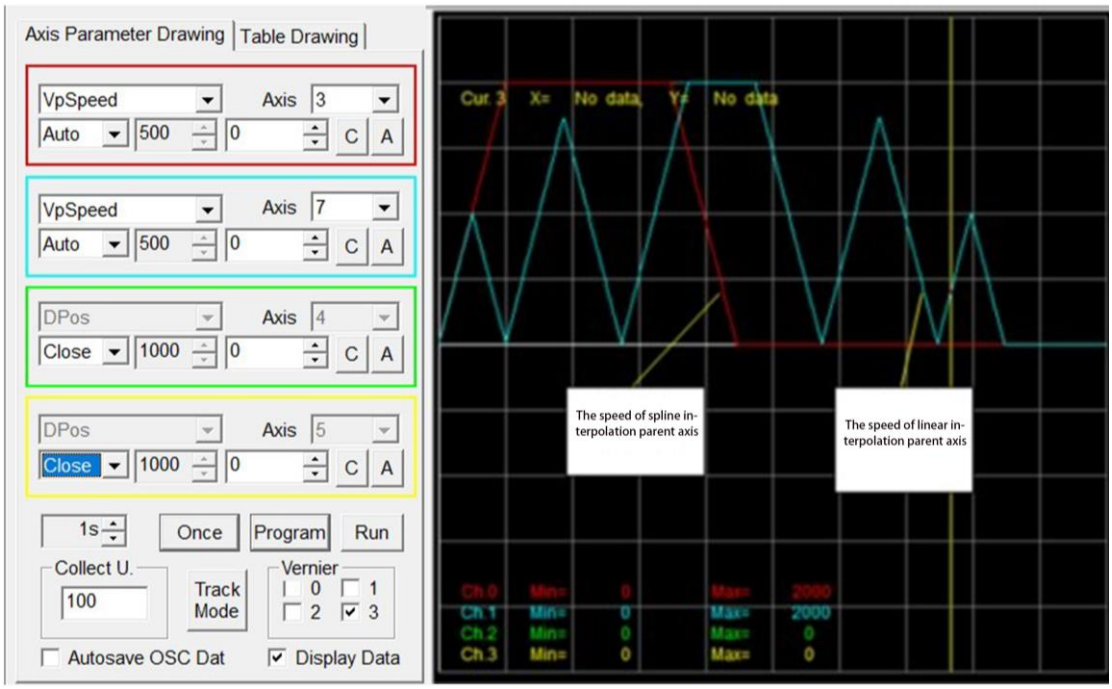
```
pArrPos[0][3] := 1500; (*Coordinate component of the fourth node point on axis 0 (X-axis)*)
pArrPos[0][4] := 2250; (*Coordinate component of the fifth node point on axis 0 (X-axis)*)
pArrPos[0][5] := 3000; (*Coordinate component of the sixth node point on axis 0 (X-axis)*)
pArrPos[1][0] :=0; (*Coordinate component of the first node point on axis 1 (Y-axis)*)
pArrPos[1][1] :=-1000; (*Coordinate component of the second node point on axis 1 (Y-axis)*)
pArrPos[1][2] :=-2000; (*Coordinate component of the third node point on axis 1 (Y-axis)*)
pArrPos[1][3] :=-3000; (*Coordinate component of the fourth node point on axis 1 (Y-axis)*)
pArrPos[1][4] :=-4000; (*Coordinate component of the fifth node point on axis 1 (Y-axis)*)
pArrPos[1][5] :=-4000; (*Coordinate component of the sixth node point on axis 1 (Y-axis)*)
pArrPos[2][0] :=500; (*Coordinate component of the first node point on axis 2 (Z-axis)*)
pArrPos[2][1] :=500; (*Coordinate component of the second node point on axis 2 (Z-axis)*)
pArrPos[2][2] :=500; (*Coordinate component of the third node point on axis 2 (Z-axis)*)
pArrPos[2][3] :=500; (*Coordinate component of the fourth node point on axis 2 (Z-axis)*)
pArrPos[2][4] :=500; (*Coordinate component of the fifth node point on axis 2 (Z-axis)*)
pArrPos[2][5] :=0; (*Coordinate component of the sixth node point on axis 2 (Z-axis)*)
(*Define arrays pArrSplineData[72], pArrSlenthData[6], and pArrJerkData [6],
save the calculated spline parameters*)
(*Call a function to generate spline parameters*)
HMC_CalcMoveSplineNDdata(pArrPos, 6, 3, 0, pArrSplineData, 72, pArrSlenthData, pArrJerkData);
(*Define the split axis number array pArrAxis[3]*)
pArrAxis[0] :=0; (*First endpoint axis number (X-axis)*)
pArrAxis[1] :=1; (*Second endpoint axis number (Y-axis)*)
pArrAxis[2] :=2; (*Third endpoint axis number (Z-axis)*)
(*Define the combined axis number AxisNo_Spline *)
AxisNo_Spline := 3;
(*Execute spline interpolation instruction*)
HMC_MoveSplineND(AxisNo_Spline, pArrAxis,3,0, pArrPos, 6, pArrSplineData, pArrSlenthData, pArrJerkData);
```

During running in AT, the top view of the XY plane of the linear interpolation trajectory (green) and spline interpolation trajectory (red) is as follows:



Motion Trajectory

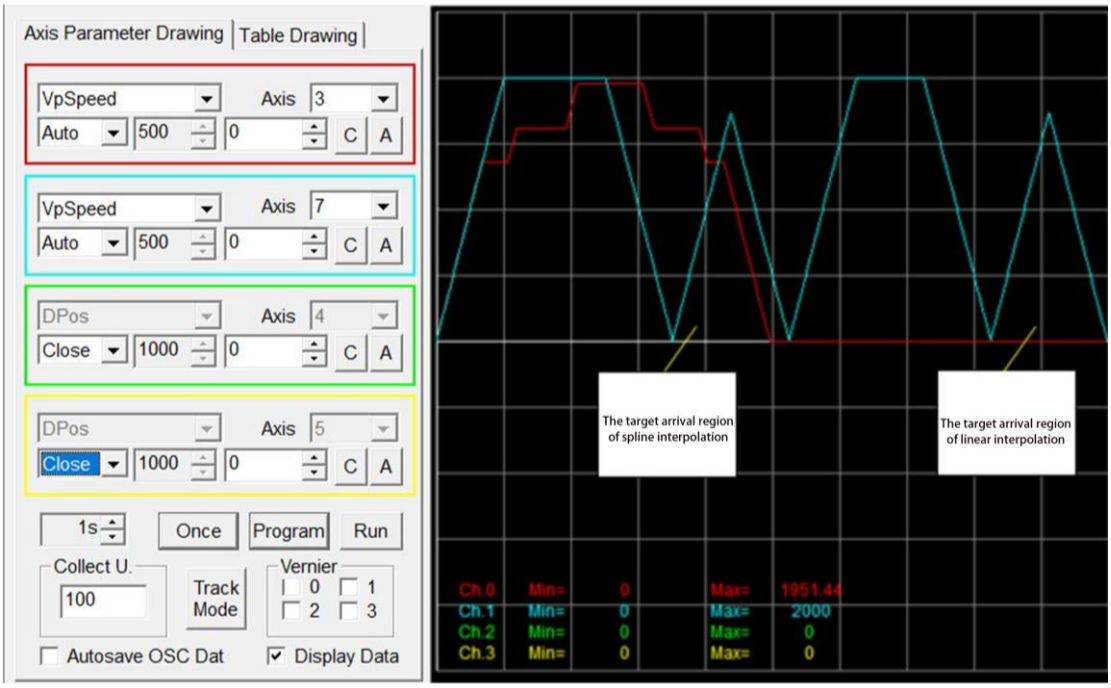
Linear interpolation and spline interpolation use the same acceleration, deceleration, and maximum speed parameters to run in AT. During the running, the speed curve of linear interpolation (cyan) and the speed curve of spline interpolation (red) are as shown in the figure below. It can be seen that at each turning point 1, 2, 3, and 4, the straight line is necessary to decelerate to 0, and then start the next motion, which greatly reduces the handling efficiency. Since the whole motion process of the spline interpolation trajectory is smooth and will not decelerate to 0 at the turning point like linear interpolation, it can greatly improve the efficiency and the smoothness of motion.



Motion Trajectory

(Red: combined axis speed of spline interpolation, cyan: combined axis speed of linear interpolation)

In order to limit the shock of spline interpolation at sharp turning points, set the limit value of the combined axis parameter Jerk in spline interpolation to 5000. The operation speed curve in AT is as shown in the figure (linear interpolation speed curve (cyan) and spline interpolation speed curve (red)). It can be seen from the figure that at the sharp turning point, by reducing the maximum speed of this section, it can ensure that the shock during running does not exceed the set value of 5000.



Motion Trajectory

(Red: combined axis speed of spline interpolation, cyan: combined axis speed of linear interpolation)

To move from point B to point A, you only need to reverse the order of spline nodes 1', 2', 3', 4', 5', and 6', and store them in the pArrayPos array to generate the parameter model for return. No further details will be given here.

5.4.4.2 HMC_MoveSplineNDEx (Advanced Spline Interpolation)

5.4.4.2.1 Function

Taking the current point as the starting point, perform N-axis spline interpolation motion (relative coordinates/incremental coordinates).

5.4.4.2.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	Input	USINT	Interpolation combined motion axis number
pAxis	Input	POINTER TO USINT	First address of the interpolation split motion axis number array. The arrangement order is the endpoint axis first and then the

Parameters	Statement	Data Type	Description
			follow axis.
ucEndPointAxisNum	Input	USINT	Number of endpoint axes. The speed of the endpoint axis combined motion is approximately the interpolation combined axis Speed. Value range: 2 or 3
ucFollowAxisNum	Input	USINT	Number of follow axes.
pArrPos	Input	POINTER TO LREAL	Spline curve node coordinates, 2-dimensional array. The length of the first dimension, length1, is the number of coordinate axes (dimensions), and the length of the second dimension, length2, is the number of spline nodes. There is the following relationship: length1 = n length2 = uiSplineNum where, n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineNum	Input	UDINT	Number of sample points in pArrPos, 1~1000.
pArrSplineData	Input	POINTER TO LREAL	First address of the array, which is the polynomial coefficient parameter of each segment of the spline curve calculated in HMC_CalcMoveSplineNDdata.
pArrSlenthData	Input	POINTER TO LREAL	First address of the array, which is the spline curve length sequence of each node relative to the starting point calculated in HMC_CalcMoveSplineNDdata.
pArrJerkData	Input	POINTER TO LREAL	First address of the array, which is the shock parameter of each segment of the spline curve calculated in HMC_CalcMoveSplineNDdata.
dSpeed	Input	LREAL	Combined axis target speed (interpolation profile speed)
HMC_MoveSplineNDEx	Output	UDINT	Axis instruction ID

5.4.4.2.3 Instruction type

Axis motion cache instructions.

5.4.4.2.4 Example

Refer to HMC_MoveSplineND (Spline interpolation).

5.4.4.3 HMC_CalcMoveSplineNDdata (Generating Spline Data)

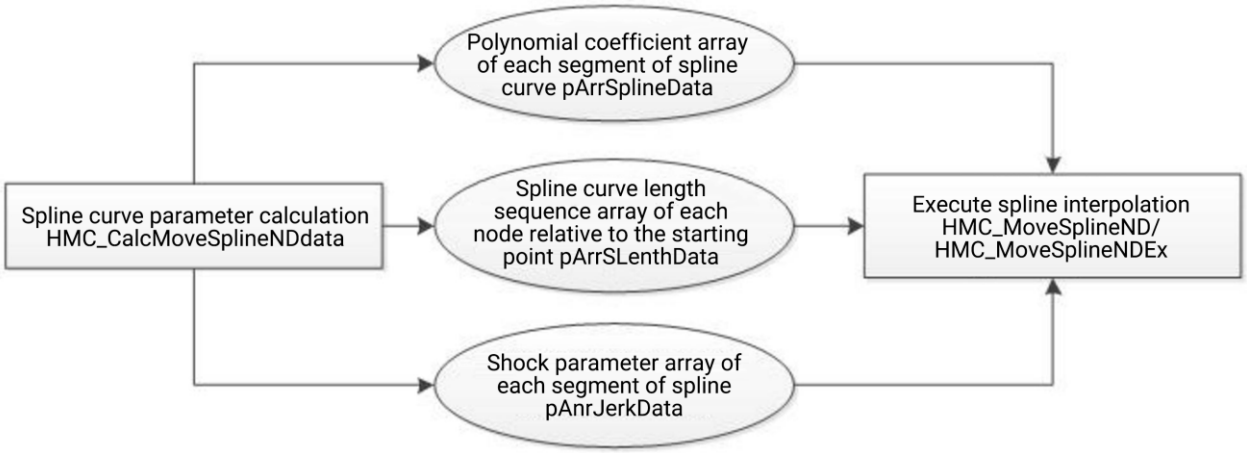
5.4.4.3.1 Function

Because the spline curve has second-order continuous differentiability, it can effectively reduce the speed/acceleration jump caused by the direction change during the interpolation motion, thereby reducing the mechanical shock and achieving a smooth motion effect.

This function generates a cubic spline curve for N-axis interpolation. The calculated spline curve parameters are saved in the array specified by the user. The user does not need to care about the specific content in the array. Subsequently, the N-axis spline interpolation function can be achieved by calling the data in the array using the

spline interpolation instructions

HMC_MoveSplineND/HMC_MoveSplineNDEx.



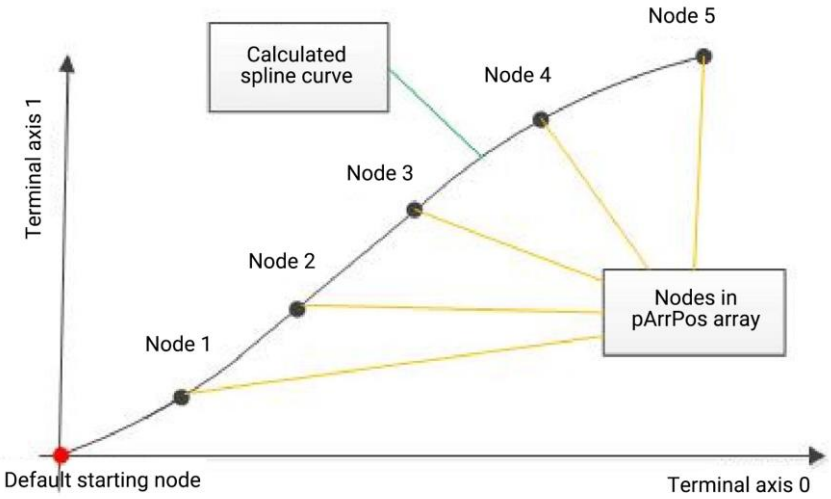
Spline interpolation function

The function calculates the parameters for generating a spline curve based on the user-specified spline node coordinate pArrPos (the starting point is the origin by default) and the number of nodes uiSplineNum. The parameters include the polynomial coefficient of each segment, the spline length sequence of each node relative to the starting point, and the shock parameters of each segment of the spline, which are stored in the user-specified arrays pArrSplineData, pArrSLenthData, and pArrJerkData respectively.

The coordinates in the pArrPos array consist of the components of each node coordinate on the endpoint axis (EndPointAxis) and the follow axis (FollowAxis). The number of endpoint axes and follow axes is determined by ucEndPointAxisNum and ucFollowAxisNum. The coordinate arrangement order in the array is first the coordinates of each endpoint axis, and then the coordinates of each follow axis. It is suggested that the array is defined as a two-dimensional array so that the array element pArrPos[i,j] represents the coordinate component of the jth axis on axis i. See examples for details.

The N-axis spline interpolation function is most commonly used to implement 2D or 3D spline interpolation. The parameters can be set as follows:

(1) Two-axis spline interpolation: follow axis ucFollowAxisNum=0, endpoint axis ucEndPointAxisNum=2.



Two-axis Spline Interpolation

(2) Three-axis spline interpolation: follow axis ucFollowAxisNum=0, endpoint axis ucEndPointAxisNum=3.

When the spline interpolation instructions HMC_MoveSplineND/HMC_MoveSplineNDEx are used for interpolation motion, the interpolation combined axis follows the trajectory of the combined motion of each endpoint axis and is not affected by the follow axis. The combined speed of each endpoint axis speed is approximately equal to the interpolation combined axis speed Speed.

The concepts of endpoint axis and follow axis can be compared to the linear axis and independent axis of N-axis linear interpolation (HMC_MoveND) (if you only use conventional two-axis/three-axis spline interpolation, you don't need to understand it). For details, see HMC_MoveSplineND (Spline interpolation).

The function first calculates the spline curve length sequence pArrSLenthData, and then uses this length sequence to calculate the spline curve model parameters of each split axis (endpoint axis, follow axis). The spline curve length sequence pArrSLenthData is calculated from the component of the node coordinates on the endpoint axis, and represents the displacement of the combined axis at each node in the spline interpolation instructions HMC_MoveSplineND/HMC_MoveSplineNDEx. The combined axis speed is approximately equal to the speed of the combined motion of each endpoint axis.

5.4.4.3.2 Parameter

Parameters	Statement	Data Type	Description
pArrPos	Input	POINTER TO LREAL	Spline curve node coordinates, first address of 2-dimensional array pArrPos[length1][length2]. The length of the first dimension of the array, length1, is the number of coordinate axes (dimensions), and the length of the second dimension, length2, is the number of spline nodes. There is the following relationship: length1 = n length2 = uiSplineNum where, n = ucEndPointAxisNum +ucFollowAxisNum

Parameters	Statement	Data Type	Description
uiSplineNum	Input	UDINT	Number of sample points in pArrPos, 1~1000.
ucEndPointAxisNum	Input	USINT	Number of endpoint axes. The spline curve length sequence pArrSlenthData is calculated from the component of the node coordinates on the endpoint axis. The speed of the endpoint axis combined motion is approximately the interpolation combined axis Speed. Value range: 2 or 3
ucFollowAxisNum	Input	USINT	Number of follow axes. The spline curve model of the follow axis is calculated based on the spline curve length sequence pArrSlenthData and the coordinate components of the spline node on the follow axis.
pArrSplineData	Input	POINTER TO LREAL	The first address of the user-specified one-dimensional array pArrSplineData[length], which is used to save the calculated polynomial coefficient parameter data of each segment of the spline curve. It shall satisfy: Array length >= n*uiSplineNum*4 where, n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineDataSize	Input	UDINT	pArrSplineData array length.
pArrSlenthData	Input	POINTER TO LREAL	The first address of the user-specified one-dimensional array pArrSlenthData[length], which is used to save the calculated spline curve length sequence of each node relative to the starting point. It shall satisfy: Total array length >= uiSplineNum
pArrJerkData	Input	POINTER TO LREAL	The first address of the array is specified by the user, which is used to save the calculated shock parameters of each segment of the spline curve. It shall satisfy: Array length >= uiSplineNum
HMC_CalcMoveSplineNDdata	Output	UDINT	Axis instruction ID

5.4.4.3.3 Instruction type

It is a non-axis motion cache instruction.

Additional information:

(1) The spatial distance between nodes shall not be too different, otherwise, the numerical calculation deviation near the nodes with large differences may affect the accuracy.

(2) When each node on the spline curve is on a straight line passing through the origin, the generated spline curve is a straight line.

(3) If there is only one node, the resulting spline curve is a line segment connecting the origin to that node. Therefore, N-axis spline interpolation can complete the functions of two-axis, three-axis, and N-axis linear

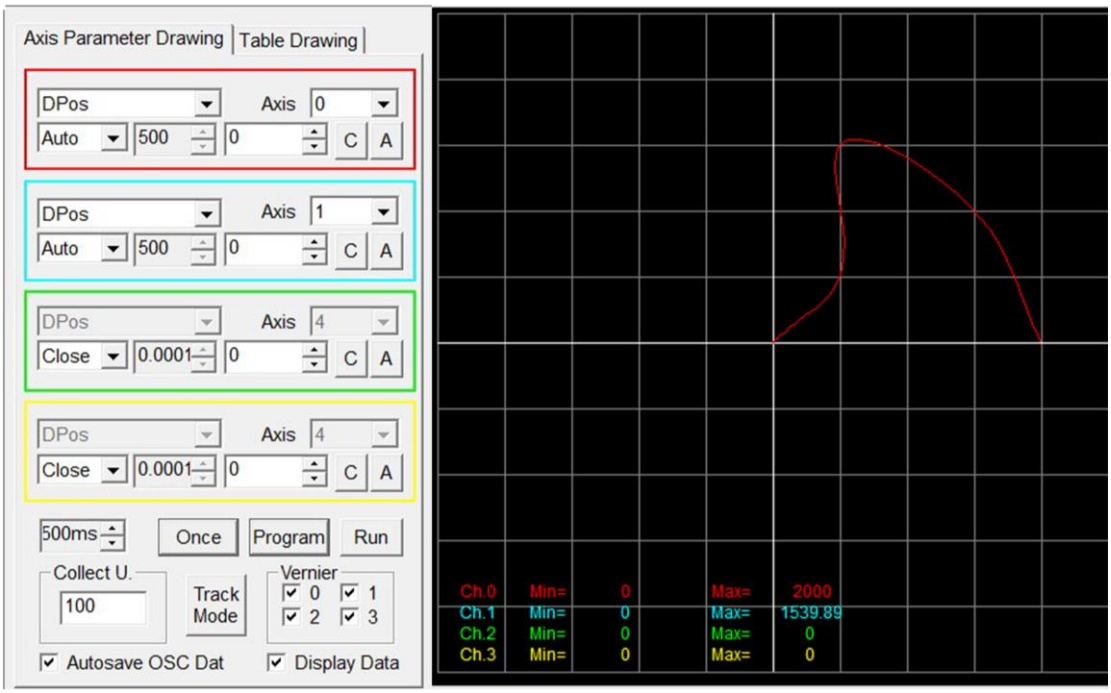
interpolation.

5.4.4.3.4 Example

Example 1: Two-axis spline interpolation

The number of endpoint axes is 2, and the number of follow axes is 0. The axis numbers of the two endpoint axes are 0 and 1, corresponding to the X-axis and Y-axis respectively. The number of nodes is 3, and the node coordinates are (500,500), (500, 1500), (1500, 1000), and (2000,0). Use HMC_CalcMoveSplineNDdata to generate spline curve parameters, and then use these parameters to call HMC_MoveSplineND to execute the spline interpolation instruction (the current point is the origin) to generate the spline curve trajectory.

The AT oscilloscope running results are as follows:



Motion Trajectory

The procedure is as follows:

(****First define a node array with a size of pArrPos[2][3], and then assign a value to the node array****)

pArrPos[0][0] := 500; (*Coordinate component of the first node on axis 0 (X-axis)*)

pArrPos[0][1] := 500; (*Coordinate component of the second node on axis 0 (X-axis)*)

pArrPos[0][2] := 1500; (*Coordinate component of the third node on axis 0 (X-axis)*)

pArrPos[0][3] := 2000; (*Coordinate component of the fourth node on axis 0 (X-axis)*)

pArrPos[1][0] := 500; (*Coordinate component of the first node on axis 1 (Y-axis)*)

pArrPos[1][1] := 1500; (*Coordinate component of the second node on axis 1 (Y-axis)*)

pArrPos[1][2] := 1000; (*Coordinate component of the third node on axis 1 (Y-axis)*)

pArrPos[1][3] := 0; (*Coordinate component of the fourth node on axis 1 (Y-axis)*)

(*Define arrays pArrSplineData[32], pArrSlenthData[4], and pArrJerkData [4] and save the calculated spline parameters*)

(*Call a function to generate spline parameters*)

HMC_CalcMoveSplineNDdata(pArrPos, 4, 2, 0, pArrSplineData, 32, pArrSlenthData, pArrJerkData);

(*Define the axis number array pArrAxis[2]*)

pArrAxis[0] :=0; (*First endpoint axis number*)

pArrAxis[1] :=1; (*Second endpoint axis number*)

(*Execute spline interpolation instruction*)

HMC_MoveSplineND(2,pArrAxis,2,0, pArrPos,4,

pArrSplineData, pArrSlenthData, pArrJerkData);

Example 2: Three-axis spline interpolation curve

The number of endpoint axes is 3, and the number of follow axes is 0. The axis numbers of the three endpoint axes are 0, 1, and 2, corresponding to the X-axis and Y-axis respectively. The number of nodes is 4, and the node coordinates are (500, 500, 500), (500, 1500, 500), (1500, 1000, 1000), and (2000, 0, 0). Use HMC_CalcMoveSplineNDdata to generate spline curve parameters, and then use these parameters to call HMC_MoveSplineND to execute the spline interpolation instruction (the current point is the origin) to generate the spline curve trajectory. The procedure is as follows:

(****First define a node array with a size of pArrPos[3][3], and then assign a value to the node array****)

pArrPos[0][0] := 500; (*Coordinate component of the first node on axis 0 (X-axis)*)

pArrPos[0][1] := 500; (*Coordinate component of the second node on axis 0 (X-axis)*)

pArrPos[0][2] := 1500; (*Coordinate component of the third node on axis 0 (X-axis)*)

pArrPos[0][3] := 2000; (*Coordinate component of the fourth node on axis 0 (X-axis)*)

pArrPos[1][0] := 500; (*Coordinate component of the first node on axis 1 (Y-axis)*)

pArrPos[1][1] := 1500; (*Coordinate component of the second node on axis 1 (Y-axis)*)

pArrPos[1][2] := 1000; (*Coordinate component of the third node on axis 1 (Y-axis)*)

pArrPos[1][3] := 0; (*Coordinate component of the fourth node on axis 1 (Y-axis)*)

pArrPos[2][0] :=500; (*Coordinate component of the first node on axis 2 (Z-axis)*)

pArrPos[2][1] :=500; (*Coordinate component of the second node on axis 2 (Z-axis)*)

pArrPos[2][2] := 1000; (*Coordinate component of the third node on axis 2 (Z-axis)*)

pArrPos[2][3] :=0; (*Coordinate component of the fourth node on axis 2 (Z-axis)*)

(*Define arrays pArrSplineData[48], pArrSlenthData[4], and pArrJerkData [4] and save the calculated spline parameters*)

(*Call a function to generate spline parameters*)

HMC_CalcMoveSplineNDdata(pArrPos, 4, 3, 0, pArrSplineData,48, pArrSlenthData, pArrJerkData);

(*Define the axis number array pArrAxis[3]*)

pArrAxis[0] :=0; (*First endpoint axis number*)

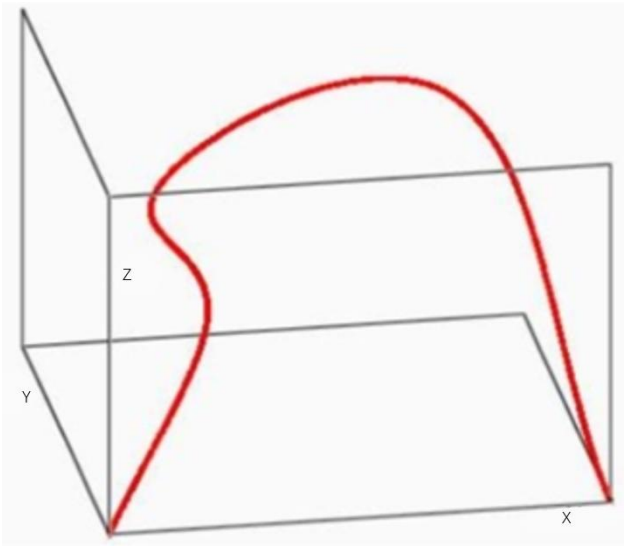
pArrAxis[1] :=1; (*Second endpoint axis number*)

pArrAxis[2] :=2; (*Third endpoint axis number*)

(*Execute spline interpolation instruction*)

HMC_MoveSplineND (3, pArrAxis, 3, 0, pArrPos, 4, pArrSplineData, pArrSlenthData, pArrJerkData):

The running results are as follows:



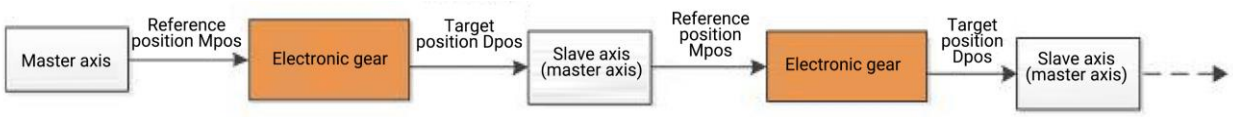
3D Trajectory

5.4.5 Electronic Gear

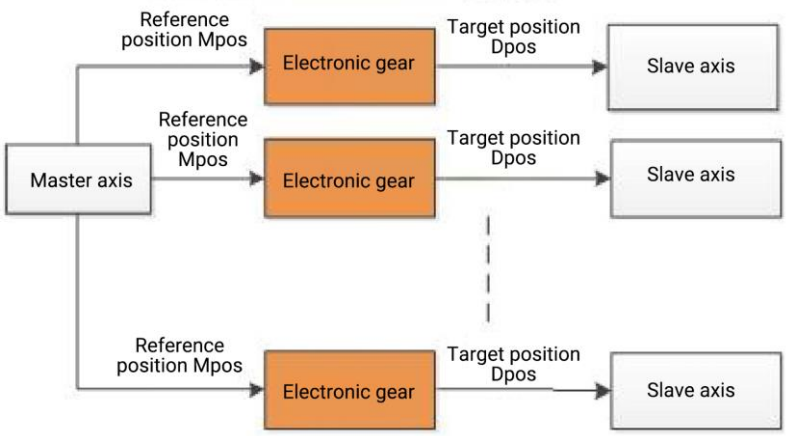
5.4.5.1 HMC_Gear (Electronic gear)

5.4.5.1.1 Function

Electronic gears are mainly used to achieve shaftless transmission. Shaftless transmission technology has the advantages of high transmission accuracy, simple structure, wide transmission ratio range, and convenient adjustment. Therefore, it can replace traditional mechanical transmission to achieve accurate transmission relationships. In fields such as printing and dyeing, textile, papermaking, and gear processing machine tool inline transmission that require multi-axis synchronous transmission, shaftless transmission technology has broad application prospects.

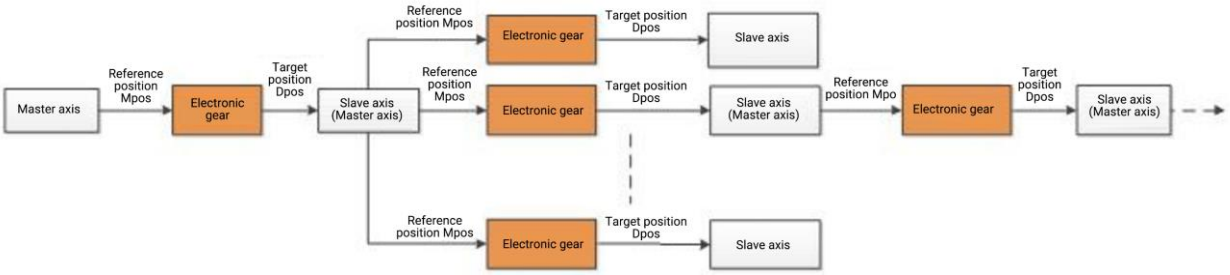


Serial Electronic Gear Structure Diagram



Parallel Electronic Gear Structure Diagram

There are two basic structures in the use of electronic gears: serial type and parallel type, as shown in the figure above. In the serial structure, the slave axis serves as the master axis of the next-level electronic gear. In the parallel structure, each slave axis has a common master axis. In actual use, it can be selected according to needs, or a combination of these two basic structures can be used to build a composite electronic gear structure.



Composite Electronic Gear Structure Diagram

After the electronic gear is used, the increment of slave axis Dpos and that of master axis Mpos satisfy the following relationship:

$$\Delta Dpos_S = \Delta Mpos_M * \left(\frac{iRatioNumerator}{iRatioDenominator}\right)$$

5.4.5.1.2 Parameter

Parameters	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number
ucMasterAxis	Input	USINT	Master axis number
iRatioNumerator	Input	DINT	Gear ratio numerator

iRatioDenominator	Input	DINT	Gear ratio denominator (cannot be 0)
HMC_Gear	Output	UDINT	Axis instruction ID: Success 0xFFFFFFFF: Function parameter error

5.4.5.1.3 Instruction type

Axis motion cache instructions.

Additional information:

(1) During the execution of this instruction, the gear ratio can be dynamically modified by calling the HMC_SetGearRatio function;

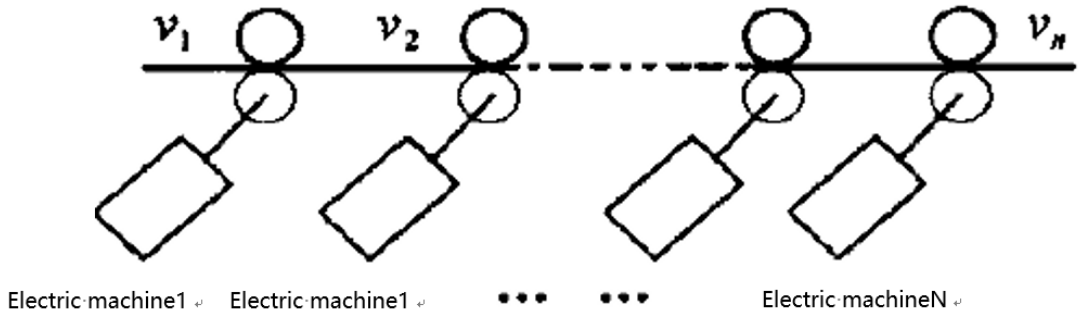
(2) From a mathematical point of view, HMC_Gear and HMC_Superimpose are both special cases of HMC_SuperimposeEx;

(3) HMC_Gear and HMC_Superimpose can be used together to realize all the functions of HMC_SuperimposeEx.

5.4.5.1.4 Example

Example 1: Shaftless control technology for continuous process equipment

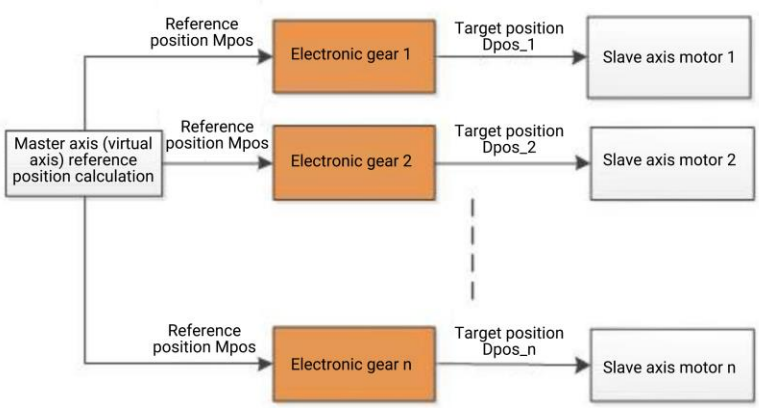
Many industrial devices for continuous processes, such as hot and cold rolling mills, paper machines, polymer processing lines, and textile dyeing machinery, have similar requirements for transmission systems, that is, using multi-unit distributed transmission. The number of units is often as high as dozens, and each unit is driven by a separate motor; starting from the process and the conditions for maintaining a given tension between the two units, the speed relationship between the unit machines must remain strictly unchanged and must be stable at the linear speed of the processed strip (metal, paper, polymer film, textile, etc.) with a certain accuracy for a long time, and; each unit machine maintains appropriate tension during processing through the synchronous transmission to avoid strip elongation or wrinkles during processing.



Transmission Diagram of Continuous Process Equipment

The transmission structure of this type of device is shown in the figure above. It usually adopts a parallel electronic gearbox structure. The system segments are coupled to each other through the jointly processed strip load and the common speed given and the speed ratio control of each segment.

The master axis is a virtual axis, which is used to generate a common reference position. The target displacement of each slave axis is generated by electronic gear transformation of the common reference position, to drive each slave axis to operate synchronously according to a predetermined position and speed relationship.



Implementation Method of Electronic Gear Transmission for Continuous Process Equipment

If there is a transmission relationship within a certain segment, a sub-electronic gear control system can be constructed in the segment to form a composite electronic gear control system with the entire system.

Example 2: Shaftless transmission for thread turning

The master axis motor of the CNC lathe is axis 0, the axial feed motor is axis 1, and the feed screw lead is l_f. If you want to turn a thread with a lead of l_x, you can use the electronic gear function to achieve this.

It can be calculated that the transmission ratio between the feed motor and the master axis when threads with this lead are processed:

$$\frac{l_x}{l_f}$$

To simplify this value to its simplest fractional form, let:

$$\frac{l_x}{l_f} = \frac{\text{iRatioNumerator}}{\text{iRatioDenominator}}$$

The program is as follows (ST language) (assume the master axis motor, axial feed motor, and radial feed motor are axes 0, 1, and 2 respectively):

(**Set the axis type to bus axis**)

HMC_SetAxisType(Axis0.AxisID, 50);

HMC_SetAxisType(Axis1.AxisID, 50);

HMC_SetAxisType(Axis2.AxisID, 50);

(**Set the speed, acceleration, and deceleration parameters of each axis (omitted)**)

(**Set the speed, acceleration, and deceleration parameters of each axis (omitted)**)

(**Move to the entry point (omitted)**)

(**Radial feed motor given thread depth (omitted)**)

(*Send the HMC_Gear instruction, with Axis 1 (axial feed motor) as the slave axis, Axis 0 (master axis motor) as the master axis, and gear ratio as

iRatioNumerator/iRatioDenominator *)

HMC_Gear(Axis1.AxisID, Axis0.AxisID, iRatioNumerator, iRatioDenominator);

(*Drive the master axis to operate, and the axial feed motor will move synchronously driven by the electronic gear to complete thread turning*)

Example 3: H-shaped synchronous toothed belt parallel mechanism motion control (see Example 1 in HMC_Superimpose).

5.4.5.2 HMC_SetGearRatio (Setting Electronic Gear Ratio)

5.4.5.2.1 Function

It is used to set the gear ratio. After the HMC_Gear instruction is started, it can be called in real time to dynamically change the ratio of the electronic gear in operation, thereby achieving the effect of dynamically changing the electronic gear.

5.4.5.2.2 Parameter

Parameters	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number
iRatioNumerator	Input	DINT	Gear ratio numerator
iRatioDenominator	Input	DINT	Gear ratio denominator
HMC_SetGearRatio	Output	UDINT	0: Success Other values: Error code

5.4.5.2.3 Instruction type

It is a non-axis motion cache instruction.

5.4.5.3 HMC_GetGearMaster (Getting Electronic Gear Master Axis)

5.4.5.3.1 Function

It is used to get the master axis number of the electronic gear. This instruction can only obtain the master axis number of the currently running HMC_Gear instruction.

5.4.5.3.2 Parameter

Parameters	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number

HMC_GetGearMaster	Output	INT	Electronic gear master axis number: Success -1: Electronic gear not currently configured or function parameter error

5.4.5.3.3 Instruction type

It is a non-axis motion cache instruction.

5.4.5.4 HMC_GetGearRatio (Getting Electronic Gear Ratio)

5.4.5.4.1 Function

It is used to get the gear ratio of the electronic gear. This instruction can only obtain the gear ratio of the currently running HMC_Gear instruction.

5.4.5.4.2 Parameter

Parameters	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number
HMC_GetGearRatio	Output	LREAL	Electronic gear ratio

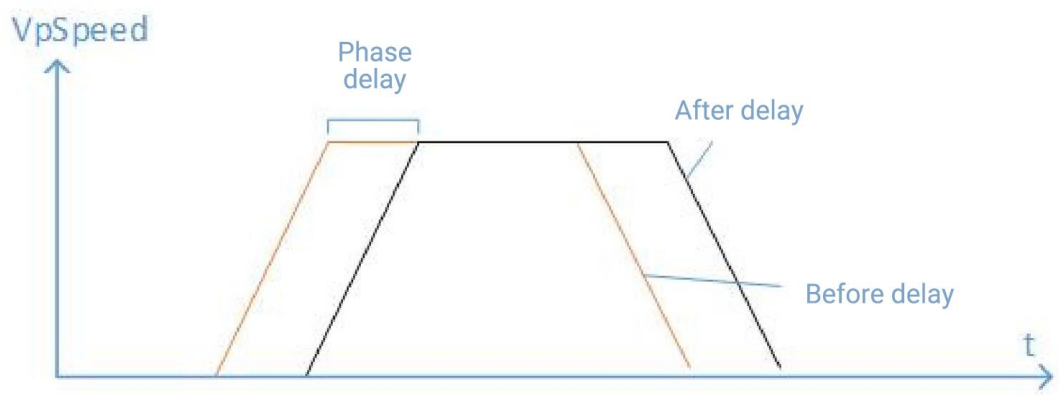
5.4.5.4.3 Instruction type

It is a non-axis motion cache instruction.

5.4.5.5 HMC_SetGearPhaseOffset (Setting Electronic Gear Phase Delay)

5.4.5.5.1 Function

This instruction can adjust the follow phase of the electronic gear slave axis, that is, make a phase delay to the original slave axis trajectory on the time axis. The adjusted slave axis trajectory lags behind the original trajectory by iPhaseOffset servo cycles (with a maximum value of 128). as shown in the figure:



Slave Axis Phase Delay Trajectory

The phase delay of other instructions can be achieved indirectly through electronic gears. For example, in the cam instruction, the cam master axis can be used as the slave axis of the 1:1 electronic gear, and the phase delay of

the cam can be indirectly achieved by adjusting the phase delay of the electronic gear.

5.4.5.5.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	Input	USINT	Slave axis number
iPhaseOffset	Input	DINT	Phase delay, 0~128, in servo cycles
HMC_SetGearPhaseOffset	Output	UDINT	Error code

5.4.5.5.3 Instruction type

It is a non-axis motion cache instruction.

5.4.5.6 HMC_GetGearPhaseOffset (Getting Electronic Gear Phase Delay)

5.4.5.6.1 Function

Get the set phase delay of electronic gear.

5.4.5.6.2 Parameter

Parameters	Statement	DataType	Description
ucAxisID	Input	USINT	Slave axis number, 0~63
HMC_GetGearPhaseOffset	Output	DINT	Phase delay, the unit is servo cycle

5.4.5.6.3 Instruction type

It is a non-axis motion cache instruction.

5.4.5.7 HMC_SetGearTransationSwitch (Electronic Gear Transition Switch)

5.4.5.7.1 Function

Turn on/off the electronic gear smoothing function. When the switch is turned on, the electronic gear will automatically insert a smooth transition process during the startup or when the electronic gear ratio is switched. The transition process uses the acceleration and deceleration of the axis itself to prevent mechanical shock.

5.4.5.7.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucValue	Input	BOOL	0: Closed 1: On
HMC_SetGearTransationSwitch	Output	UDINT	0: Execution succeeded 0xFFFFFFFF: Execution failed

5.4.5.7.3 Instruction type

It is a non-axis motion cache instruction.

5.4.6 Electronic Cam

5.4.6.1 HMC_Cam (Electronic Cam)

5.4.6.1.1 Function

By selecting discrete coordinate points on the target motion trajectory (two-dimensional), the electronic cam function can be used to approximate the target motion trajectory through master-slave axis linkage. The approximation method is "substituting polylines for curves", that is, using polyline segments connecting the above discrete coordinate points to approximate the target curve.

Electronic cams can perform similar functions to mechanical cams, but do not have the disadvantages of mechanical cams such as difficulty in design, high processing costs, high motion and easy wear and tear. For this, electronic cams are ideal substitutes for mechanical cams in many situations. Electronic cams can also be used to approximate irregular curves. Used in conjunction with the HMC_ClcMoveLinkData or HMC_ClcFlexLinkData instruction, the action processes required for application scenarios such as follow-up shearing, flying shearing, and rotary cutting can also be realized.

Electronic cams are divided into master and slave axes. The positions of the cam master and slave axes are pre-stored in the master/slave axis position array pArrPos. pArrPos is a two-dimensional array. The first dimension pArrPos[0] is the master axis position, and the second dimension pArrPos[1] is the slave axis position, between which there is a one-to-one correspondence. When the cam starts, the starting position of the cam master axis is set according to different starting methods; during the cam operation, the master axis runs normal motion control instructions, and the controller calculates the theoretical position of the slave axis based on the current Mpos value of the master axis and the data in pArrPos and drives the slave axis to perform corresponding linkage actions, and at the same time, it is calculated based on the current master axis Mpos value to determine whether the cam cancellation condition is met.

According to the boundary range of the master axis position, cams can be divided into:

(1) Single cam——After the cam is started, if the master axis position of the cam is within the effective stroke range, the cam can maintain normal linkage operation (the master axis can move in one direction or reciprocate. When the master axis reciprocates within the effective stroke range, the slave axis will also perform reciprocating linkage actions). When the master axis exceeds the effective stroke range, the cam instruction will be automatically canceled.

(2) Cyclic cam——If the master axis position of the cam has no boundary range limit, when the corresponding cancellation instruction is not executed after startup, the cam function will always be effective.

According to the triggering method of cam starting conditions, cams can be divided into the following types:

(1) Immediate start——The cam function starts immediately after the cam instruction is sent, and the starting position is the current Mpos position of the master axis;

(2) Fixed position start (in-position start)——After the cam instruction is sent, when the cam master axis Mpos crosses the cam start position from the negative direction, the cam function starts, and the starting position is the fixed position;

(3) Event start——After the cam instruction is sent, the cam is started when a certain high-speed capture channel is triggered, and the starting position is the master axis Mpos position captured by the high-speed capture channel;

(4) DI start——After the cam instruction is sent, the cam is started when a certain DI signal is 1, and the starting position is the master axis Mpos position when the program detects that DI is 1.

The event start method is similar to the DI start method, both of which are triggered by external signals. The difference is that, in event start mode, the high-speed capture channel can be used to record the precise position at the moment when the DI signal is triggered, so a more accurate starting position can be obtained, which has higher accuracy than the DI start method.

The stroke length Delta of the cam master axis is the last element value of the cam master axis position array minus the first element value, that is Delta = pArrPos[0,uiPosNum- 1] - pArrPos[0,0], where uiPosNum is the number of positions. Suppose the starting point of the cam is Mpos0, then the effective stroke range of the single cam is [Mpos0, Mpos0+Delta). When the master axis position of the single cam exceeds this stroke range, the cam will be canceled. When the cam master axis is within the effective stroke range, the master axis can perform unidirectional and reciprocating motion. When the master axis reciprocates within the effective stroke range, the slave axis will also perform reciprocating linkage actions.

For cyclic cams, when the cam master axis position crosses a stroke length Delta of the cam master axis, it will enter the next cam stroke. At this time, the slave axis will take the slave axis position at the end of the previous cam stroke as the starting point, and perform the incremental displacement calculation based on this to repeat the linkage actions of the previous cam stroke on and on. The cam master axis can also perform unidirectional and reciprocating motion. To cancel the cyclic cam, you can execute Hmc_Cancel to cancel it immediately, or execute the HMC_CancelREPCam instruction to cancel at an integer multiple of the cam master axis stroke. For detailed usage, please refer to the relevant instructions.

5.4.6.1.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucSlaveAxis	Input	USINT	I, Q, M or constant	Slave axis number
ucMasterAxis	Input	USINT	I, Q, M or constant	Master axis number
pArrPos	Input	POINTER TO LREAL	I, Q, M or constant	The first address of position array (two-dimensional). In the two-dimensional array pArrPos[2,uiPosNum], pArrPos[0] is the master axis position array, and pArrPos[1] is the slave axis position array.
uiPosNum	Input	UDINT	I, Q, M or constant	Number of positions, which shall be greater

Parameter	Statement	Data Type	Storage Area	Description
				than or equal to 3
dScale	Input	LREAL	I, Q, M or constant	Cam slave axis position data scale factor. Actual position output of cam slave axis = set output of slave axis position array*dScale
ucMode	Input	USINT	I, Q, M or constant	0: Single immediate start; 1: Single fixed position start; 2: Single event start; 3: Single DI start; 4: Immediate cyclic start; 5: Cyclic fixed-position start; 6: Cyclic event start; 7: Cyclic DI start.
dTrigMes	Input	LREAL	I, Q, M or constant	Startup information: 1. When the cam starts in in-position start mode, this value indicates the starting position of the master axis; 2. When the cam is triggered in event start mode, it indicates a certain high-speed capture channel number; 3. When the cam is triggered in DI (1) start mode, it indicates the DI channel (0~63).

5.4.6.1.3 Instruction type

Axis motion cache instructions.

Additional information:

(1) The elements in the master axis position array pArrPos[0] must be monotonically increasing (they cannot be the same number), and the data can be positive or negative. The master/slave axis position data cannot be modified during cam operation, otherwise it may cause system errors!

(2) When the cam is triggered in event start mode, dTrigMes represents a high-speed capture channel, so a high-speed capture channel shall be configured in advance to capture the start (CaptureAxis in the HMC_CaptureConfig functional block must be the cam master axis number).

(3) The reference starting point when the cam starts, the dynamic coordinate point used for calculation during operation, and the reference coordinate point when canceling are all based on the Mpos value of the cam master axis as the reference base. When the master axis is under position closed-loop control, since the master axis Mpos comes from an external detection device, in order to avoid accidental cancellation of the cam caused by small disturbances in Mpos at the boundary of the cam's effective stroke range, the cancellation conditions will also take into account the Dpos value of master axis.

(4) For single cams, when Mpos comes from an external detection device, in order to avoid abnormal cancellation of the cam caused by small disturbances, it shall ensure that there is a certain margin between the normal working range of the cam and the cam boundary position (to ensure that the disturbance of Mpos does not exceed the cam boundary position) .

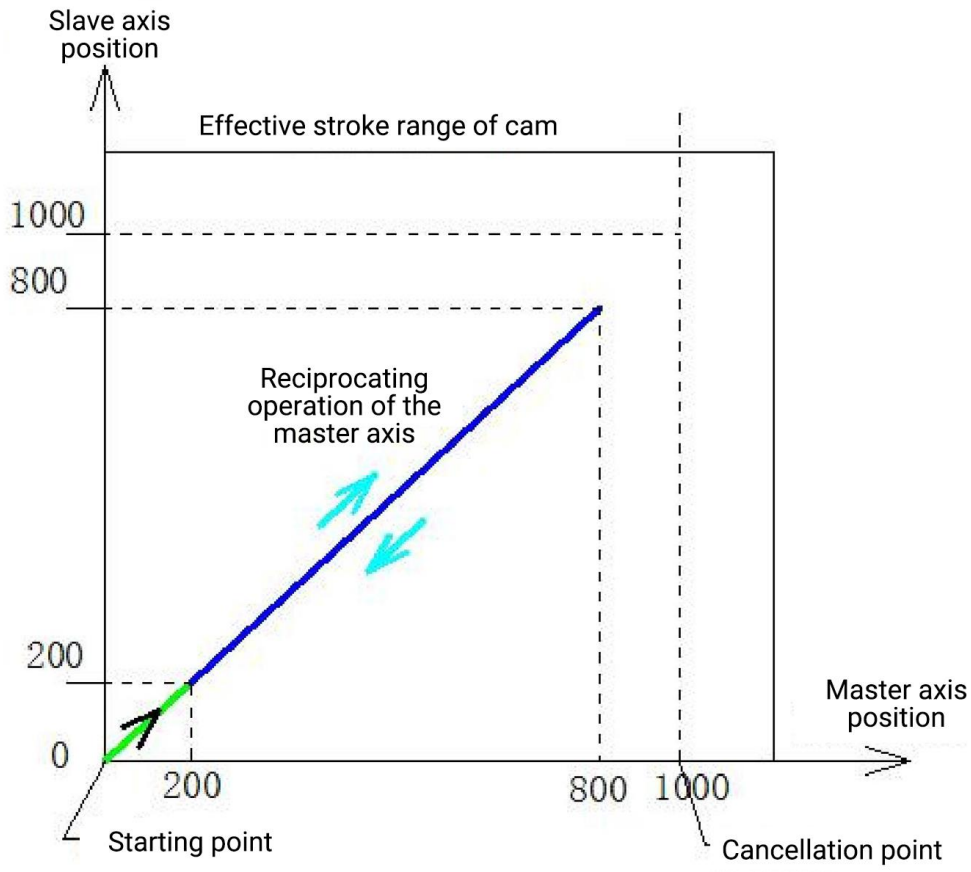
5.4.6.1.4 Example

(1) Single immediate cam start (ucMode = 0)

Define ArrPos as the master/slave axis position array. The example program is as follows (ST language). By selecting three coordinate points on a straight line, this program realizes that the linkage operation trajectory of the cam master axis and slave axis is a straight line with an inclination angle of 45 degrees.

After the cam instruction is put into operation, the cam function is immediately started with the current master axis Mpos value as the starting position, and the Hmc_Move instruction is sent to the cam master axis to drive the cam master axis for moving. At this time, the cam slave axis makes corresponding linkage actions. When the master axis reciprocates within the effective stroke range of the cam, the slave axis also performs reciprocating linkage actions (the starting positions of the master and slave axes are both 0).

```
(**Master/slave axis position array assignment**)

ArrPos[0, 0] := 0:

ArrPos[1, 0] := 0; (*Point 1 coordinates*)

ArrPos[0, 1] := 500;

ArrPos[1, 1] := 500; (*Point 2 coordinates*)

ArrPos[0, 2] := 1000;

ArrPos[1, 2] := 1000; (*Point 3 coordinates*)

pArrPos := ADR(ArrPos): (*Get the first address of master/slave axis position array ArrPos*)

HMC_Cam (Axisl.AxisID, Axis0.AxisID, pArrPos, 3, 1, 0, 0); (*Send a single immediate cam start instruction, with Axis 0 as the master axis, Axis 1 as the slave axis, and the cam master axis stroke range as *)

(*Send the Move instruction to the master axis to drive the cam master axis to reciprocate. At this time, the cam slave axis will perform corresponding linkage actions*)

HMC_Move (Axis0. AxisID, 800);

HMC_Move (Axis0. AxisID, -600);

HMC_Move (Axis0. AxisID, 600);

HMC_Move (Axis0. AxisID, -600);

HMC_Move (Axis0. AxisID, 600);
```



Linkage Trajectory in Single Immediate Cam Start

(2) Single fixed position cam start (ucMode= 1)

The example program is as follows (ST language). The trajectory of the cam master axis and slave axis in this program is a straight line with an inclination angle of 45 degrees. Send the Hmc_Move instruction to the cam master axis to drive the cam master axis to move. When the cam master axis reaches the starting position, the cam instruction starts running, and the cam slave axis starts to perform corresponding linkage actions. When the master axis crosses the effective stroke boundary of the cam, the cam function is automatically canceled (the starting positions of the master and slave axes are both 0).

(**Master/slave axis position array assignment**)

ArrPos[0, 0] :=0;

ArrPos[1, 0] := 0; (*Point 1 coordinates*)

ArrPos[0, 1] :=500;

ArrPos[1, 1] := 500; (*Point 2 coordinates*)

ArrPos[0, 2] := 1000;

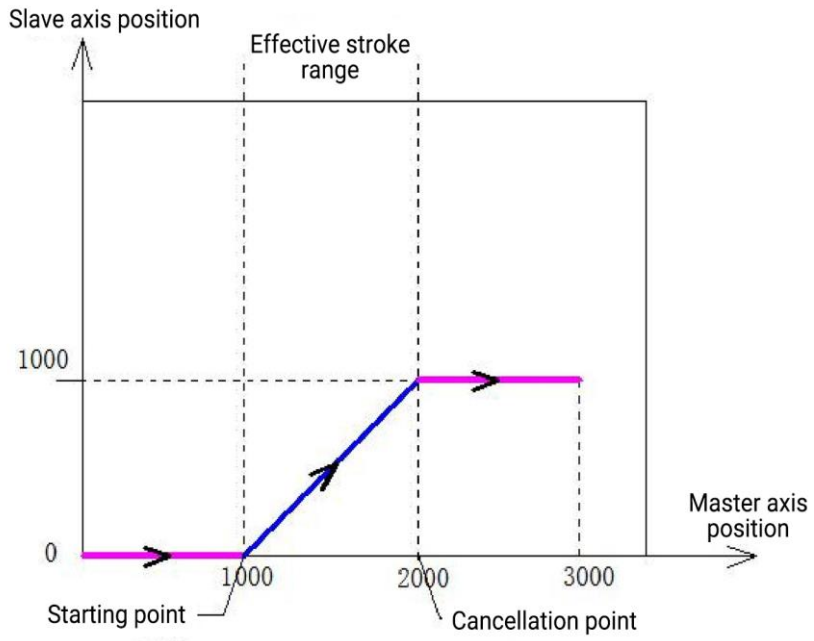
ArrPos[1, 2] := 1000; (*Point 3 coordinates*)

(*Send a single in-position cam start instruction, with Axis 0 as the master axis, Axis 1 as the slave axis, and the starting position as 1000*)

HMC_Cam (Axis1.AxisID, Axis0.AxisID, ADR(ArrPos2),101,1,1,1000);

(*Send the Move instruction to the master axis to drive the cam master axis to move in one way. Single cam start - running - cancellation*)

HMC_Move (Axis0.AxisID, 3000);



Linkage Trajectory in Single Fixed-position Cam Start

3) Single event cam start (ucMode=2)

The example program is as follows (ST language). The linkage trajectory of the cam master axis and slave axis in this program is a sine curve. Send the Hmc_Move instruction to the cam master axis to drive the cam master axis to move. When the rising edge of DI of channel 0 comes, the high-speed capture channel 0 is triggered. The cam instruction starts running with the captured master axis position as the starting position, and the cam slave axis starts to make corresponding linkage actions. When the master axis crosses the effective stroke boundary of the cam, the cam function is automatically canceled.

(**Generate master/slave axis position array**)

delta := 3.1415926*2/100;

FOR n :=0 TO 100 DO

ArrPos2[0,n] := (delta*n)*150;

ArrPos2[1,n] := SIN(delta*n)*1000;

END_FOR

(*Configure high-speed capture channel: trigger channel 0 DI, trigger rising edge, and the axis to be captured is axis 0*)

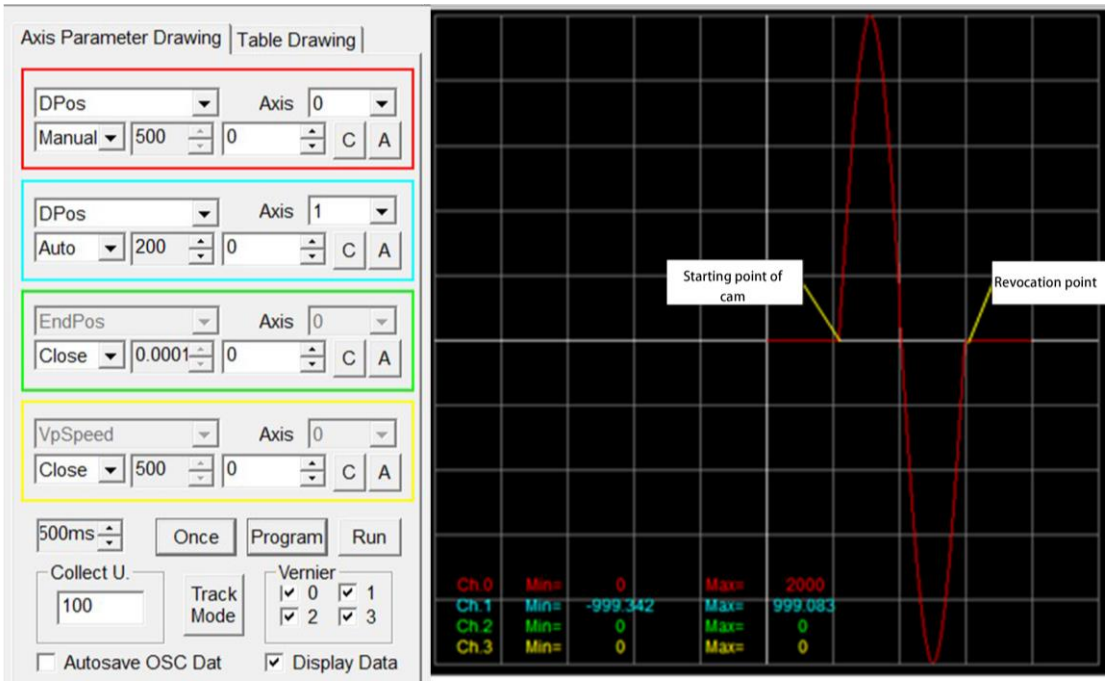
HMC_CaptureConfig (0, 0, 1, 0, Axis0.AxisID, 0, 0, 0);

(*Send a single event cam start instruction, with axis 0 as the master axis, axis 1 as the slave axis, and the high-speed capture channel number as 0*)

HMC_Cam (Axis1, AxisID, Axis0, AxisID, ADR (ArrPos2), 101, 1, 2, 0);

(*Send the Move instruction to the master axis to drive the cam master axis to move in one way. *)

HMC_Move (Axis0.AxisID, 2000);



Linkage Trajectory in Single Event Cam Start

(4) Single DI cam start (ucMode = 3)

The example program is as follows (ST language). The linkage trajectory of the cam master axis and slave axis in this program is a sine curve. When the rising edge of DI of channel 0 comes, the cam instruction starts running with the current master axis Mpos position as the starting position. At this time, if the Hmc_Move instruction is sent to the cam master axis to drive the cam master axis to move, the cam slave axis will start to perform corresponding linkage actions.

(**Generate master/slave axis position array**)

delta := 3.1415926*2/100;

FOR n:=0 TO 100 DO

ArrPos2[0,n] := (delta*n)*150;

ArrPos2[1,n] := SIN(delta*n)*1000;

END_FOR

(*Send a single DI cam start instruction, with axis 0 as the master axis, axis 1 as the slave axis, and the DI channel number as 0*)

HMC_Cam (Axis1, AxisID, Axis0, AxisID, ADR(ArrPos2),101,1,3,0);

(5) Cyclic immediate cam start (ucMode=4)

The example program is as follows (ST language). The trajectory of the cam master axis and slave axis in this program is a straight line with an inclination angle of 45 degrees.

After the cam instruction is sent, the cam function is immediately started with the current master axis Mpos value as the starting position, and the Hmc_Move instruction is sent the cam master axis to drive the cam master axis to reciprocate. At this time, the cam slave axis performs corresponding linkage actions. When the master axis completes a cam stroke, it enters the next cam stroke, and the slave axis makes incremental calculations from the end point of the previous cam stroke as the starting point and performs periodic linkage actions (the starting positions of the master and slave axes are both 0).

(**Master/slave axis position array assignment**)

ArrPos[0,0] :=0;

ArrPos[1, 0] := 0; (*Point 1 coordinates*)

ArrPos[0,1] := 500;

ArrPos[1, 1] := 500; (*Point 2 coordinates*)

ArrPos[0,2] := 1000;

ArrPos[1, 2] := 1000; (*Point 3 coordinates*)

HMC_Cam (Axis l, AxisID, Axis0, AxisID, ADR (ArrPos), 3, 1, 0, 0); (*Send a single immediate cam start instruction, with axis 0 as the master axis, axis 1 as the slave axis, and the cam master axis stroke range as *)

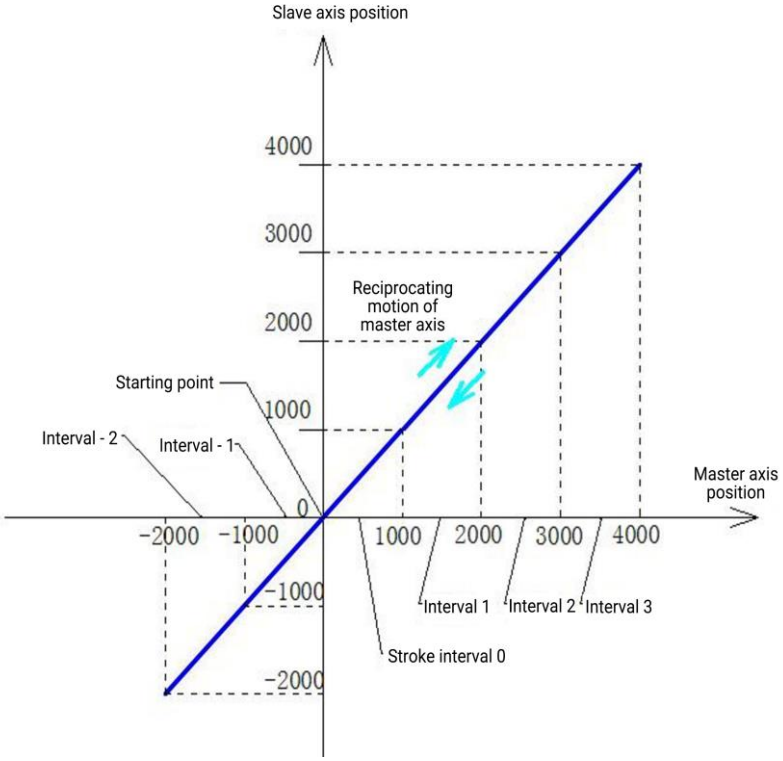
(*Send the Move instruction to the master axis to drive the cam master axis to reciprocate. At this time, the cam slave axis will perform corresponding linkage actions*)

HMC_Move (Axis0.AxisID, 4000);

HMC_Move (Axis0.AxisID, -6000);

HMC_Move (Axis0.AxisID, 6000);

HMC_Move (Axis0.AxisID, -4000);

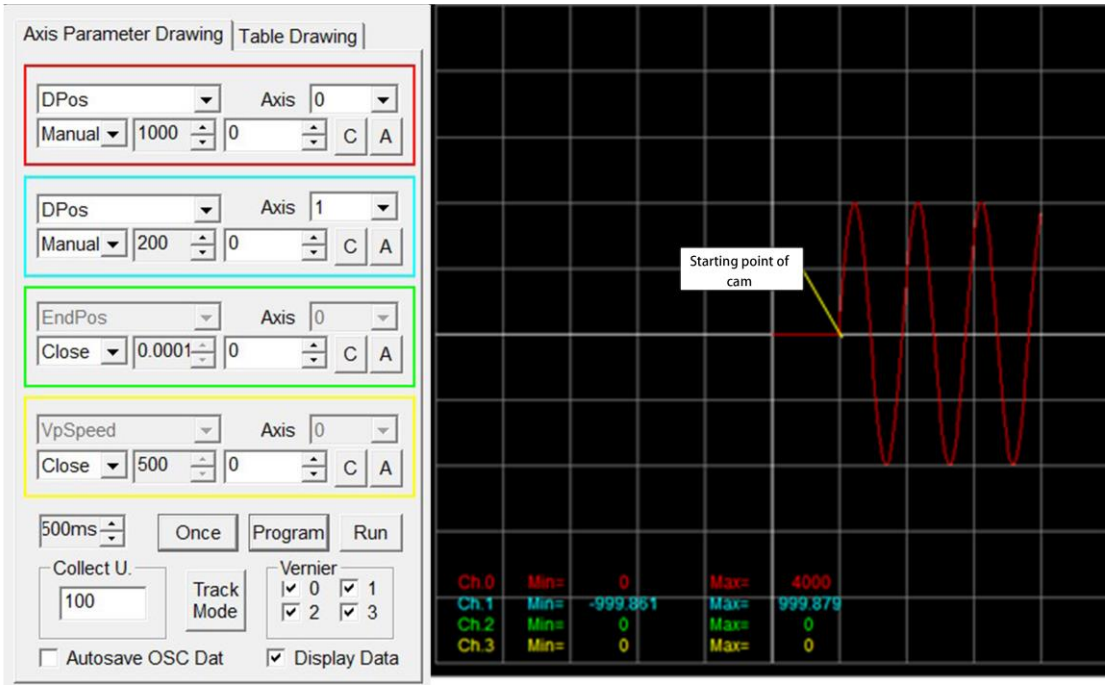


Linkage Trajectory in Cyclic Immediate Cam Start

(6) Cyclic fixed-position cam start (ucMode= 5)

The example program is as follows (ST language). The linkage trajectory of the cam master axis and slave axis in this program is a sine curve. Send the Hmc_Move instruction to the cam master axis to drive the cam master axis to move. When the cam master axis reaches the starting position, the cam instruction starts running, and the cam slave axis starts to perform corresponding linkage actions (the starting position of both the master and slave axes is 0).

```
(**Generate master/slave axis position array**)
delta :=3.1415926*2/100;
FOR n:=0 TO 100 DO
ArrPos2[0,n] :=(delta*n)*150;
ArrPos2[1,n] := SIN(delta*n)*1000;
END_FOR
(*Send a single event cam start instruction, with axis 0 as the master axis, axis 1 as the slave axis, and the fixed starting position as 1000*)
HMC_Cam (Axis1, AxisID, Axis0, AxisID, ADR(ArrPos2),101,1,5,1000);
(*Send the Move instruction to the master axis to drive the cam master axis to move. *)
HMC_Move (Axis0.AxisID, 4000);
```



Linkage Trajectory of Cyclic Fixed-position Cam Start

(7) Cyclic event cam start (ucMode=6)

Except for the different cancellation conditions after startup, the rest is similar to the single event cam start. Please refer to the example in (3).

(8) Cyclic DI cam start (ucMode=7)

Except for the different cancellation conditions after startup, the rest is similar to the single DI cam start. Please refer to the example in (4).

5.4.6.2 HMC_SplineCam (Spline Cam)

5.4.6.2.1 Function

1. The function of this instruction is similar to that of HMC_Cam. The difference is that HMC_SplineCam uses spline curve fitting to connect the master/slave axis position points, while HMC_Cam uses polyline connection. Therefore, the trajectory generated by the spline cam is smoother and there is no jump in speed during the motion.

2. For other related instructions and instruction examples, please refer to HMC_Cam (Electronic cam).

5.4.6.2.2 Parameter

Parameter	Statement	Data Type	Description
ucSlaveAxis	Input/Output	USINT	Slave axis number
ucMasterAxis	Input	USINT	Master axis number
pArrPos	Input	POINTER TO LREAL	First address of position array (two-dimensional) In the two-dimensional array

Parameter	Statement	Data Type	Description
			pArrPos[2,uiPosNum], pArrPos[0] is the master axis position array, and pArrPos[1] is the slave axis position array
uiPosNum	Input	UDINT	Number of positions, which shall be greater than or equal to 3
dScale	Input	LREAL	Cam slave axis position data scale factor. Actual position output of cam slave axis = set output of slave axis position array*dScale
ucMode	Input	USINT	0: Single immediate start 1: Single fixed position start 2: Single event start 3: Single DI start 4: Cyclic immediate start 5: Cyclic fixed-position start 6: Cyclic event start 7: Cyclic DI start
dTrigMes	Input	LREAL	Startup information: 1. When the cam starts in in-position start mode, this value indicates the starting position of the master axis 2. When the cam is triggered in event start mode, it indicates a certain high-speed capture channel number 3. When the cam is triggered in DI (1) start mode, it indicates the DI channel (0~63)

5.4.6.2.3 Instruction type

Axis motion cache instructions.

5.4.6.3 HMC_GetCamMasterPosIndex (Getting Electronic Cam Master Axis Position Index)

5.4.6.3.1 Function

It is used to get the position index of the currently running cam master axis (the interval index of the current cam master axis position in the cam master axis position array, which shall be 0 to uiPosNum-1). If the current instruction is not a cam, the function returns 0.

5.4.6.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BOOL	Axis number

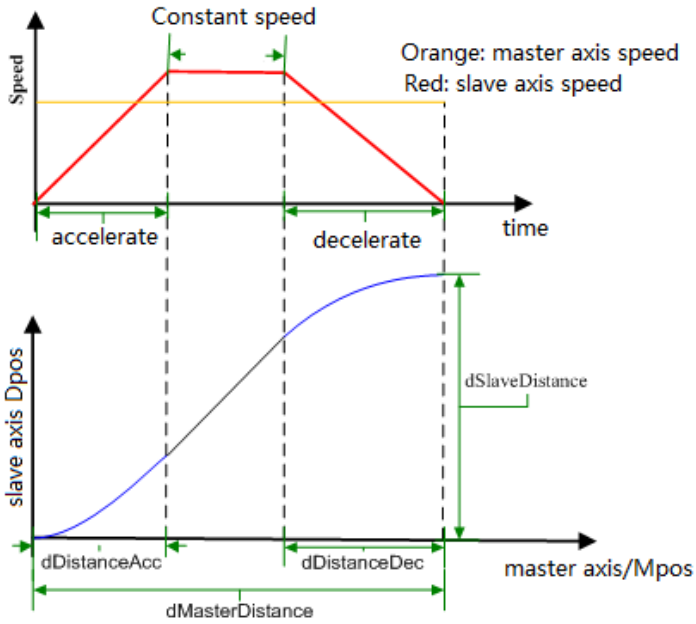
5.4.6.3.3 Instruction type

It is a non-axis motion cache instruction.

5.4.6.4 HMC_ClcMoveLinkData (Follow-up Shear Operation)

5.4.6.4.1 Function

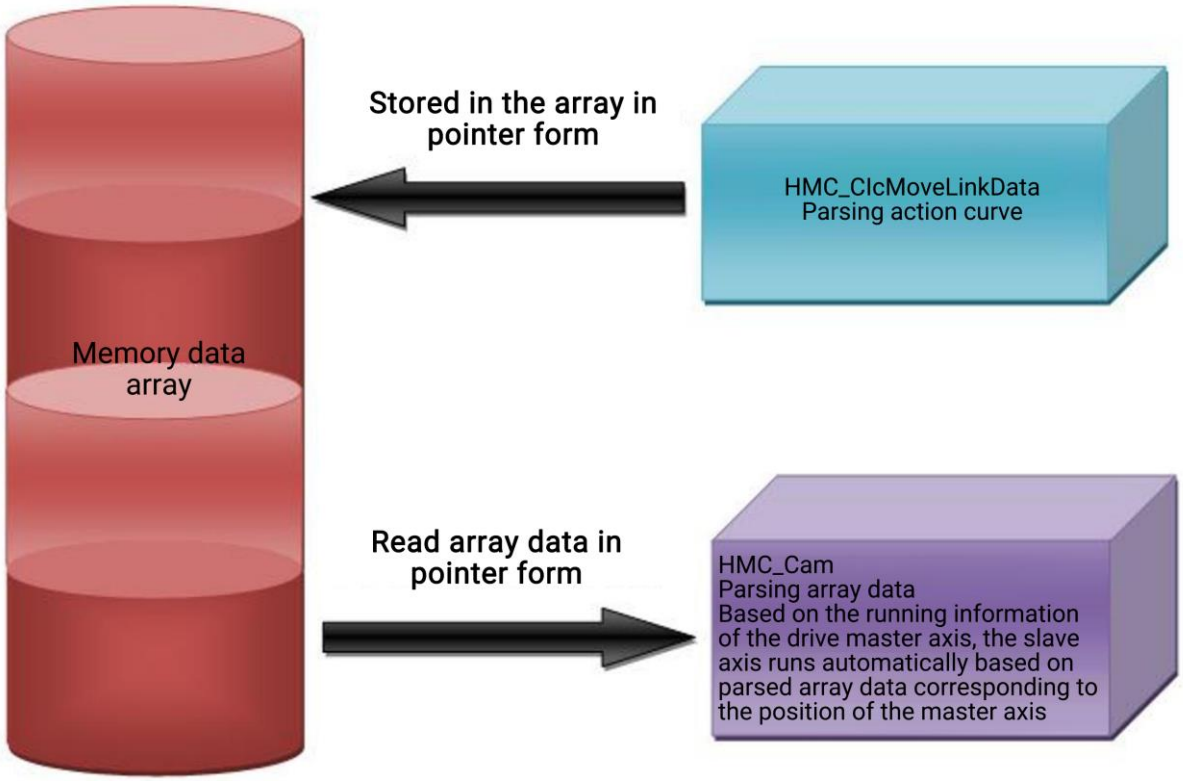
This instruction can generate master-slave axis position data according to the set relationship. The position data is saved in an array and can be used by the cam instruction to control the slave axis to follow the master axis and move according to a certain pattern. If the master-slave axis position data generated by this instruction is used, when the cam master axis moves at a constant speed, the master-slave axis curve is as shown in the figure below. The motion has separate variable acceleration and deceleration stages.



Master/Slave Axis Diagram

By setting the slave axis displacement dSlaveDistance, the master axis displacement dMasterDistance, the master axis displacement dDistanceAcc during the slave axis acceleration stage, and the master axis displacement dDistanceDec during the slave axis deceleration stage, HMC_ClcMoveLinkData can generate the master-slave axis position data that meets the requirements shown in the figure above. The data is stored in the array pArrPos. The number of master/slave axis positions is specified by the parameter uiPosNum.

When HMC_ClcMoveLinkData is used in conjunction with the HMC_Cam instruction, an array is used to transfer the master/slave axis position data. First, place the master/slave axis position data calculated by the curve through the HMC_ClcMoveLinkData instruction in the array, then call the array data through the HMC_Cam instruction, and finally, drive the master axis to move. The slave axis uses the HMC_Cam instruction to analyze the position data in the array to follow the position of the master axis.



HMC_ClcMoveLinkData and HMC_Cam Data Flow

This instruction can be used in follow-up shearing scenarios, combined with the cam instruction HMC_Cam, to quickly generate the action flow required for follow-up shearing, without the need for complex intermediate calculations and programming.

5.4.6.4.2 Parameter

Parameter	Statement	Data Type	Description
dSlaveDistance	Input	LREAL	Slave axis motion distance
dMasterDistance	Input	LREAL	Master axis motion distance (must be positive)
dDistanceAcc	Input	LREAL	Positive distance moved by the master axis during the acceleration stage of the slave axis, non-negative
dDistanceDec	Input	LREAL	Forward distance moved by the master axis during the deceleration stage of the slave axis, non-negative
pArrPos	Input	POINTER TO LREAL	Position array (2D) In the two-dimensional array pArrPos[2][uiPosNum], pArrPos[0] is the master axis position array, and pArrPos[1] is the slave axis position array
uiPosNum	Input	UDINT	Number of positions, which shall be greater than or equal to 6

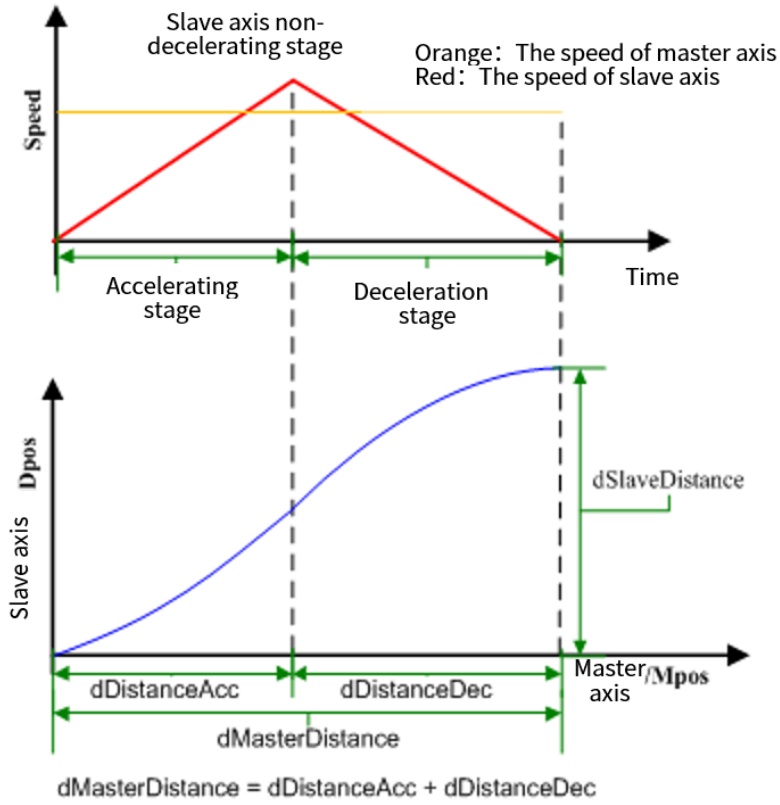
5.4.6.4.3 Instruction type

It is a non-axis motion cache instruction.

Additional information:

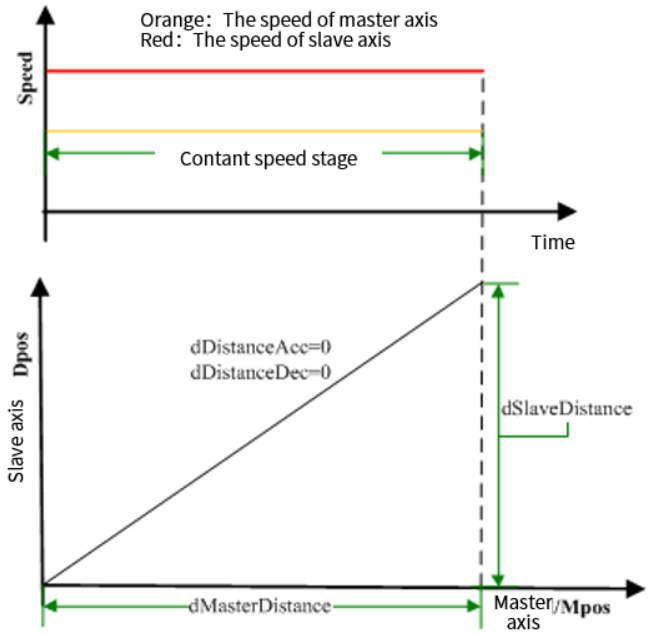
(1) All distances in this function are in user units. It must satisfy dDistanceAcc + dDistanceDec ≤ dMasterDistance, otherwise a parameter error will be returned.

(2) When dDistanceAcc + dDistanceDec = dMasterDistance, the cam slave axis in the figure below will not have a constant speed stage, and at this time, the cam master/slave axis curve is as follows:



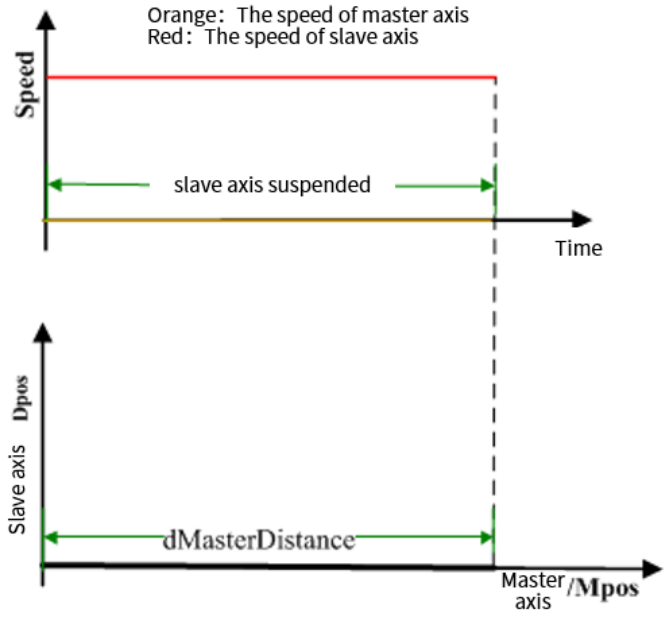
Master/Slave Axis Diagram

(3) When dDistanceAcc and dDistanceDec are both zero, that is, when there is no separation of acceleration and deceleration, the linkage at this time is equivalent to an electronic gear. The cam master/slave axis curve is as shown in the figure below:



Master/Slave Axis Diagram

(4) If dSlavetDistance is 0, when the cam master axis moves in the set area, the slave axis will pause and wait. The cam master/slave axis curve in the master_slave axis diagram is as shown in the figure below:



Master/Slave Axis Diagram

5.4.6.4.4 Example

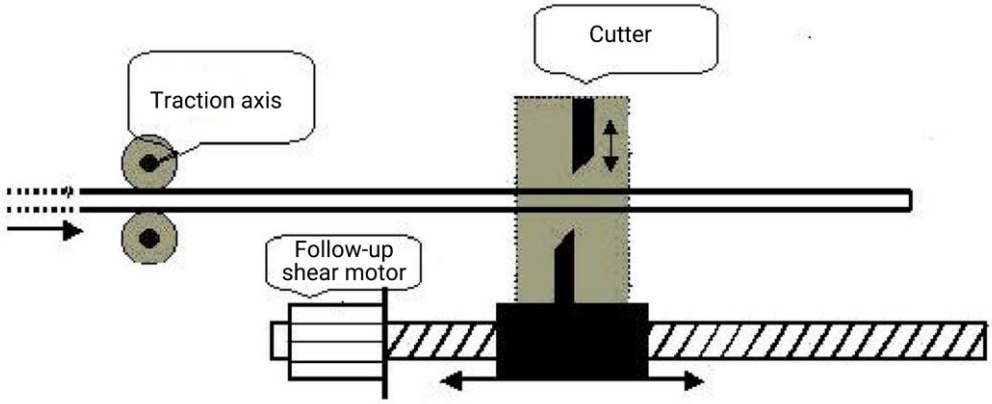
(1) Follow-up shear

In the traditional fixed-length material processing system, the feeding servo motor generally outputs the material to a set length and then stops the feeding motor, and then drives the processing mechanism (such as a cutter) for processing. Only after the processing is completed can the feeding continue. Such traditional devices include

steel plate leveling and rolling machines, fixed-length cutting devices of various types of pipes, and shakeout devices of textile machinery, etc. Since the feeding action needs to be stopped intermittently during processing, the production efficiency of this type of devices is low, and it can only be used in situations where the output of fixed-length materials can be stopped intermittently. In situations such as seamless steel pipe production lines, where the feeding device is an extruder, or in application scenarios that have high requirements for production efficiency, the device is not allowed to stop intermittently. The above-mentioned traditional devices cannot meet customer needs, so a follow-up shear solution needs to be introduced.

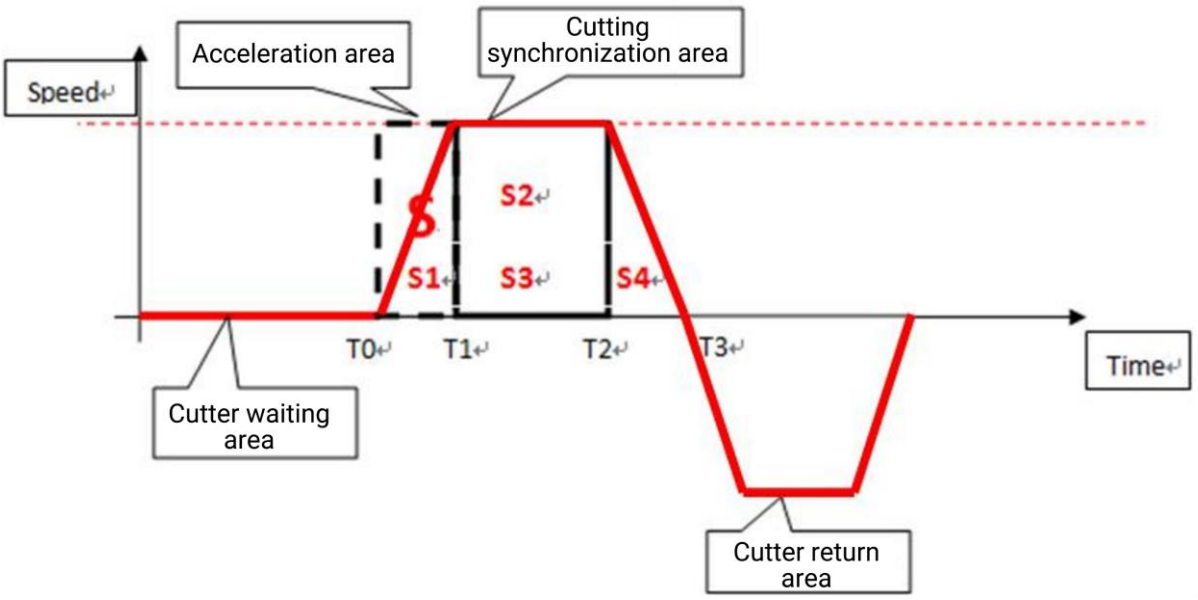
Follow-up shearing means that, during the material cutting process, the material is continuously transported while the cutting device makes reciprocating motion. By planning the speed and position of the cutting device, when the speed of the cutting device and the material are synchronized, the length of the material to be cut just meet the preset requirements. At this time, cut the material, then return to the waiting position and wait for a certain period of time. After that, the material is tracked and cut simultaneously, and the cycle starts again. The example is as described below.

The material runs at a constant speed from left to right through the traction axis. The cutter starts from the initial position and tracks the material to the synchronization area before cutting. After the cutting is completed, the cutter quickly returns to the initial position and waits for a period of time before continuing to the next action cycle.

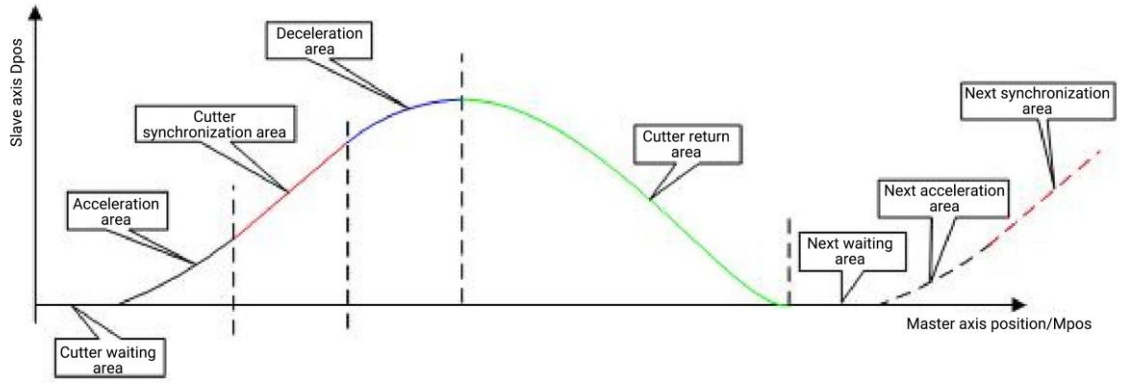


Follow-up Shear Plan Diagram

The speed curve that meets the requirements of follow-up shear is shown in the figure below, where the red dotted line is the traction axis speed, and the red solid line is the cutting axis speed. The follow-up shear action process can be realized through the HMC_ClcMoveLinkData and HMC_Cam instructions. Assume that the traction axis is the master axis of the system and the follow-up shear motor is the slave axis of the system.



Follow-up Shear Speed Planning Graph



Follow-up Shear Master/Slave Axis Displacement Curve

The cutting length is set to 100 m, the cutter moving stroke is 3 m, the screw lead is 30 mm, the speed of follow-up shear motor is 2000 r/min, and the speed ratio is 1:1. The motion trajectory is divided into the following parts: waiting area, acceleration area, synchronization area, deceleration area, and return area. In the constant speed area, the curve can be refined to match the cutter action. Assume that in the acceleration area (time T0~T1), the walking distance of the cutting axis is S1, and the walking distance of the traction axis is S, and the calculation shows: S=2S1; in the synchronization area (time T1~T2), the walking distance of the cutting axis at the cutting speed is S2 = the walking distance of traction axis S3; if the acceleration and deceleration times are equal, the walking distance of cutting axis S4 = S1 in the deceleration area (T2~T3 time), and the walking distance of traction axis is S.

From the known parameters, it is known that the cutting axis speed = 2000*30 = 60000 mm/min = 1000 mm/s. Assume that the acceleration and deceleration time is 1 s, and it can be calculated that:

S= 1000mm

S1=S4=500mm

S2=S3=2000 mm

Master axis displacement in the cutter waiting area = total cutting length - master axis displacement in the cutter's forward action area - master axis displacement in the cutter's reverse action area.

In this example, the acceleration, deceleration and constant speed used for the forward and reverse actions of the cutter are set to be the same, so the forward and reverse action times of the cutter are equal, therefore:

Master axis displacement in the cutter's reverse action area = master axis displacement in the cutter's forward action area = S+S2+S = 4000 (mm)

Slave axis displacement in the forward action area of the cutter = S1+S2+S4 = 3000 (mm)

Slave axis displacement in the reverse action area of the cutter = slave axis displacement in the forward action area of the cutter = -3000 (mm)

Master axis displacement in the cutter waiting area = 100000 – 4000 – 4000 = 92000 (mm)

Slave axis displacement in the cutter waiting area = 0

The procedure is as follows:

TASK1 (Task 1): Initialize axis parameters and follow-up shear data

(**** Use HMC_SetUnits to convert user units to mm (omitted) ****)

(****Use ADR to get the first address of the position array****)

add1 := ADR(Pos);

add2 := ADR(Pos1);

add3 := ADR(Pos2);

add4 := ADR(Pos3);

add5 := ADR(Pos4);

(****Generate master/slave axis position array****)

HMC_ClcMoveLinkData(0, 92000, 0, 0, add1, PosNum); (*Cutting waiting area*)

HMC_ClcMoveLinkData(500, 1000, 1000, 0, add2, PosNum); (*Cutting acceleration area*)

HMC_ClcMoveLinkData(2000, 2000, 0, 0, add3, PosNum); (*Cutting synchronization area*)

HMC_ClcMoveLinkData(500, 1000, 0, 1000, add4, PosNum); (*Cutting deceleration area*)

HMC_ClcMoveLinkData(-3000, 4000, 1000, 1000, add2, PosNum); (*Cutter return area*)

(****Use the master/slave axis position arrays of each stage to send the cam instruction****)

While 1 do (*Cyclic delivery follow-up shear cam*)

HMC_Cam(1, 0, add1, PosNum, 1, 0, 0); (*Cutting waiting area*)

HMC_Cam(1, 0, add2, PosNum, 1, 0, 0); (*Cutting acceleration area*)

Cut_Enable := HMC_Cam (1, 0, add3, PosNum, 1, 0, 0); (*Cutting sync area*)

HMC_Cam(1, 0, add4, PosNum, 1, 0, 0); (*Cutting deceleration area*)

HMC_Cam(1, 0, add5, PosNum, 1, 0, 0); (*Cutter return area*)

(****Perform cutting action in sync area****)

Camload_Ok := HMC_WaitCmdLoaded(Cut_Enable); (*Drive cutting in sync area*)

%QX0.0:= 1; (*Cutting action. It is assumed that the cutting action is controlled by switching quantity. *)

(****Retract the cutter after the synchronization area ends, and cutting ends****)

CamFinish_Ok := HMC_WaitCmd Finish(Cut_Enable); (*Stop cutting immediately after the synchronization area ends*)

%QX0.0 := 0; (Cutter retracted)

End_While;

(****Note: In actual application, the synchronization area can be further refined in order to protect the cutter. ****)

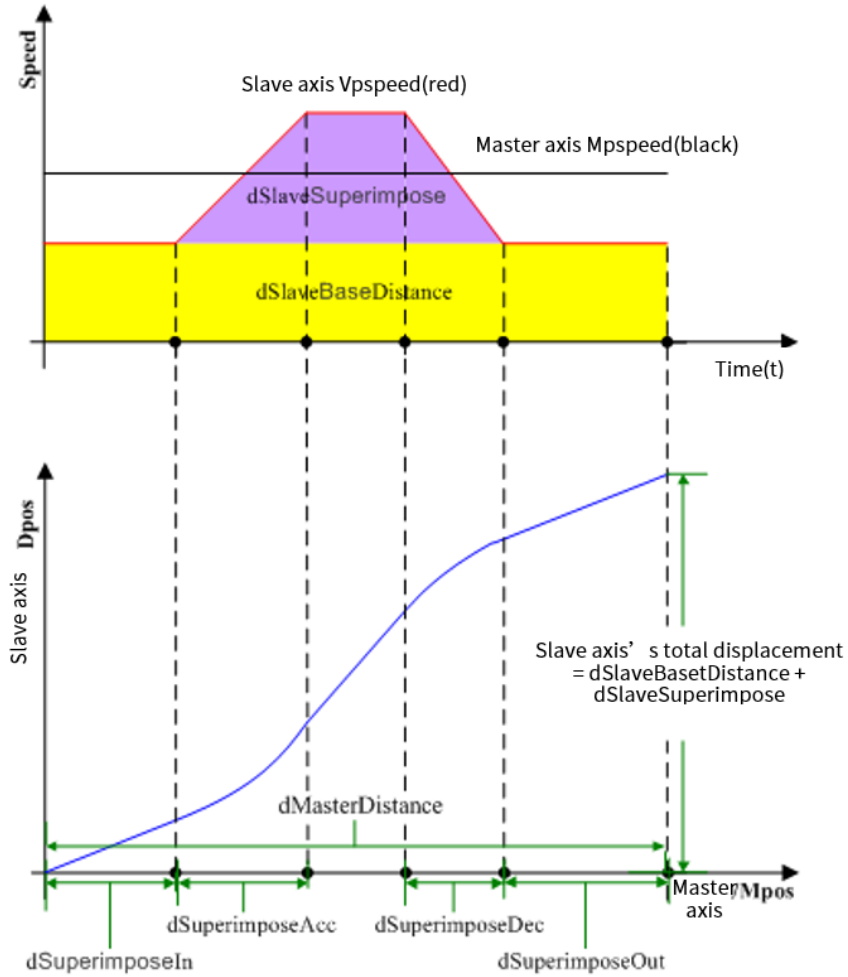
TASK2 (Task 2):

HMC_Forward(0); (*Master axis displacement start*)

5.4.6.5 HMC_ClcFlexLinkData (Flying Shear/Rotary Cutting Operation)

5.4.6.5.1 Function

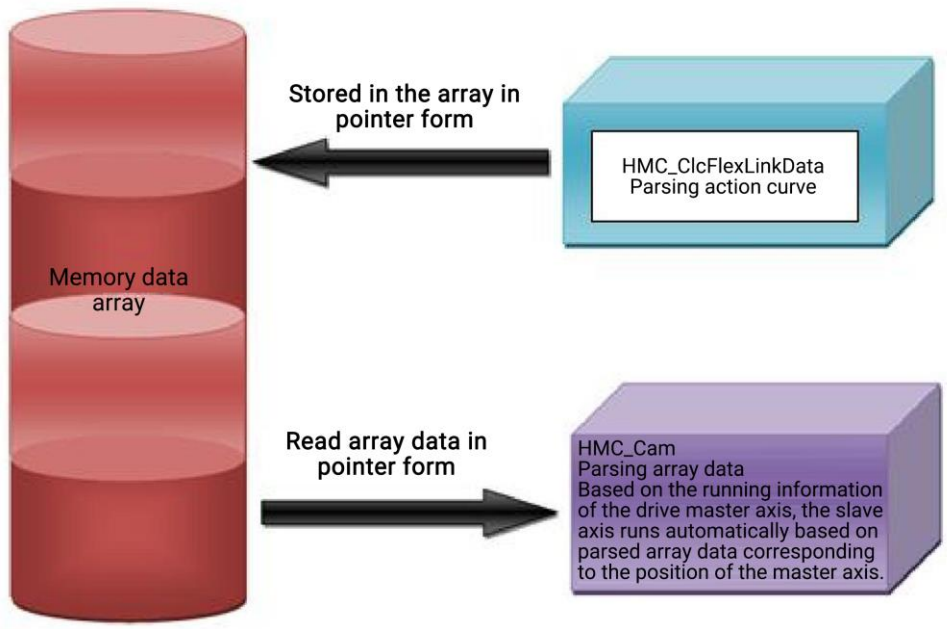
This instruction can generate master-slave axis position data according to the set relationship. The position data is saved in an array and can be used by the cam instruction to control the slave axis to follow the master axis and move according to a certain pattern. If the master-slave axis position data generated by this instruction is used, when the cam master axis moves at a constant speed, the master-slave axis curve is as shown in the figure below. The follow-up motion of the slave axis is superposed by the constant speed distance (dSlaveBasetDistance) and the variable speed distance with separated acceleration and deceleration (dSlaveSuperimpose).



Master/Slave Axis Diagram

By setting the required parameters (see the input parameter list), HMC_ClcMoveLinkData can generate the master/slave axis position data that meets the requirements shown in the figure above. The data is stored in the array pArrPos. The number of master/slave axis positions is specified by the parameter uiPosNum.

When HMC_ClcFlexLinkData is used in conjunction with the HMC_Cam instruction, an array is used to transfer the master/slave axis position data. First, place the master/slave axis position data calculated by the curve through the HMC_ClcFlexLinkData instruction in the array, then call the array data through the HMC_Cam instruction, and finally, drive the master axis to move. The slave axis uses the HMC_Cam instruction to analyze the position data in the array to follow the position of the master axis.



HMC_ClcFlexLinkData and HMC_Cam Data Flow

This instruction can be used in flying shearing/rotary cutting scenarios, combined with the cam instruction HMC_Cam, to quickly generate the action flow required for follow-up shearing, without the need for complex intermediate calculations and programming.

5.4.6.5.2 Parameter

Parameter	Statement	Data Type	Description
dSlaveBaseDistance	Input	LREAL	Slave axis constant speed follow-up distance
dSlaveSuperimpose	Input	LREAL	Slave axis superimposition motion distance
dMasterDistance	Input	LREAL	Master axis motion distance (must be positive)
dSuperimposeIn	Input	LREAL	Forward distance completed by the master axis when the superimposition motion (dSlaveSuperimpose) starts, non-negative
dSuperimposeOut	Input	LREAL	Remaining forward distance of the master axis when the superimposition motion (dSlaveSuperimpose) ends, non-negative
dSuperimposeAcc	Input	LREAL	Forward distance moved by the master axis during the acceleration stage of the superimposition motion, non-negative
dSuperimposeDec	Input	LREAL	Master axis movement during deceleration stage of superimposition motion
pArrPos	Input	POINTER TO LREAL	Position array (2D) In the two-dimensional array pArrPos[2][uiPosNum], pArrPos[0] is the master axis position array, and pArrPos[1] is

			the slave axis position array
uiPosNum	Input	UDINT	Number of positions, which shall be greater than or equal to 6

5.4.6.5.3 Instruction type

It is a non-axis motion cache instruction.

Additional information:

(1) The master axis motion distance (dMasterDistance) must be strictly positive, otherwise a parameter error will be returned.

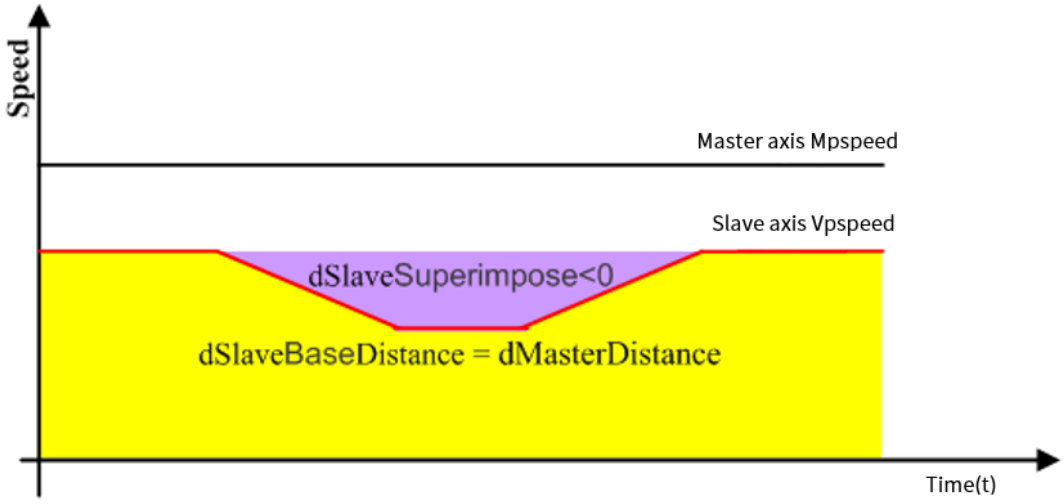
The slave axis displacement is equal to the sum of dSlaveBaseDistance and dSlaveSuperimpose.

(2) It shall satisfy:

$$dSuperimposeIn + dSuperimposeOut + dDistanceAcc + dDistanceDec \leq dMasterDistance$$

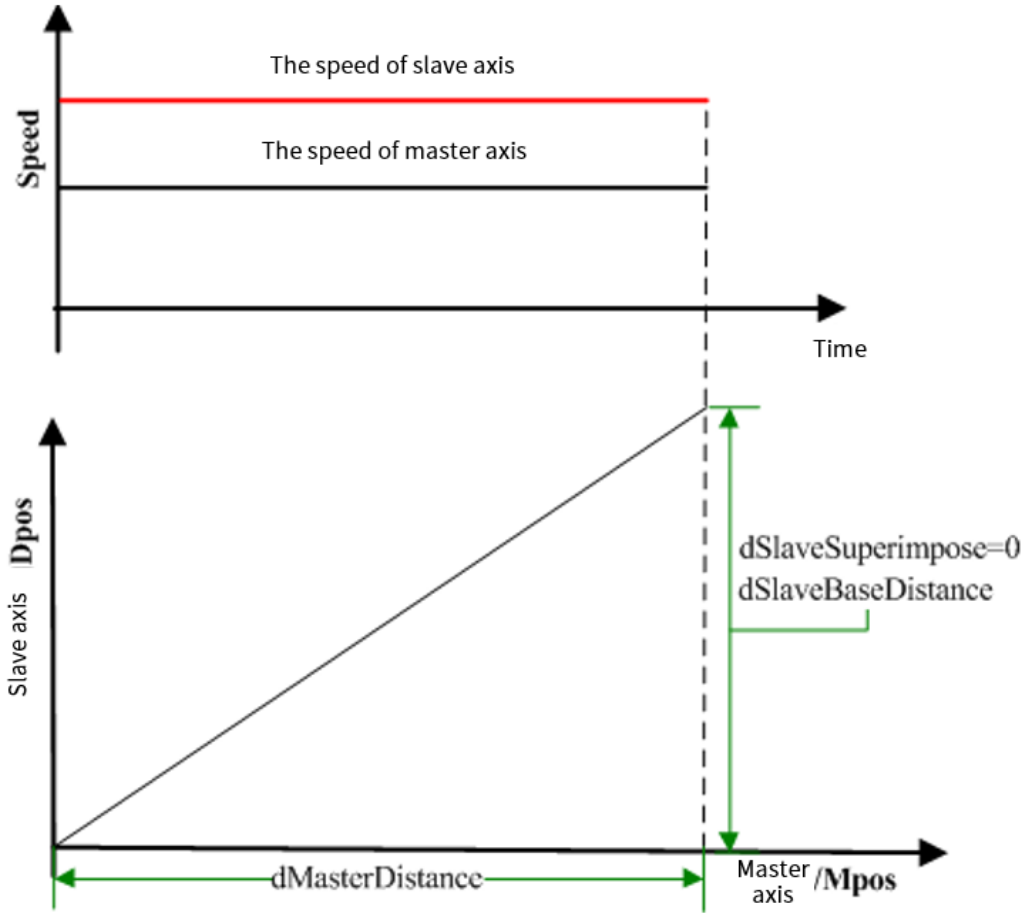
Otherwise, a parameter error (0xFFFFFFFF) is returned.

(3) dSlaveBaseDistance and dSlaveSuperimpose can be positive or negative. If both the parameters are 0, the slave axis will be in a waiting status until the master axis completes dMasterDistance, which is similar to the pause and wait of HMC_MoveLink.



Situation With a Negative Superimposition Distance of Slave Axis

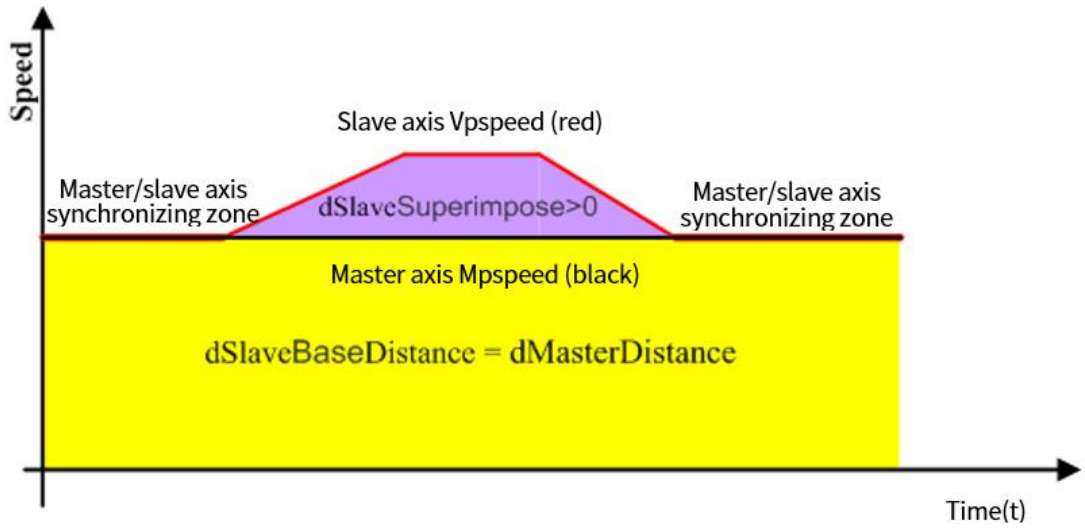
(4) When dSlaveSuperimpose is 0, the master and slave axes will move according to a certain speed proportional relationship, and the settings of parameters dSuperimposeIn, dSuperimposeOut, dSuperimposeAcc, and dSuperimposeDec will be invalid. The linkage at this time is equivalent to an electronic gear.

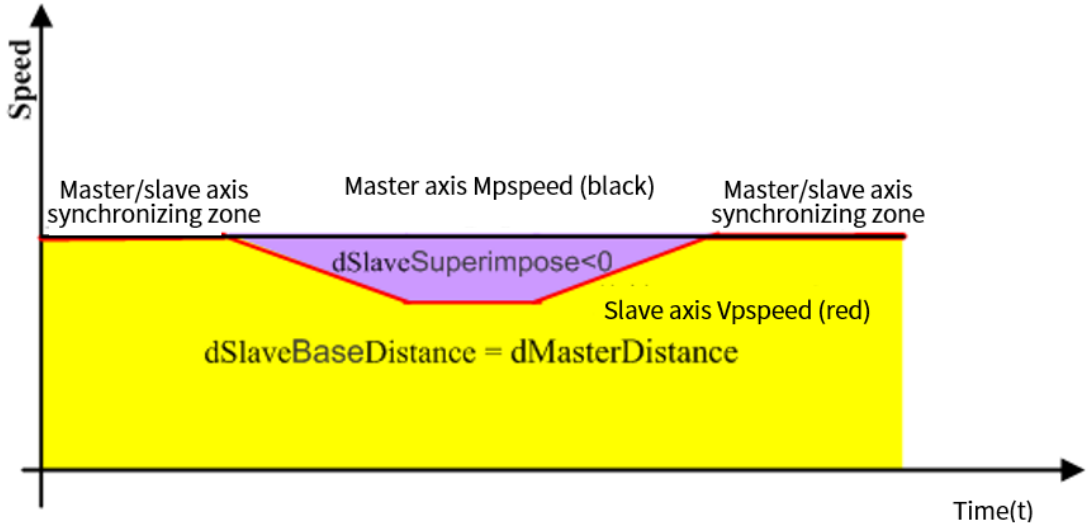


Motion Diagram

(5) When the slave axis's follow displacement at a constant speed is equal to the master axis displacement, that is:

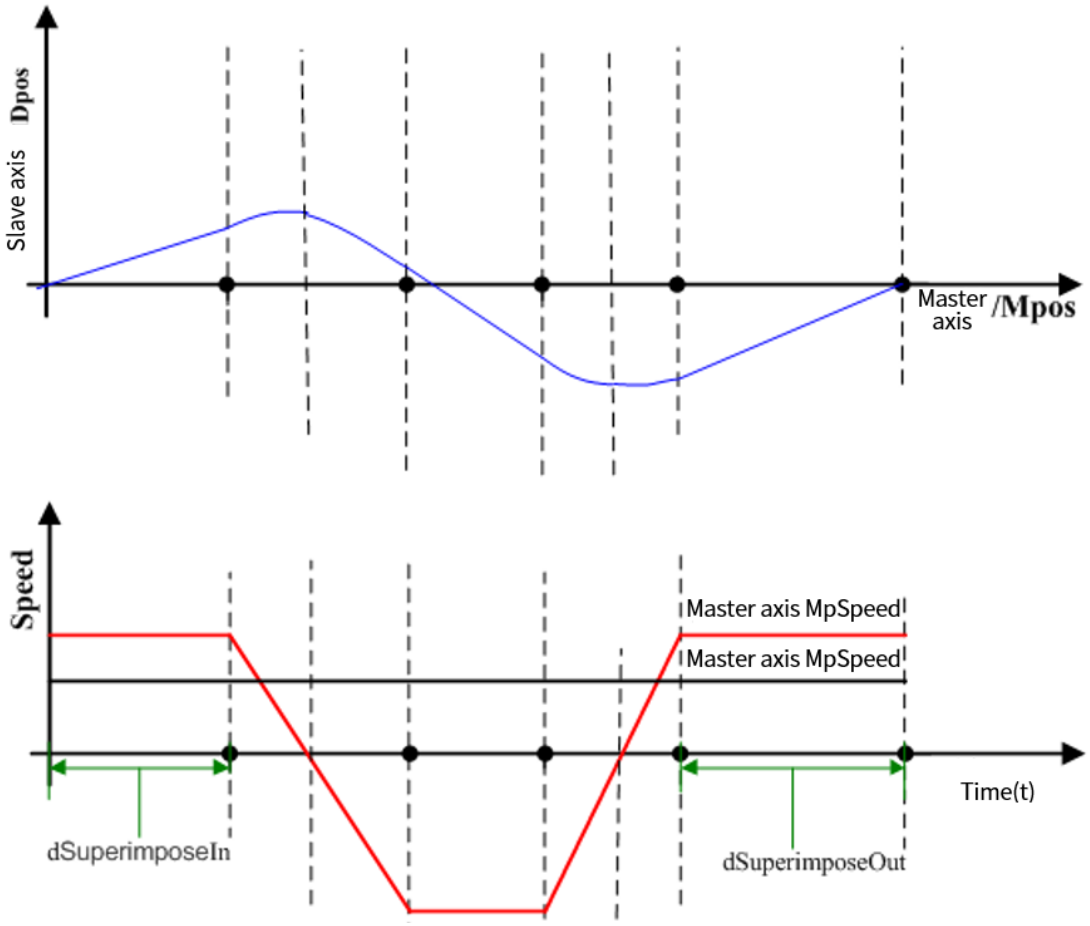
dSlaveBaseDistance = dMasterDistance, outside the slave axis superimposing area, the motion of the slave axis and the master axis will remain synchronized.





Motion Diagram

(6) When dSlaveBaseDistance + dSlaveSuperimpose = 0, and dSlaveBaseDistance is not 0, the slave axis returns to the starting point after superimposition compensation. This situation can also be used when it is necessary to return to the trigger position through reversal.

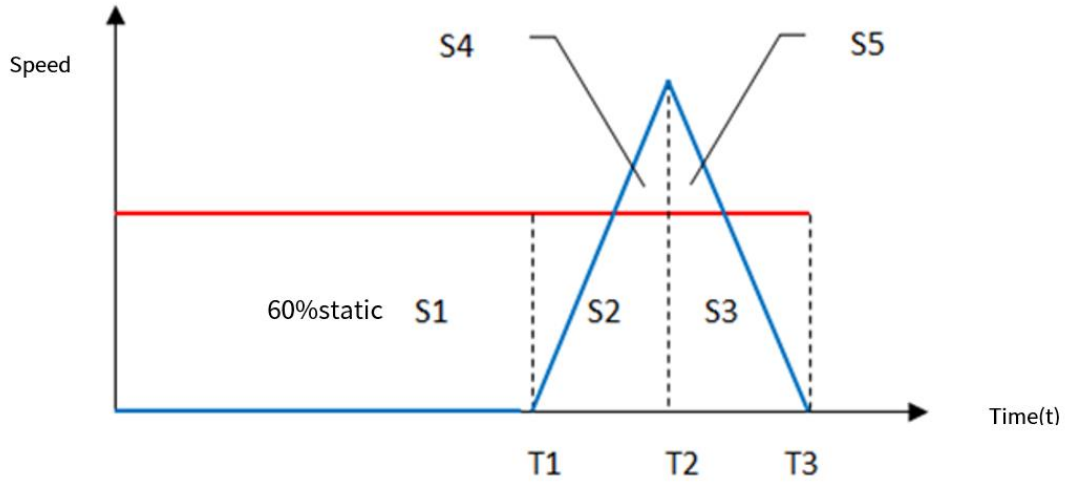


Master/Slave Axis Diagram

5.4.6.5.4 Example

(1) Generating a specified linkage action

In a cyclic action, 40% of the distance is smooth acceleration and deceleration, and the remaining part remains stationary. HMC_ClcFlexLinkData is now used to generate the action process.

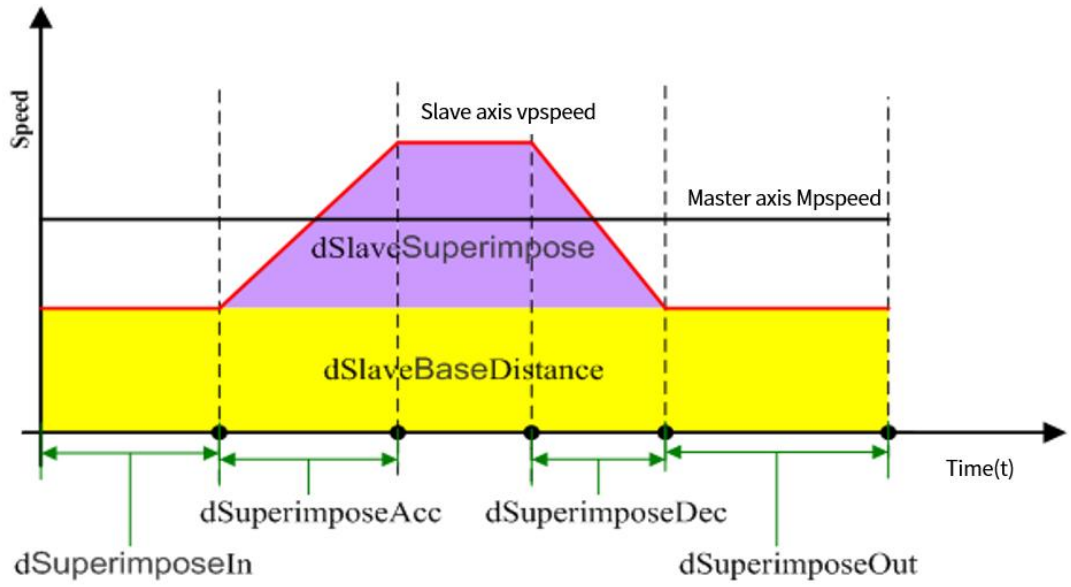


Motion Diagram

Assume that the moving distance of the master axis is 2000 mm, the moving distance of the slave axis is 1000 mm, and the acceleration and deceleration distances are equal. Based on the above data, the parameter values required by HMC_ClcFlexLinkData can be calculated.

As shown in the figure above, we divide the area of the master axis into 3 parts, the area S1 (accounting for 60% of the master axis area), which is 2000*0.6=1200; the displacement of the master axis during the T1-T2 period is the area S2, while that of the slave axis during the T1-T2 period is the acceleration process S4; the displacement of the master axis during the T2-T3 period is represented by the area S3, while that of the slave axis during the T2-T3 period is the deceleration process S5. Assuming that the acceleration and deceleration time of the slave axis is the same, then: slave axis displacement S4=S5=500 (half of the blue triangle), and master axis displacement S2=S3=(2000-1200)/2=400.

Comparing the original graphics of the HMC_ClcFlexLinkData function, since the slave axis only has acceleration and deceleration, but no reference speed, the area of the yellow part in the figure below is dSlaveBaseDistance=0. The area of the purple part of the slave axis is the area of acceleration, deceleration, and constant speed of the slave axis. Since the slave axis does not have a constant speed part, but only has acceleration and deceleration,



Motion Diagram

dSlaveBaseDistance=0;

dSlaveSuperimpose= 1000;

dSuperimposeIn=S1= 1200;

dSuperimposeOut=0;

dSuperimposeAcc=dSuperimposeDec=S2=400.

The specific program is as follows:

(****Initialization program is as follows****)

(*Use HMC_SetUnits to convert user units to mm (omitted) *)

DR_RESULT := HMC_DriveEnable(1); (*Signal enable operation*)

X_TYPE := HMC_SetAxisType(0, 0); (*Set axis0 attribute*)

X_UNIT := HMC_SetUnits(0, 100); (*Assume that 100 pulses correspond to 1 mm*)

AXIS0.SPEED := 100; (*In mm/s *)

AXIS0.ACCEL := 1000;

AXIS0.DECEL := 1000;

Y_TYPE := HMC_SetAxisType(1, 0); (*Set axis1 attribute*)

Y_UNIT := HMC_SetUnits(1, 100);

AXIS1.SPEED := 100; (*In mm/s *)

AXIS1.ACCEL := 1000;

AXIS1.DECEL := 1000;

Pt_array := ADR(arraympos);

NumOfPos := 100;

(*The main program is as follows*)

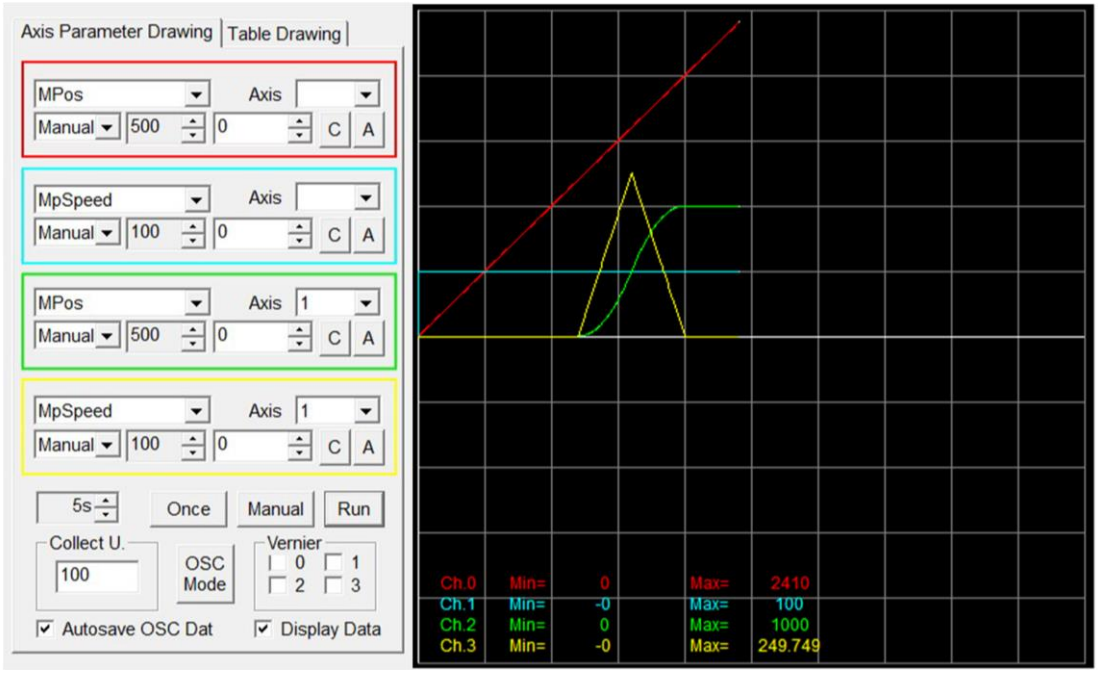
HMC_ClcFlexLinkData(0, 1000, 2000, 1200, 0, 400, 400, Pt_array, NumOfPos);

fff := HMC_TriggerScope(); (*Oscilloscope enabled*)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0); (*Send to cam, and cycle starts immediately*)

X_RESULT := HMC_Forward(0); (*Master axis starts*)

The running results in the controller are as follows (the cyan and yellow curves are the master and slave axis speeds respectively):



Oscilloscope Display Results

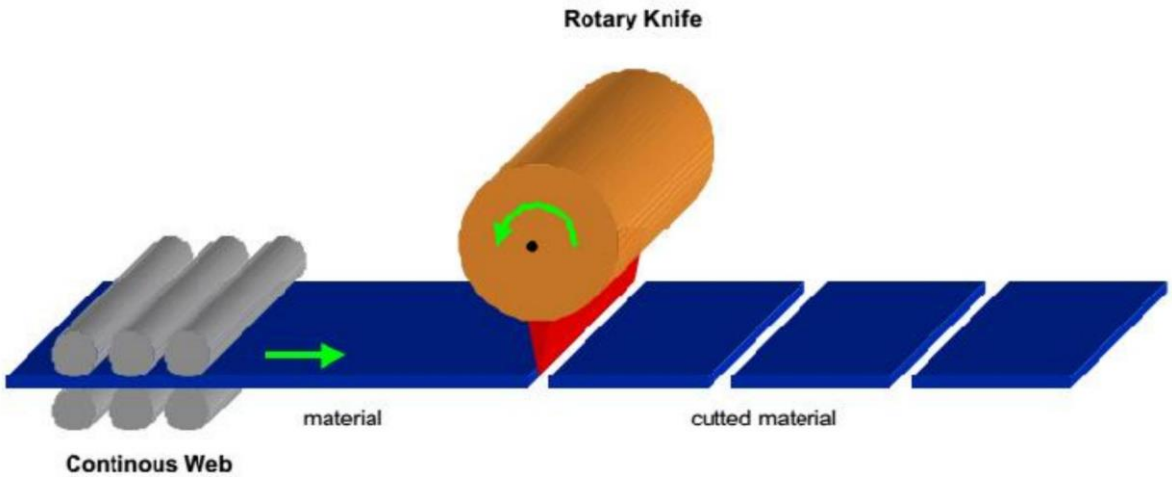
(2) Flying shear/rotary cutting operation

Flying shears are often used in packaging, papermaking, steel rolling, and other production lines. In the continuous rolling billet workshop or small section steel workshop, it is placed at the rear of the rolling line to cut the rolled pieces into fixed sizes or only cut the ends. Various types of flying shears are equipped in the cross-cutting unit, heavy-cutting unit, galvanizing unit and tin-plating unit in the cold and hot strip steel workshops to cut the strip steel into fixed lengths or into steel coils with specified weights. The widespread use of flying shears is conducive to the rapid development of papermaking, packaging, and steel rolling production lines into high-speed and continuous directions. Therefore, it is one of the important links in the development of high-speed production lines.

In the rotary cutting process, the action of the cutter has a synchronous area and a non-synchronous area. The area where the cutter cuts the material is the synchronous area. During the cutting process, the cutter head moves

synchronously with the material being cut to ensure the cutting quality and protect the tool at the same time. After the cutting is completed, the cutter roller makes accelerated or decelerated motions to adjust the cutter head for the next shear based on different cutting lengths, which is the non-synchronous area.

Using the HMC_ClcFlexLinkData instruction, we do not need to use complex mathematical expressions to solve for position points. The underlying algorithm is automatically completed by the controller. We only need to plan the distances for synchronization, acceleration, and deceleration, so that the master and slave axis position data required for flying shear/rotary cutting motion can be automatically generated. Then, we only need to use the position data in HMC_Cam.



Flying Shear/Rotary Cutting Diagram

Assume that the circumference of the cutter roller is 600 mm, and the length of the cutting synchronization area is 150 mm (75 mm on both sides).

Known: dSuperimposeIn=dSuperimposeOut=75 mm;

Due to the existence of synchronization area, hence: dSlaveBaseDistance=dMasterDistance=length of material to be cut.

According to the required cutting length, it can be divided into three situations: short material cutting, medium material cutting, and long material cutting, which can be achieved by adjusting the slave axis superimposition distance dSlaveSuperimpose (cutter roller circumference = dSlaveBaseDistance + dSlaveSuperimpose).

The initialization program is as follows:

```
DR_RESULT := HMC_DriveEnable( 1); (*Signal enable operation*)

X_TYPE := HMC_SetAxisType(0, 0); (*Set axis0 attribute*)

X_UNIT := HMC_SetUnits(0, 100); (*Assume that 100 pulses correspond to 1 mm*)

AXIS0.SPEED := 100; (*In mm/s *)

AXIS0.ACCEL := 1000;

AXIS0.DECEL := 1000;
```

Y_TYPE := HMC_SetAxisType(1, 0); (*Set axis1 attribute*)

Y_UNIT := HMC_SetUnits(1, 100);

AXIS1.SPEED := 100; (*In mm/s *)

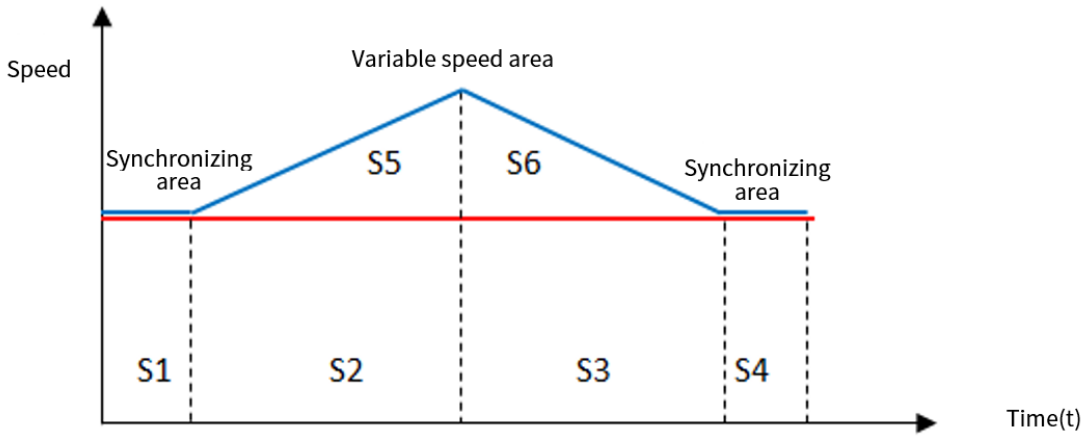
AXIS1.ACCEL := 1000;

AXIS1.DECEL := 1000;

Pt_array := ADR(arraympos) (*Get the array pointer, and the array dimension shall be [2,NumOfpos]*)

NumOfPos := 100;

1) Scenario 1: Short material cutting (L< scissor circumference) L = 400 mm



Short Material Shearing Diagram (Red for Master Axis, and Blue for Slave Axis)

Since the linear speed of the master axis and the slave axis must be synchronized during materials cutting, here we take the synchronization length as 150 mm, so with the cutting point as the center, the synchronization distance on both sides is 75 mm, that is, S1=S4=75 mm. Assuming that the distances of the acceleration area and the deceleration area of the slave axis are equal, then the corresponding displacement of the master axis is S2=S3=(400-150)/2= 125 mm, and the entire displacement of the slave axis is the area included in the blue line, which is the circumference of the cutter, that is, 600 mm. S5+S6=(600-400)=200 mm. The S5+S6 area is the blue line area minus the red line area.

Therefore:

dSuperimposeIn = dSuperimposeOut = S1= S4=75mm

dSlaveBaseDistance = dMasterDistance = L=400 mm

dSlaveSuperimpose= S5+S6=200mm

dSuperimposeAcc=dSuperimposeDec=S2= S3= 125 mm.

The specific cam program is as follows:

FlexLinkReVal := HMC_ClcFlexLinkData(400, 200, 400, 75, 75, 125, 125, Pt_array, NumOfPos);

fff := HMC_TriggerScope(); (*Oscilloscope enabled*)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0); (*Send to cam, and cycle starts immediately*)

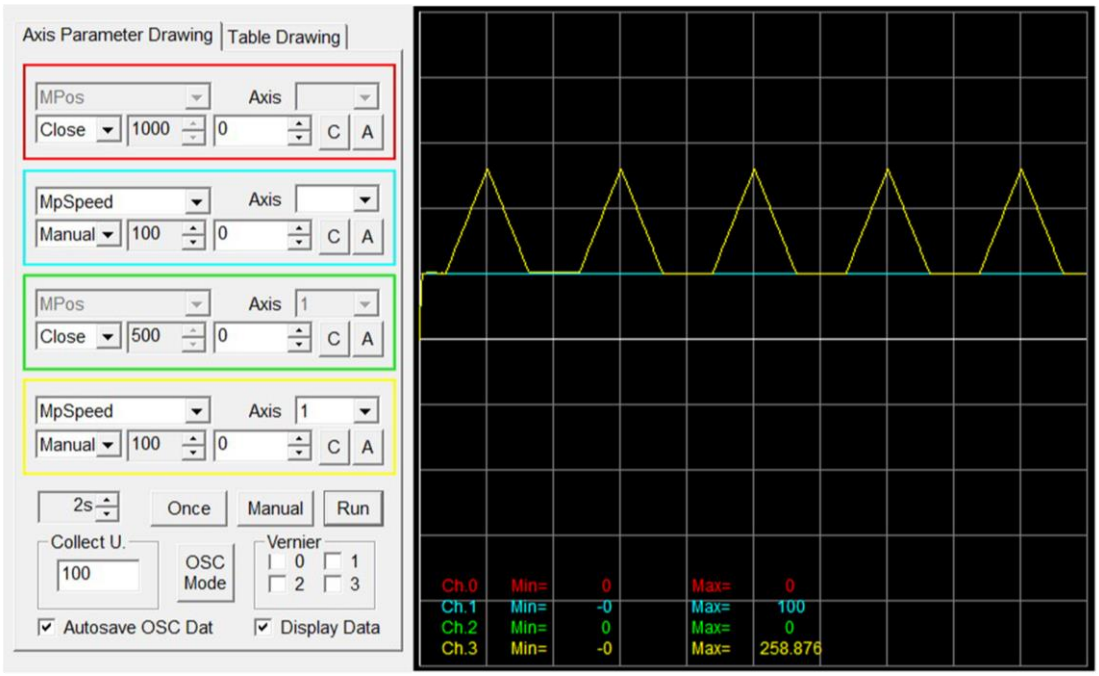
X_RESULT := HMC_Forward(0); (*Master axis starts*)

The dimension of arraympos array in AT project shall be [2,NumOfpos].

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	arraympos	%MD0		ARRAY[0..1,0..1000] ...		FALSE	Not Public	

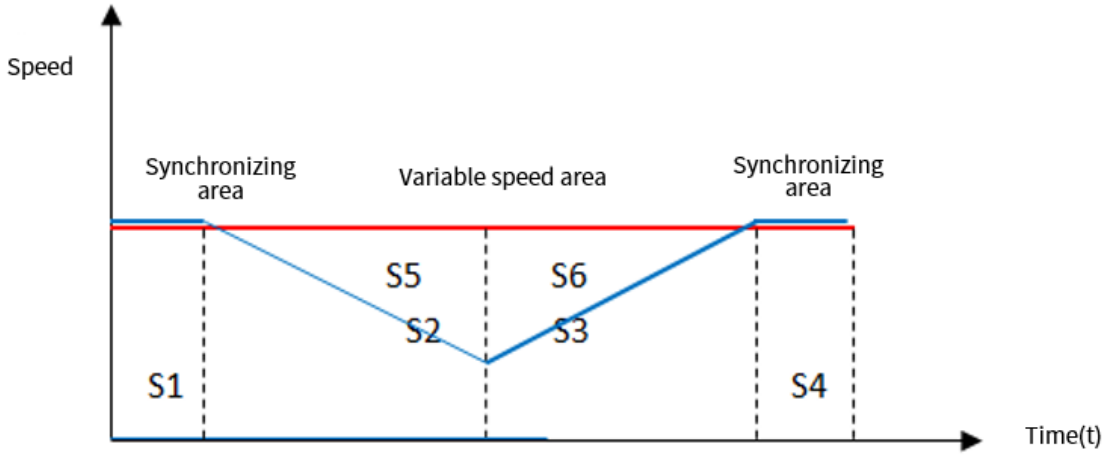
Arraympos Variable Definition

Run the task, and the oscilloscope displays:



Oscilloscope Display Results

2) Scenario 2: Medium material cutting (L>= scissor circumference) L = 800 mm



Medium Material Cutting Diagram

Area of red area (cut material length/master axis displacement): S1+S2+S3+S4=800 mm;

Area of blue area (cutter roller circumference/slave axis displacement): S1+(S2-S5)+(S3-S6)+S4=600;

S2 = S3 = (800-S1-S4)/2 = 325 mm;

S5+S6 = 800-600 = -200 mm;

The S5+S6 area is the red line area minus the blue line area.

Therefore:

dSuperimposeIn = dSuperimposeOut= S1= S4=75mm

dSlaveBaseDistance = dMasterDistance = L=800 mm

dSlaveSuperimpose = S5+S6 = -200mm

dSuperimposeAcc=dSuperimposeDec=S2= S3=325mm.

Specific cam motion program:

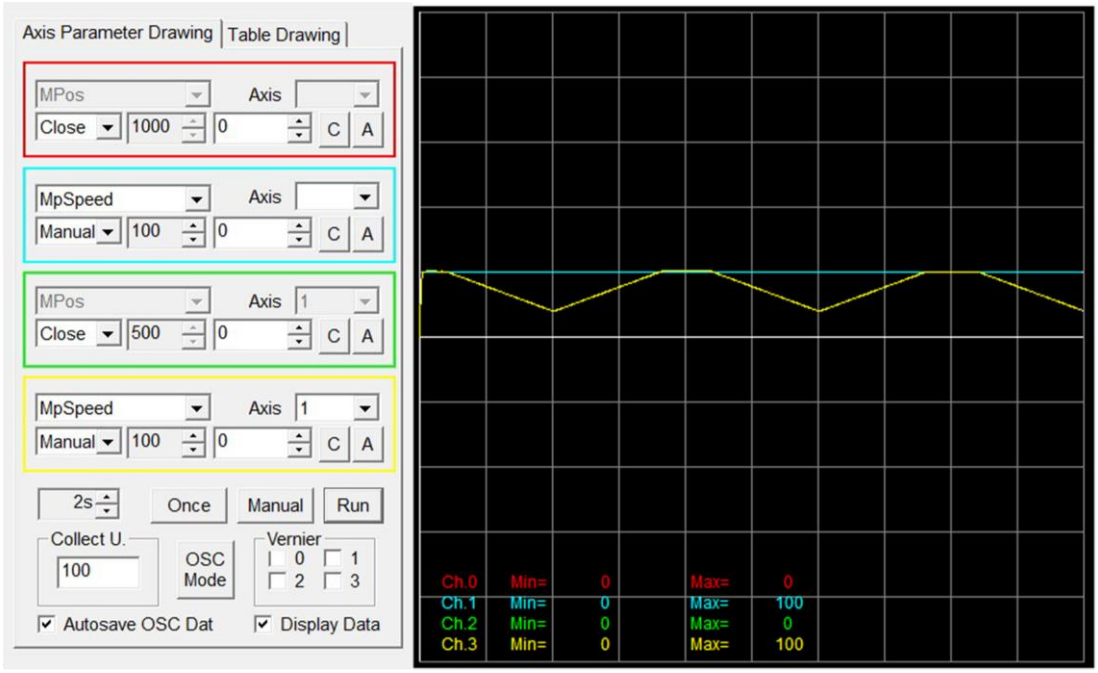
FlexLinkReVal := HMC_ClcFlexLinkData(800, -200, 800, 75, 75, 325, 325, Pt_array, NumOfPos);

fff := HMC_TriggerScope(); (*Oscilloscope enabled*)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0); (*Send to cam, and cycle starts immediately*)

X_RESULT := HMC_Forward(0); (*Master axis starts*)

Run the task, The oscilloscope displays the following:



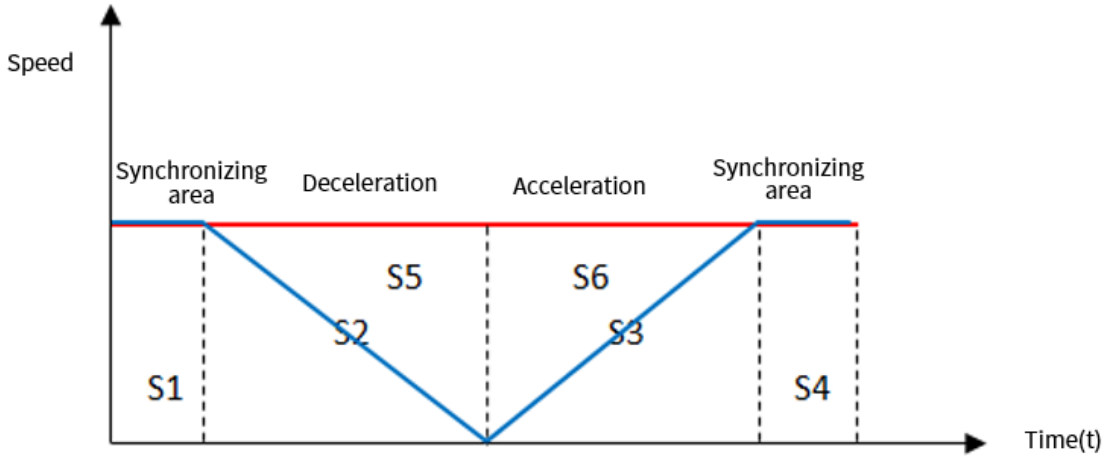
Oscilloscope Display Results

3) Scenario 3: Long material cutting

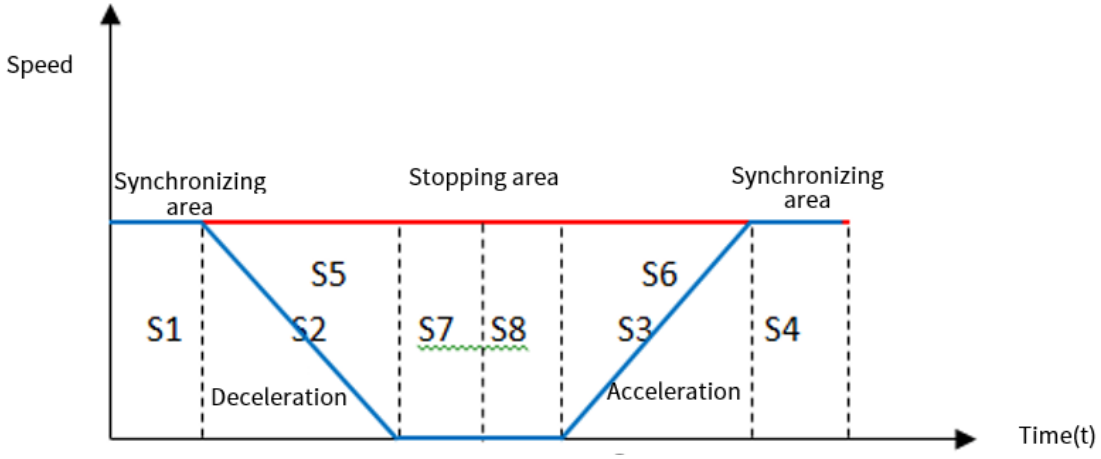
First calculate the cutting length (the area surrounded by red lines) when the cutter roller just stops (the dividing point between long material cutting and medium material cutting):

L0=S1+S2+S3+S4=(S1+S4)+2*(S5+S6)=(600- 150)*2+150= 1050 mm

When L>1050 mm, the cutter roller appears in a stopping and waiting area, which is long material cutting.



Calculation of Dividing Point Between Long Material Cutting and Medium Material Cutting



Long Material Cutting Diagram

Assuming L= 1500 mm, then the distance traveled by the master axis when the cutter roller stops for waiting:

S7+S8= L- L0= 1500- 1050=450 mm

Therefore:

dSuperimposeIn = dSuperimposeOut= S1= S4=75mm

dSlaveBaseDistance = dMasterDistance = L= 1500 mm

dSlaveSuperimpose = Cutter roller circumference - L = 600- 1500 = -900 mm

dSuperimposeAcc = dSuperimposeDec = (L - (S1+S4) - (S7+S8))/2 = (1500 – 150 – 450)/2 = 450

The specific motion program is as follows:

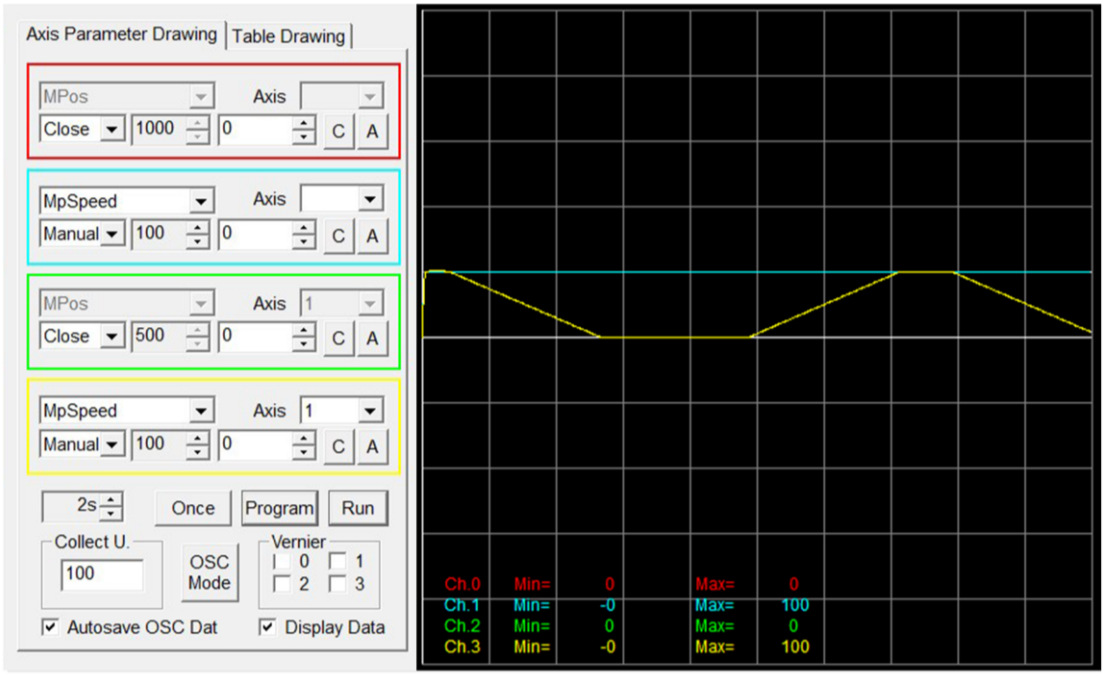
FlexLinkReVal := HMC_ClcFlexLinkData(1500, -900, 1500, 75, 75, 450, 450, Pt_array, NumOfPos);

fff := HMC_TriggerScope(); (*Oscilloscope enabled*)

X_RESULT := HMC_Forward(0); (*Master axis starts*)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0);

Run the task, and the oscilloscope displays:



Oscilloscope Display Results

5.4.6.6 HMC_CancelREPCam (Cyclic cam condition cancellation control)

5.4.6.6.1 Function

It is to use the condition cancellation function for the running cyclic cam linkage instruction. When the function is effective, the cyclic cam instruction will be canceled after completing the complete stroke of the current cam.

Different from the HMC_Cancel instruction that directly cancels the current instruction, HMC_CancelREPCam will wait for the cyclic cam instruction to complete the full stroke of the current cam before canceling the current cam instruction. The cam slave axis will follow the master axis to run to the cam master axis stroke boundary. After cancellation, the cam slave axis will stop exactly at the slave axis position corresponding to the cam master axis stroke boundary.

HMC_CancelREPCam has two modes: enabling cancellation and disabling cancellation. After HMC_CancelREPCam takes effect, it will wait for the cyclic cam instruction to complete the full stroke of the current cam before canceling the current cam instruction. Before the cam master axis runs to the stroke boundary, the HMC_CancelREPCam instruction that has taken effect can still be withdrawn just by calling HMC_CancelREPCam(ucSlaveAxis,0).

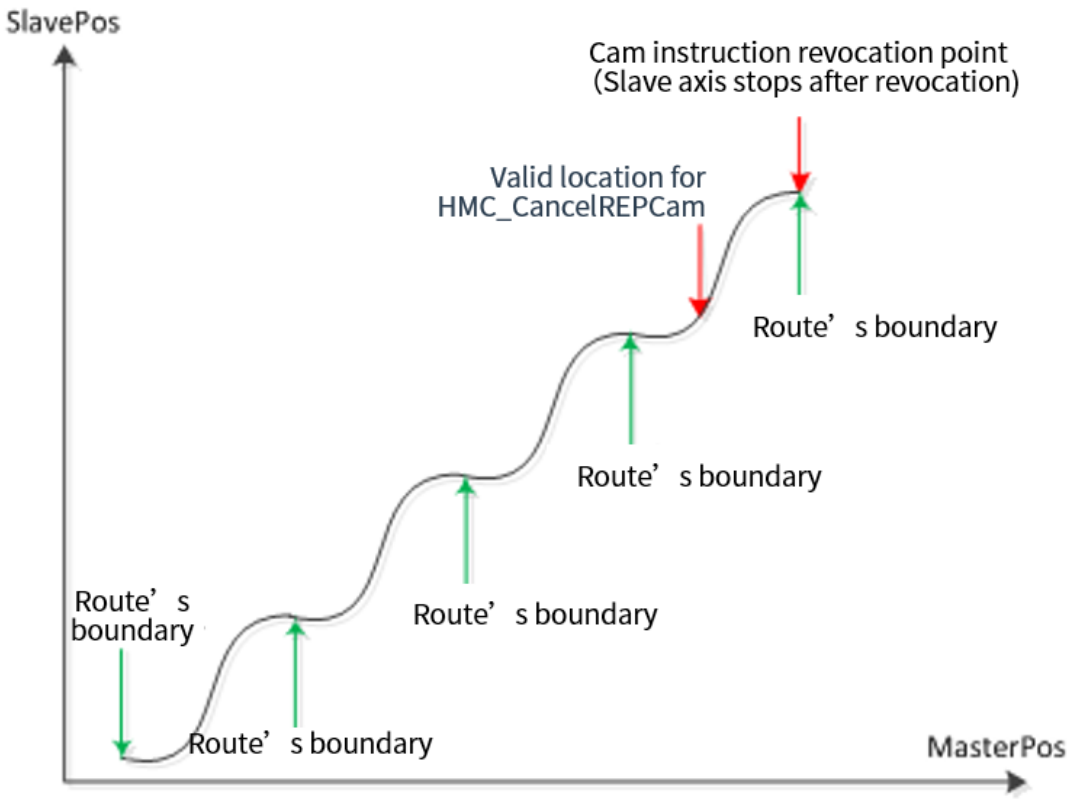


Diagram of Cyclic Cam Executing HMC_CancelREPCam Instruction

5.4.6.6.2 Parameter

Parameter	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number Here is a brief description of this parameter, and detailed signal flow is explained in "Function"
ucCancelCycMode	Input	USINT	Cancellation mode. 1: Enable cancellation 0: Disable cancellation

5.4.6.6.3 Instruction type

Non-axis motion cache instruction

Additional information:

After HMC_CancelREPCam takes effect, the cam slave axis will follow the master axis to run to the cam master axis stroke boundary, and then stop directly. If the cam slave axis speed is not 0 at this time, it will cause a large shock. Please design a suitable master/slave axis position curve to ensure that the slave axis speed is 0 or a small value at the stroke boundary.

Note: The HMC_ClcMoveLinkData and HMC_ClcFlexLinkData instructions can be used to automatically generate the master/slave axis position array to meet the requirement that the slave axis speed is 0 at the stroke

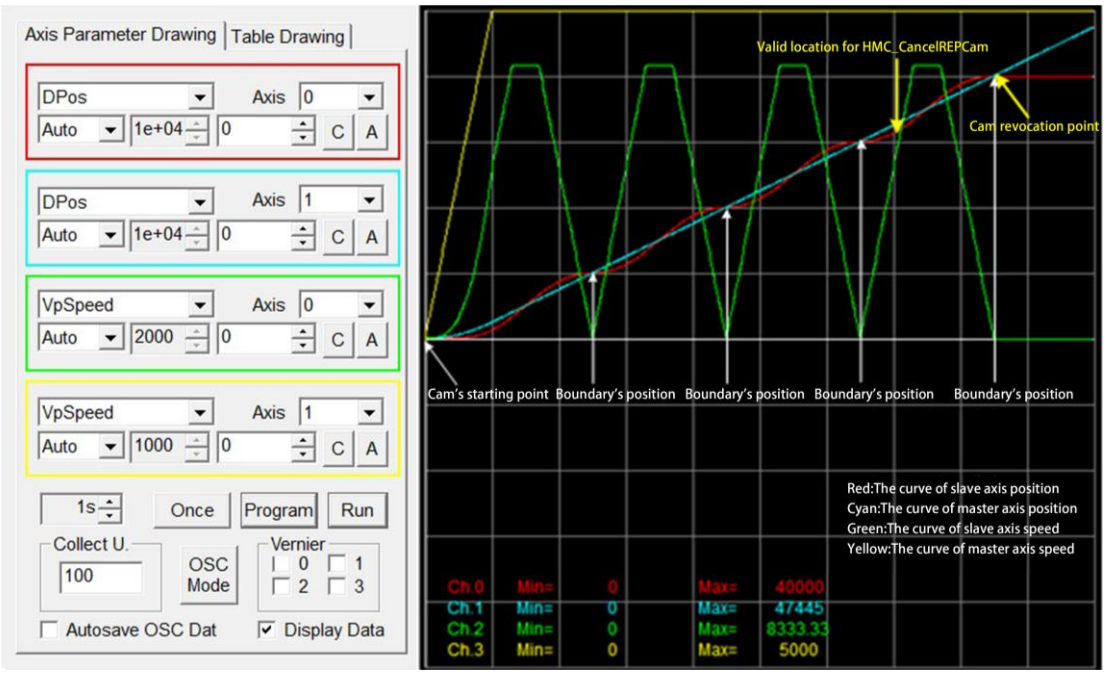
boundary.

5.4.6.6.4 Example

Use HMC_ClcMoveLinkData to generate the master/slave axis position array and save it in the user-defined array arrPos. Then use the master/slave axis position array in arrPos to send the HMC_Cam cam instruction. The mode is cyclic cam and the startup mode is immediate start. Use Hmc_Move to drive the master axis for motion, and use HMC_CancelREPCam to cancel the cyclic cam after a delay of 7 seconds.

The procedure is as follows:

```
MasterAxisID := 1; (*Master axis number*)

SlaveAxisID := 0; (*Slave axis number*)

HMC_TriggerScope(); (*The program triggers the oscilloscope to start oscilloscope sampling*)

HMC_ClcMoveLinkData(10000, 10000,4000,4000,ADR(arrPos), 100); (*Generate master/slave axis position array and save it in user-defined array arrPos*)

HMC_Cam(SlaveAxisID,MasterAxisID,ADR(arrPos), 100, 1,4,0); (*Use the master/slave axis position array in arrPos to send the cam instruction, the mode is cyclic cam, and the startup mode is immediate start*)

Hmc_Move(MasterAxisID, 200000); (*Drive master axis for motion*)

TimeDelay(7000); (*Delay 7 seconds*)

HMC_CancelREPCam(SlaveAxisID, 1); (*Cancel cyclic cam*)
```



Operation Result Diagram of Cyclic Cam Executing HMC_CancelREPCam Instruction

5.4.6.7 HMC_MoveLink (Follow-up Shearing Motion)

5.4.6.7.1 Function

This instruction integrates the cam and follow-up shearing operation functions to control the slave axis to follow the master axis to move according to a certain pattern. It can be used in follow-up shearing scenarios to quickly generate the action flow required for follow-up shearing, without the need for complex intermediate calculations and programming.

The motion has separate variable acceleration and deceleration stages. For the parameter setting method of each stage, please refer to HMC_ClcMoveLinkData (Follow-up shearing operation), which will not be repeated here.

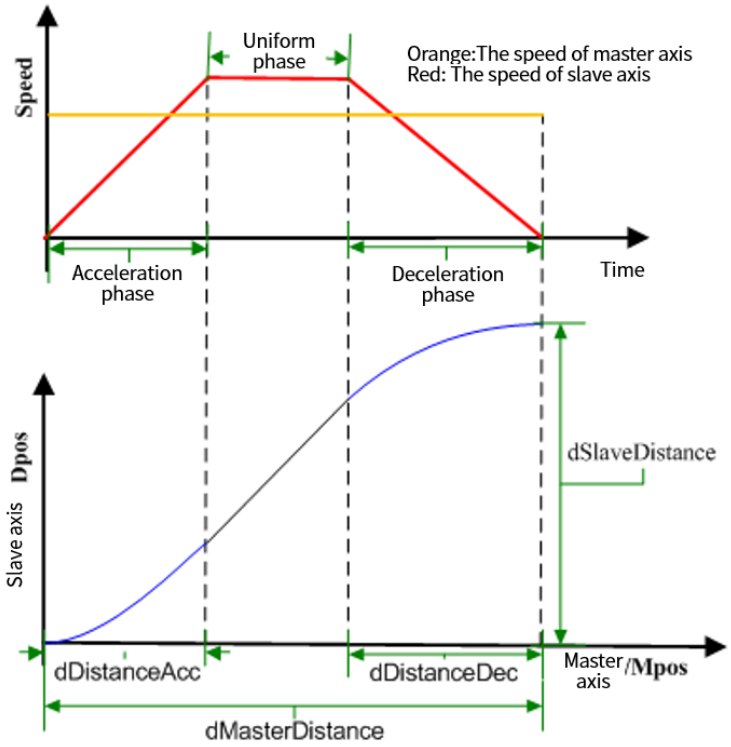
This functional block supports acceleration and deceleration according to the trapezoidal or S-shaped curve during the acceleration and deceleration stages. It does not use the calculation to generate the master/slave axis position array functional block and then controls the slave axis to follow the master axis through linear interpolation. Instead, it selects trapezoidal or S-shaped acceleration and deceleration models based on the set acceleration and deceleration stage distance, and acceleration and deceleration methods, and then calculates the slave axis position in real time based on the master axis position. This method can avoid the speed step caused by the small number of selected points in the master/slave axis position array.

Acceleration and deceleration stage planning method:

(1) Trapezoidal acceleration and deceleration dSRatio=0

When dSRatio=0, the algorithm plans the acceleration and deceleration of the slave axis according to the trapezoidal curve, and calculates the target position of the slave axis in this cycle in real time based on the trapezoidal model and the position of the master axis.

During trapezoidal acceleration and deceleration, taking the master axis motion at a constant speed as an example, the speed and displacement curves of the master and slave axes are as follows:



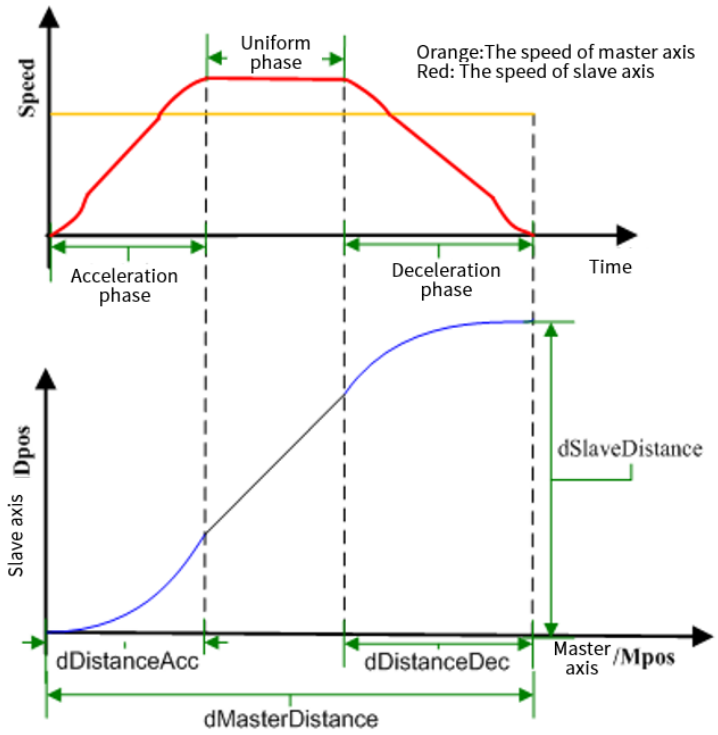
Motion Diagram

(2) S-shaped curve acceleration and deceleration dSRatio (0, 1]

When dSRatio is in the range (0, 1], the algorithm plans the acceleration and deceleration of the slave axis according to the S-shaped curve, and calculates the target position of the slave axis in this cycle in real time based on the S-shaped model and the position of the master axis.

dSRatio represents the proportion of S-shaped acceleration and deceleration in the entire acceleration/deceleration process. Taking the acceleration stage as an example, the proportion of the time used in the jerk stage to the entire acceleration phase is dSRatio/2, the proportion of the time used in the uniform acceleration stage to the entire acceleration stage is 1-dSRatio, and the proportion of the time used in the deceleration stage to the entire acceleration stage is dSRatio/2.

When dSRatio=0.5, taking the master axis motion at a constant speed as an example, the speed and displacement curves of the master and slave axes are as follows:



Motion Diagram

When dSRatio=1, there is no uniform acceleration/deceleration stages in the acceleration and deceleration stages. The jerk and deceleration ratios in the acceleration stage are 0.5 respectively, which is similar to the deceleration stage.

5.4.6.7.2 Parameter

Parameter	Statement	Data Type	Description
dSlaveDistance	Input	LREAL	Slave axis motion distance
dMasterDistance	Input	LREAL	Master axis motion distance (must be positive)
dMasterDistAcc	Input	LREAL	Positive distance moved by the master axis during the acceleration stage of the slave axis, non-negative
dMasterDistDec	Input	LREAL	Forward distance moved by the master axis during the deceleration stage of the slave axis, non-negative
dSRatio	Input	LREAL	Proportion range of S-shaped acceleration and deceleration in the slave axis acceleration/deceleration stage: [0, 1]
ucSlaveAxis	Input	USINT	Slave axis number
ucMasterAxis	Input	USINT	Master axis number
ucMode	Input	USINT	0: Single immediate start 1: Single fixed position start 2: Single event start 3: Single DI start 4: Cyclic immediate start 5: Cyclic fixed-position start

Parameter	Statement	Data Type	Description
			6: Cyclic event start 7: Cyclic DI start 13: Cyclic fixed-position start, when the master axis moves in a reverse direction 29: Single fixed-position start, when the master axis moves in a reverse direction
dTrigMes	Input	LREAL	Start information: 1: When the cam starts in in-position start mode, this value indicates the starting position of the master axis 2: When the cam is triggered in event start mode, it indicates a certain high-speed capture channel number 3: When the cam is triggered in DI (1) start mode, it indicates the DI channel (0~63)

5.4.6.7.3 Instruction type

Axis motion cache instructions.

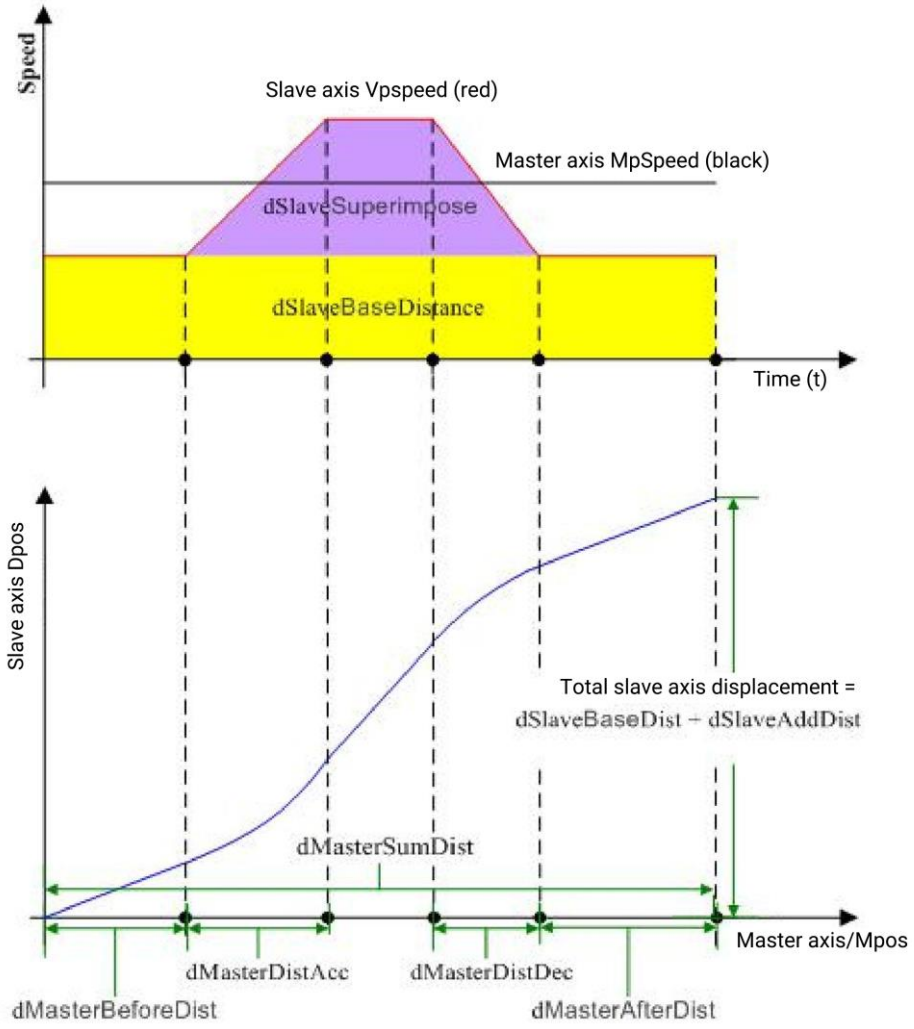
5.4.6.7.4 Example

Each parameter and algorithm method of the functional block are the same as HMC_ClcMoveLinkData. The only difference is that the acceleration mode is optional. Therefore, see HMC_ClcMoveLinkData for application examples.

5.4.6.8 HMC_MoveFlexLink (Flying Shear/Rotary Cutting Motion)

5.4.6.8.1 Function

It is used to complete the cam function with master/slave axis position and acceleration and deceleration planning (when dSRatio is set to 0, it is trapezoidal acceleration and deceleration; the proportion of S-shaped acceleration and deceleration can be adjusted through the parameter dSRatio), which can be used in follow-up shearing, flying shearing, and rotary cutting scenarios.



Master/Slave Axis Diagram

5.4.6.8.2 Parameter

Parameter	Statement	Data Type	Description
dSlaveBaseDist	Input	LREAL	Slave axis constant speed follow-up distance, positive or negative
dSlaveAddDist	Input	LREAL	Total distance of superimposed motion of the slave axis, positive or negative
dMasterSumDist	Input	LREAL	Total master axis motion distance (must be positive)
dMasterBeforeDist	Input	LREAL	Forward distance completed by the master axis when the superimposition motion (dSlaveAddDist) starts, non-negative
dMasterAfterDist	Input	LREAL	Remaining forward distance of the master axis at the end of superimposed motion (dSlaveAddDist), non-negative
dMasterDistAcc	Input	LREAL	Forward distance moved by the master axis during the acceleration stage of the slave axis

Parameter	Statement	Data Type	Description
			superimposition motion, non-negative
dMasterDistDec	Input	LREAL	Forward distance moved by the master axis during the deceleration stage of the slave axis superimposition motion, non-negative
dSRatio	Input	LREAL	When dSRatio=0, the slave axis performs trapezoidal acceleration and deceleration; when dsRatioE(0,1], the slave axis performs S-shaped acceleration and deceleration, and the proportion of S-shaped acceleration and deceleration is determined by dSRatio Value range: [0, 1]
ucSlaveAxis	Input	USINT	Slave axis number
ucMasterAxis	Input	USINT	Master axis number
ucMode	Input	USINT	0: Single immediate start 1: Single fixed position start 2: Single event start 3: Single DI start 4: Cyclic immediate start 5: Cyclic fixed-position start 6: Cyclic event start 7: Cyclic DI start 13: Cyclic fixed-position start, when the master axis moves in a reverse direction 29: Single fixed-position start, when the master axis moves in a reverse direction
dTrigMes	Input	LREAL	Start information: (1) When the cam starts in in-position start mode, this value indicates the starting position of the master axis (2) When the cam is triggered in event start mode, it indicates a certain high-speed capture channel number (3) When the cam is triggered in DI 1 start mode, it indicates the DI channel (0~63)

5.4.6.8.3 Instruction type

Axis motion cache instructions.

Additional information:

■ When dSlaveBaseDist, dMasterBeforeDist, and dMasterAfterDist are all 0, they are equivalent to the HMC_MoveLink function.

5.5 Advanced Application Instructions

5.5.1 Motion Superimposing

5.5.1.1 HMC_Superimpose (Superimposition)

5.5.1.1.1 Function

This function can realize the simple superimposition of the incremental displacement of the two master axes, and the superimposition result is output through the slave axis. After the superimposition function takes effect, assuming that the incremental displacements of the two master axes are ΔMpos_M0 and ΔMpos_M1, the incremental displacement of the slave axis Dpos is:

ΔMpos_S=ΔMpos_M0 +ΔMpos_M1

When the superimposition function is used, the two master axes execute normal motion control instructions, and the slave axis performs corresponding linkage actions following the superimposition results of the positions of the two master axes.

5.5.1.1.2 Parameter

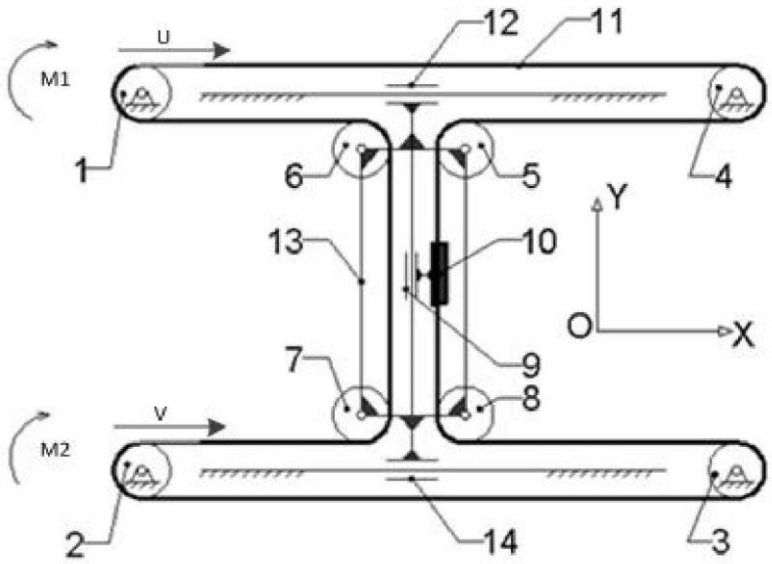
Parameter	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number
ucMasterAxis0	Input	USINT	First master axis number
ucMasterAxis1	Input	USINT	Second master axis number
HMC_Superimpose	Output	UDINT	Axis instruction ID: Success 0xFFFFFFFF: Function parameter error

5.5.1.1.3 Instruction type

Axis motion cache instructions.

5.5.1.1.4 Example

Example 1: Motion control of H-shaped synchronous toothed belt parallel mechanism



- Simplified diagram of H-shaped parallel mechanism
- 1, 2, 3, 4——Static wheel (fixed on the base, 1 and 2 selected as the driving wheels M1 and M2)
- 5, 6, 7, 8——Moving wheel
- 9, 12, 14——Sliding pair
- 10——Slider
- 11——Synchronous toothed belt
- 13——Mobile platform

Assume the user coordinate space is X-Y and the motor coordinate space is U-V. Through mechanical analysis, the forward and inverse kinematic equations of the mechanism can be obtained as follows:

Forward solution:

$$\begin{cases} \Delta x = (-\Delta u + \Delta v)/2 \\ \Delta y = -(\Delta u + \Delta v)/2 \end{cases}$$

Inverse solution:

$$\begin{cases} \Delta u = -(\Delta x + \Delta y) \\ \Delta v = \Delta x - \Delta y \end{cases}$$

Now we want to write a normal user program in the user coordinate space X-Y to drive the motor to the corresponding position in the motor coordinate space U-V. According to the kinematic inverse solution model of the mechanism, the program to realize the control of the parallel mechanism using the HMC_Superimpose and HMC_Gear (electronic gear) functions is as follows (assume that X-Y is corresponding to the master axes 0 and 1, U-V is corresponding to the slave axes 2 and 3 (motor axes), and axes 10 and 11 are used to complete the electronic gear conversion function (realizing reversal of X-axis and Y-axis).):

(**Set the master axis as a virtual axis. **)

HMC_SetAxisType(Axis0.AxisID, 0);

HMC_SetAxisType(Axis1.AxisID, 0);

(**Set axes 10 and 11 that complete the electronic gear conversion function as virtual axes. **)

HMC_SetAxisType(Axis10.AxisID, 0);

HMC_SetAxisType(Axis11.AxisID, 0);

(**Set the slave axis (motor axis) as the bus axis. **)

HMC_SetAxisType(Axis2.AxisID, 50);

HMC_SetAxisType(Axis3.AxisID, 50);

(**Set master axis motion parameters**)

Axis0.Speed := 1000; (*Set the motion speed of axis 0*)

Axis0.Accel := 2000; (*Set the acceleration of axis 0*)

Axis0.Decel: = 2000; (*Set the deceleration of axis 0*)

Axis1.Speed := 1000; (*Set the motion speed of axis 1*)

Axis1.Accel := 2000; (*Set the acceleration of axis 1*)

Axis1.Decel := 2000; (*Set the acceleration of axis 1*)

(*Send the HMC_Gear instruction, axis 10 is the slave axis, axis 0 is the master axis, and the gear ratio is -1 (X-axis reversal)*)

HMC_Gear(Axis10.AxisID,Axis0.AxisID,- 1, 1);

(*Send the HMC_Gear instruction, axis 11 is the slave axis, axis 1 is the master axis, and the gear ratio is -1 (Y-axis reversal)*)

HMC_Gear(Axis11.AxisID, Axis1.AxisID, -1, 1);

(*Send the HMC_SuperImpose instruction, axis 2 is the slave axis, and axes 10 and 11 are the master axes. *)

HMC_Superimpose(Axis2.AxisID,Axis10.AxisID, Axis11.AxisID);

(*Send the HMC_SuperImpose instruction, axis 3 is the slave axis, and axes 0 and 11 are the master axes. *)

HMC_Superimpose(Axis3.AxisID,Axis0.AxisID, Axis11.AxisID);

(*Afterwards, the required motion control instructions can be sent to the master axes 0 and 1 to drive the master axis to move, and the slave axes 2 and 3 drive the motor to the corresponding position. *)

For a simpler control implementation method of this mechanism, please refer to the motion control example for cross-shaped parallel mechanisms in HMC_SuperImposeEx.

Example 2: Synchronous tracking dispensing

The workpiece to be dispensed is transported through the conveyor. The Cartesian coordinate mechanism is responsible for completing the dispensing on the surface of the workpiece. The dispensing trajectory is an irregular geometric curve. To improve production efficiency, it is required that the dispensing head synchronously completes the dispensing of workpieces on the conveyor while the conveyor maintains normal operation.

The corresponding axis numbers of X, Y, and Z of the Cartesian coordinate mechanism are 0, 1, and 2 respectively. When installing, ensure that the X-axis of the Cartesian coordinate mechanism is parallel to the conveyor. Axis 4 is used for superimposition motion in the X direction. When the synchronous action starts, by using the HMC_Superimpose instruction, the motion of axis 4 is superimposed on the motion of the conveyor axis (axis 3), and the superimposition result is output through the X-axis (axis 0) motor.

Through motion superimposition, after the synchronous motion starts, the user only needs to program axes 4, 1, and 2 (corresponding to X, Y, and Z). When dispensing, the user only needs to control axes 4, 1, and 2 to perform interpolation motion according to the original geometric curve of the dispensing trajectory, without the need to consider the additional motion of the conveyor, while the synchronous tracking action in the X-axis direction is completed automatically by the superimposition function.

(Set the axis type. Stepper axis——Cartesian coordinate X-axis: axis 0, Y-axis: axis 1, Z-axis: axis 2, conveyor axis: axis 3. Virtual axis——Axis used for superimposition motion in the X-axis direction: axis 4*)

HMC_SetAxisType(Axis0.AxisID, 50);

HMC_SetAxisType(Axis1.AxisID, 50);

HMC_SetAxisType(Axis2.AxisID, 50);

HMC_SetAxisType(Axis3.AxisID, 50);

HMC_SetAxisType(Axis4.AxisID, 0); (*Set the axis used for superimposition motion in the X-axis direction as virtual axis*)

(*Set the speed, acceleration, and deceleration parameters of each axis. (omitted)*)

(*Detect whether the workpiece has arrived, and wait if it does not arrive (omitted)*)

(*When the workpiece arrives, the manipulator first accelerates to the same speed as the conveyor (omitted)*)

(*Send the HMC_Superimpose instruction, axis 0 (X-axis) is the slave axis, axis 3 (conveyor axis), and axis 4 (arc interpolation trajectory calculation axis) are the master axes. *)

HMC_ Superimpose(Axis0.AxisID, Axis3.AxisID, Axis4.AxisID);

(*The corresponding X, Y, and Z axis numbers when controlling the motion of the manipulator are: 4, 1, and 2 respectively. There is no need to consider the additional motion of the conveyor. The synchronous tracking action in the X-axis direction is automatically completed by the superimposition function*)

(*First move the manipulator to the starting point of workpiece dispensing (omitted)*)

(*According to the original geometric curve of the dispensing trajectory, control axes 4, 1, and 2 for interpolation motion (omitted)*)

5.5.1.2 HMC_SuperImposeEx (Advanced Superimposition)

5.5.1.2.1 Function

This function can realize the linear superimposition of the incremental displacement of the two master axes, and the superimposition result is output through the slave axis. After the superimposition function takes effect, assuming that the incremental displacements of the two master axes are ΔMpos_M0 and ΔMpos_M1, the incremental displacement of the slave axis Dpos is:

$$\triangle \mathrm{Dpos_S} = \left(\frac{\mathrm{iRatioNumerator0}}{\mathrm{iRatioDenominator0}}\right) \times \triangle \mathrm{Mpos_{M0}} + \left(\frac{\mathrm{iRatioNumerator1}}{\mathrm{iRatioDenominator1}}\right) \times \triangle \mathrm{Mpos_{M1}}$$

It supports users to dynamically modify the superimposition ratio (For details, see Example 3: Electronic gearbox realizing helical gear hobbing).

5.5.1.2.2 Parameter

Parameter	Statement	Data Type	Description
ucSlaveAxis	Input	USINT	Slave axis number
ucMasterAxis0	Input	USINT	First master axis number
ucMasterAxis1	Input	USINT	Second master axis number
iRatioNumerator0	Input	DINT	Numerator of the superimposition ratio of the first master axis
iRatioDenominator0	Input	DINT	Denominator of the superimposition ratio of the first master axis
iRatioNumerator1	Input	DINT	Numerator of the superimposition ratio of the second master axis
iRatioDenominator1	Input	DINT	Denominator of the superimposition ratio of the second master axis
HMC_SuperImposeEx	Output	UDINT	Axis instruction ID: Success 0xFFFFFFFF: Function parameter error

5.5.1.2.3 Instruction type

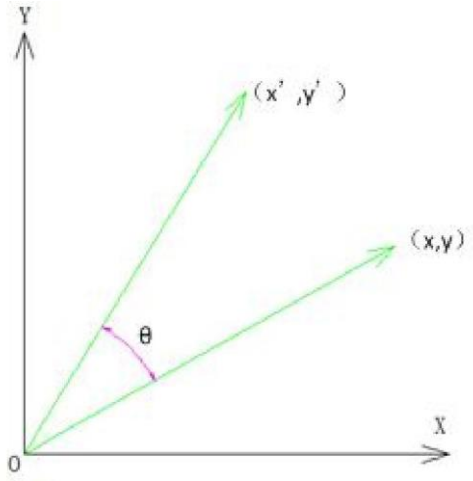
Axis motion cache instructions.

Additional information:

- From a mathematical point of view, HMC_SuperImpose and HMC_Gear are both special cases of HMC_SuperImposeEx;
- During the execution of this instruction, the superimposition ratio can be dynamically modified by calling the HMC_SetSuperImposeRatio function.

5.5.1.2.4 Example

Example 1: Coordinate transformation



Coordinate Transformation Diagram

The combined motion trajectory coordinate point (x, y) of axes 0 and 1 and the combined motion trajectory coordinate point (x', y') of axes 2 and 3 have the following relationship: (x, y) rotates by θ counterclockwise to get (x' ,y'). According to the coordinate transformation theory:

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

HMC_SuperImposeEx can be used to realize the transformation of these two sets of coordinates. The program is as follows (let θ=45)°):

(**Set master axis motion parameters**)

Axis0.Speed := 1000; (*Set the motion speed of axis 0*)

Axis0.Accel := 2000; (*Set the acceleration of axis 0*)

Axis0.Decel: = 2000; (*Set the deceleration of axis 0*)

Axis1.Speed := 1000; (*Set the motion speed of axis 1*)

Axis1.Accel := 2000; (*Set the acceleration of axis 1*)

Axis1.Decel := 2000; (*Set the acceleration of axis 1*)

(*Set the current point to zero*)

HMC_DefPos(Axis0.AxisID,0);

HMC_DefPos(Axis1.AxisID,0);

HMC_DefPos(Axis2.AxisID,0);

HMC_DefPos(Axis3.AxisID,0);

(*Send the HMC_SuperImpose instruction, axis 2 is the slave axis, and axes 0 and 1 are the master axes. Sin45 °=Cos45 ° ≈707/1000*)

HMC_SuperImposeEx(Axis2.AxisID, Axis0.AxisID, Axis1.AxisID, 707, 1000, -707, 1000);

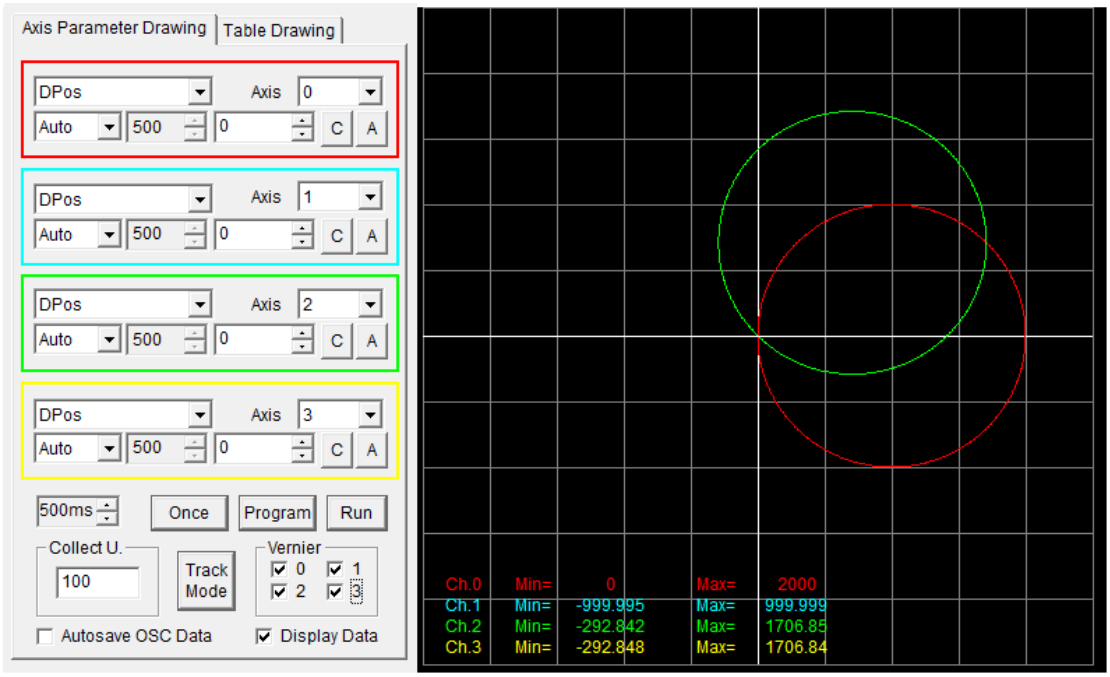
(*Send the HMC_SuperImpose instruction, axis 3 is the slave axis, and axes 0 and 1 are the master axes. *)

HMC_SuperImposeEx(Axis3.AxisID, Axis0.AxisID, Axis1.AxisID, 707, 1000, 707, 1000);

(*Then send the arc interpolation instruction to the master axes 0 and 1 to drive the master axis to move (red), and the slave axes 2 and 3 will output the converted result (green). *)

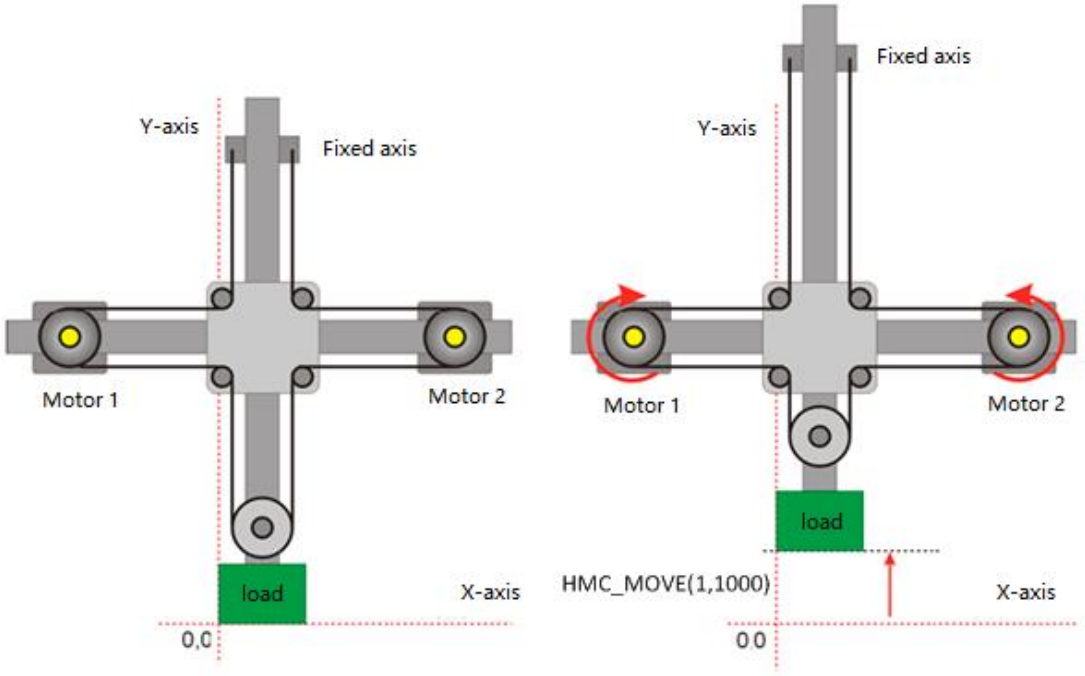
HMC_MoveCircle(10, Axis0.AxisID, Axis1.AxisID, 1, 0, 0, 1000, 0);

The running results in AT are as follows:



HMC_SuperImposeEx Realizing Coordinate Transformation

Example 2: Motion control of cross-shaped synchronous toothed belt parallel mechanism



Cross-shaped Synchronous Toothed Belt Parallel Mechanism

Assume the user coordinate space is X-Y and the motor coordinate space is U-V. Through mechanical analysis, the forward and inverse kinematic equations of the mechanism can be obtained as follows:

Forward solution:

$$\begin{cases}\Delta x = 0.5 * \Delta u + 0.5 * \Delta v \\ \Delta y = 0.5 * \Delta u - 0.5 * \Delta v\end{cases}$$

Inverse solution:

$$\begin{cases}\Delta u = \Delta x + \Delta y \\ \Delta v = \Delta x - \Delta y\end{cases}$$

Now we want to write a normal user program in the user coordinate space X-Y to drive the motor to the corresponding position in the motor coordinate space U-V. The program implemented with HMC_SuperimposeEx is as follows (assume that X-Y is corresponding to the master axes 0 and 1, and U-V is corresponding to the slave axes 2 and 3):

```
(**Set the master axis as a virtual axis. **)

HMC_SetAxisType(Axis0.AxisID, 0);

HMC_SetAxisType(Axis1.AxisID, 0);

(**Set the slave axis as bus axis. **)

HMC_SetAxisType(Axis2.AxisID, 50);

HMC_SetAxisType(Axis3.AxisID, 50);

(**Set master axis motion parameters**)

Axis0.Speed := 1000; (*Set the motion speed of axis 0*)

Axis0.Accel := 2000; (*Set the acceleration of axis 0*)

Axis0.Decel: = 2000; (*Set the deceleration of axis 0*)

Axis1.Speed := 1000; (*Set the motion speed of axis 1*)

Axis1.Accel := 2000; (*Set the acceleration of axis 1*)

Axis1.Decel := 2000; (*Set the acceleration of axis 1*)

(*Send the HMC_SuperImpose instruction, axis 2 is the slave axis, and axes 0 and 1 are the master axes. *)

HMC_SuperimposeEx(Axis2.AxisID, Axis0.AxisID, Axis1.AxisID, 1, 1, 1, 1);

(*Send the HMC_SuperImpose instruction, axis 3 is the slave axis, and axes 0 and 1 are the master axes. *)

HMC_SuperimposeEx(Axis3.AxisID, Axis0.AxisID, Axis1.AxisID, 1, 1, - 1, 1);

(*Afterwards, the required motion control instructions can be sent to the master axes 0 and 1 to drive the master axis to move, and the slave axes 2 and 3 drive the motor to the corresponding position. *)
```

Example 3: Electronic gearbox realizing helical gear hobbing

CNC gear hobbing machines and gear shaping machines use electronic gearboxes to replace the original mechanical inline transmission chains. Compared with conventional mechanical inline transmission chains, CNC gear processing machines adopting electronic gearboxes have shorter transmission chains, which improves transmission stiffness and transmission accuracy. the moving axes in each direction can move alone or in multi-axis linkage. The processing process is controlled by the program, which features flexible motion, accurate positioning, high precision, and high efficiency, so it can process parts that cannot be processed by ordinary gear processing machines.

The HMC_SuperImposeEx instruction can realize the electronic gearbox function required by CNC gear hobbing machines and gear shaping machines. The following analysis takes the gear hobbing machine as an example.

First, based on the motion characteristics of each axis of the gear hobbing machine, a mathematical model for the control of the shaftless transmission electronic gearbox is established. The motion axes of the hobbing machine are: the master motion axis B-axis (hobbing master axis), workpiece axis (C-axis), X-axis (radial feed axis), Y-axis (shifting tool axis), and Z-axis (axial feed axis).

Assume that the number of heads of the hob (B-axis) is zB with the rotation speed as nB, and the number of teeth of the workpiece (C-axis) is zC. To process helical gears using the differential method or process gears using the diagonal hobbing method, in addition, there must not only be an accurate speed ratio relationship between the master axis of the machine tool workpiece and the master axis of the machine tool, but the workpiece master axis must also complete accurate follow-up to the Z-axis or Y-axis, so to keep strict generating motion between the hob and the workpiece. Select the workpiece axis as the driven axis of the electronic gearbox, and its motion speed can be described as:

$$n_C = K_B \frac{z_B}{z_C} n_B + K_Z \frac{\sin\beta}{\pi m_n z_C} v_Z + K_Y \frac{\cos\lambda}{\pi m_n z_C} v_Y$$

Where vY and vZ are the moving speeds of the Y-axis and Z-axis respectively in mm/min; β is the helix angle of the helical gear; λ is the installation angle of the tool; mn is the normal module of the gear, and; KB, KZ, and KY are the coefficients.

It can be seen from the above formula that during gear hobbing, the C-axis is not only related to the speed of the B-axis, but the motion of the Y-axis and Z-axis will also produce additional motion of the C-axis. Assuming that both the Y-axis and the Z-axis adopt the servo motor + ball screw transmission, the above formula can be simplified to (the simplest fractional form):

$$n_C = \frac{i_{Bn}}{i_{Bd}} n_B + \frac{i_{Zn}}{i_{Zd}} n_Z + \frac{i_{Yn}}{i_{Yd}} n_Y$$

nB, nC, nY, and nZ are the driving motor speeds of the B-axis, C-axis, Y-axis, and Z-axis respectively.

Assume that the corresponding axis numbers of the drive motors of the B-axis, C-axis, X-axis, Y-axis, and Z-axis are 0, 1, 2, 3, and 4 respectively, and the example program using the HMC_SuperImposeEx instruction to

implement the motion relationship is as follows (ST language):

(**Set the B-axis, C-axis, X-axis, Y-axis, and Z-axis as closed-loop bus axes. **)

HMC_SetAxisType(Axis0.AxisID, 50);

HMC_SetAxisType(Axis1.AxisID, 50);

HMC_SetAxisType(Axis2.AxisID, 50);

HMC_SetAxisType(Axis3.AxisID, 50);

HMC_SetAxisType(Axis4.AxisID, 50);

(**Set axis 5 used for intermediate operations as a virtual axis. **)

HMC_SetAxisType(Axis5.AxisID, 0);

(**Set motion parameters for the B-axis, C-axis, X-axis, Y-axis, and Z-axis (omitted)**)

(**Adjust the device to the starting position of processing (omitted)**)

(*Send the HMC_SuperImpose instruction, axis 5 is the slave axis, and the B-axis (0) and Z-axis (4) are the master axes. *)

HMC_SuperimposeEx(Axis5.AxisID, Axis0.AxisID, Axis4.AxisID, iBn, iBd, iZn, iZd);

(*Send the HMC_SuperImpose instruction, the C-axis (1) is the slave axis, and the Y-axis (3) and axis 5 are the master axes. *)

HMC_SuperimposeEx(Axis1.AxisID, Axis3AxisID, Axis5.AxisID, iYn, iYd, 1, 1);

(*Adjust the appropriate radial feed amount through the X-axis, and then drive the B-axis/Y-axis/Z-axis to move. The slave axis C will move according to the above relationship to achieve generating processing of helical gear*)

5.5.1.3 HMC_SetSuperimposeRatio (Setting Superimposition Ratio)

5.5.1.3.1 Function

It is used to set the superimposition ratio. For details on superimposition, see HMC_Superimpose (Superimposition) and HMC_SuperImposeEx (Advanced superimposition). After the HMC_Superimpose or HMC_SuperimposeEx instruction is started, this instruction can be called in real time to dynamically change the superimposition ratio during operation, thereby achieving the effect of dynamically changing the superimposition ratio.

This instruction can only be set when the HMC_Superimpose or HMC_SuperimposeEx instruction is currently running.

5.5.1.3.2 Parameter

Parameter	Statement	Data Type	Description

ucSlaveAxis	Input	USINT	Slave axis number
iRatioNumerator0	Input	DINT	Numerator of the superimposition ratio of the first master axis
iRatioDenominator0	Input	DINT	Denominator of the superimposition ratio of the first master axis
iRatioNumerator1	Input	DINT	Numerator of the superimposition ratio of the second master axis
iRatioDenominator1	Input	DINT	Denominator of the superimposition ratio of the second master axis
HMC_SetSuperimposeRatio	Output	UDINT	0: Success Other values: Error code

5.5.1.3.3 Instruction type

It is a non-axis motion cache instruction.

5.5.1.3.4 Example

The example program is as follows:

Axis 0.Speed:= 1000; (*Set motion speed*)

Axis0.Accel:= 2000; (*Set the acceleration*)

Axis0.Decel: = 2000; (*Set the deceleration*)

Axis1.Speed := 1000; (*Set motion speed*)

Axis1.Accel := 2000; (*Set the acceleration*)

Axis1.Decel: = 2000; (*Set the deceleration*)

(*Send the HMC_SuperImpose instruction, axis 3 is the slave axis, and axes 0 and 1 are the master axes. *)

HMC_SuperimposeEx(Axis3.AxisID, Axis0.AxisID, Axis1.AxisID, 1, 1, 1, 1);

Cmd_move0 := HMC_Move(Axis 0.AxisID, 1000); (*Send the HMC_Move instruction to axis 0, with a motion increment distance of 1000*)

Cmd_move1 := HMC_Move(Axis1.AxisID, 2000); (*Send the HMC_Move instruction to axis 0, with a motion increment distance of 2000*)

HMC_WaitCmd Finish(Cmd_move0); (*Wait for the instruction to complete*)

HMC_WaitCmd Finish(Cmd_move1); (*Wait for the instruction to complete*)

HMC_ SetSuperimposeRatio(Axis3.AxisID, 1, 1, - 1, 1); (*Change the superimposition ratio*)

HMC_Move(Axis0.AxisID, 1000); (Send the HMC_Move instruction to axis 0, with a motion increment distance of 1000*)

HMC_Move(Axis1.AxisID, -1000); (Send the HMC_Move instruction to axis 0, with a motion increment

distance of -1000*)

5.5.1.4 HMC_GetSuperimposeMaster1 (Getting superimposed master axis 1)

5.5.1.4.1 Function

It is used to get the first superimposed master axis.

5.5.1.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Slave axis number
HMC_GetSuperimposeMaster1	Output	INT	First superimposed master axis: Success -1: No superimposition motion performed currently or function parameters error

5.5.1.4.3 Instruction type

It is a non-axis motion cache instruction.

5.5.1.5 HMC_GetSuperimposeMaster2 (Getting superimposed master axis 2)

5.5.1.5.1 Function

Get the second superimposed master axis.

5.5.1.5.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Slave axis number
HMC_GetSuperimposeMaster2	Output	INT	Second superimposed master axis: Success -1: No superimposition motion performed currently or function parameters error

5.5.1.5.3 Instruction type

It is a non-axis motion cache instruction.

5.5.1.6 HMC_GetSuperimposeRatio1 (Getting superimposed master axis 1 ratio)

5.5.1.6.1 Function

Get the first superimposed master axis ratio.

5.5.1.6.2 Parameter

Parameter	Statement	Data Type	Description

ucAxisID	Input	USINT	Slave axis number
HMC_GetSuperimposeRatio1	Output	LREAL	First superimposed master axis ratio: Success 0xFFFFFFFF: No superimposition motion performed currently or function parameters error

5.5.1.6.3 Instruction type

It is a non-axis motion cache instruction.

5.5.1.7 HMC_GetSuperimposeRatio2 (Getting superimposed master axis 2 ratio)

5.5.1.7.1 Function

Get the second superimposed master axis ratio.

5.5.1.7.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Slave axis number
HMC_GetSuperimposeRatio2	Output	LRAEL	Second superimposed master axis ratio: Success 0xFFFFFFFF: No superimposition motion performed currently or function parameters error

5.5.1.7.3 Instruction type

It is a non-axis motion cache instruction.

5.5.1.8 HMC_SetAddAxis (Setting summation motion axes)

5.5.1.8.1 Function

This instruction is used to complete the summation of two-axis motion. After the instruction is called successfully, in the subsequent motion process, there is base axis displacement = base axis own instruction displacement + superimposed axis displacement. The displacements in the above formula are all Dpos increments.

When ucAddAxisID=255, it means the summation motion of the ucBaseAxisID axis and other axes is canceled. You can query the superimposed axis number through the HMC_GetAddAxis(ucBaseAxisID) instruction.

5.5.1.8.2 Parameter

Parameter	Statement	Data Type	Description
ucBaseAxisID	Input	USINT	Base axis number, 0~63
ucAddAxisID	Input	USINT	Superimposed axis number (valid associated axis number:

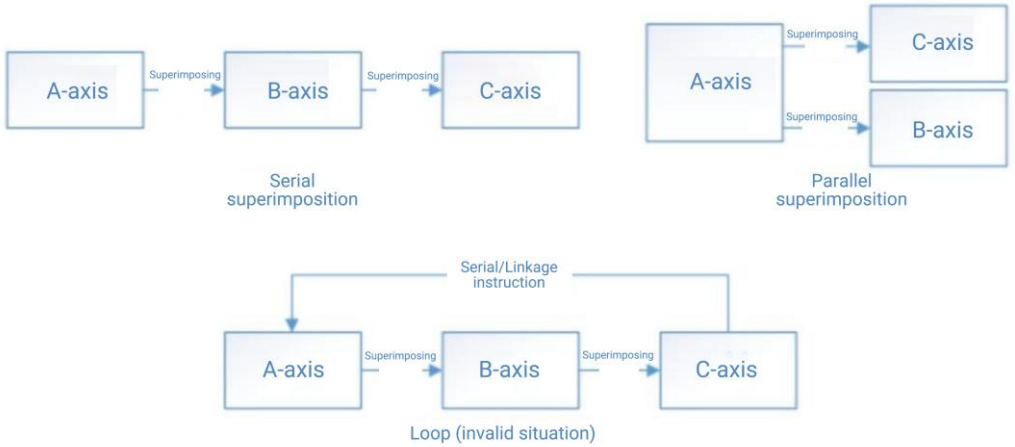
			0~63; 255 means canceling the association.)
HMC_SetAddAxis	Output	UDINT	Success: 0 Error: error code

5.5.1.8.3 Instruction type

It is a non-axis motion cache instruction.

5.5.1.8.4 Additional information

(1) Loops shall not be formed when the superimposition instruction is used, otherwise, the axis speed may diverge out of control. The following figure shows the normal series/parallel open circuit situation and the abnormal loop situation in sequence. Due to the diversity of abnormal loops, detection is difficult, so the user needs to ensure normal usage when using it.



(2) After the instruction is called successfully, Dpos/Mpos reflects the actual position of the base axis, and MpSpeed reflects the actual superimposed speed, but VpSpeed is the instruction speed of the base axis, not the superimposed value.

5.5.1.9 HMC_GetAddAxis (Getting summation motion superimposition axis)

5.5.1.9.1 Function

Used to get the summation motion superimposition axis number, with 255 indicating that no superimposed axis is associated.

5.5.1.9.2 Parameter

Parameter	Statement	Data Type	Description
ucBaseAxisID	Input/Output	USINT	Base axis number, 0~63
HMC_GetAddAxis	Output	DINT	Associated axis number: Success 255: no superimposition axis is associated

5.5.1.9.3 Instruction type

It is a non-axis motion cache instruction.

5.5.2 Low Speed In-position Output

The low-speed in-position output is similar to the high-speed in-position output, but the accuracy of position comparison is lower than that of the high-speed in-position output. The position comparison accuracy is related to the servo cycle and axis speed, so it is suitable for application scenarios that do not require high control accuracy.

The low-speed in-position output supports richer sub-functions:

- Input

It supports all axis types, including bus, virtual axis, etc.;

For signal source comparison, it supports the target position Dpos and the actual feedback position Mpos.

- Output

It can output through any type of DOs, including high-speed, low-speed, and virtual DOs;

For signal output mode, it supports level flipping and pulse output of specified width.

- Position comparison

The comparison position sequence can be set arbitrarily, but it must conform to the combined axis position transformation requirements, otherwise, it will be in a status that is waiting for a certain point to be judged.

It supports one-dimensional position comparison and two-dimensional position comparison.

5.5.2.1 HMC_NormalSwitch1Dconfig (Setting one-dimensional low-speed output)

5.5.2.1.1 Function

Configure the one-dimensional low-speed output. It compares with the axis position according to the set position comparison sequence. When the axis position passes the target comparison position in this cycle, the output is activated.

It supports target position sequences that are not strictly monotonic. When compared according to the target position sequence array from front to back, if the current position to be compared cannot be triggered, it will always be in the status of judging the target point and subsequent positions will not be compared.

5.5.2.1.2 Parameter

Parameter	Statement	Data Type	Description
ucChIID	Input	USINT	Normal in-position output channel number, with channel numbers 0~7 shared with HMC_NormalSwitch2DConfig
ucOutputID	Input	UDINT	Output position selection, 0~65535

ucTrigAxis	Input	USINT	Sampling axis, which is compared with the target position value by the switch to determine the output level
ucTrigPosType	Input	USINT	Position comparison source mode: 0: Compare the Mpos of the sampling axis 1: Compare the Dpos of the sampling axis
ucOutputType	Input	USINT	Output mode: 0: Level flipping 1: Pulse with settable pulse width
bSwitchValue	Input	USINT	When ucOutputType=0, it means the output level value of the first position is 0 or 1 When ucOutputType=1, it means that the output pulse level value is 0 or 1
uiPluseWidth	Input	UDINT	When ucOutputType= 1, this parameter is valid and represents the pulse width in ms (the set value shall be ≥ 1 servo cycle, and the resolution shall be plus or minus 1 servo cycle.)
pPos	Input	POINTER TO LREAL	Target position storage address, and the array stores the target position set value
uiPosNum	Input	UDINT	Number of target Mpos sequences
HMC_NormalSwitch1DConfig	Output	UDINT	0: Configured successfully others: Error code

5.5.2.1.3 Instruction type

It is a non-axis motion cache instruction.

5.5.2.2 HMC_NormalSwitch2DConfig (Setting two-dimensional low-speed output)

5.5.2.2.1 Function

It is used to configure a two-dimensional low-speed output and perform comparisons according to the set position comparison sequence. When the two-dimensional coordinates corresponding to the two-axis position enter the target position equal area and select the better position, the output is activated. The specific position determination method is as follows:

If the distance between the current point and the target point is smaller than or equal to the position comparison threshold dThreshold, it is considered that entering the target position equal area is a necessary condition for triggering the output;

After entering the equal area, try to select a point at the closest distance to the target point to output; by

continuously comparing the distance to the target point, confirm whether it is approaching or far away, so as to find the optimal opportunity;

When the comparison signal source is the actual feedback position of the axis, anti-shake processing needs to be performed by using the anti-shake threshold parameter dDisturbValue;

As long as the equal area is entered, the output will be triggered.

Based on the above multiple processes, the output timing is not necessarily at the closest point to the target point, but near it. This depends on many factors such as the judgment threshold set by the user and the speed of the axis.

5.5.2.2.2 Parameter

Parameter	Statement	Data Type	Description
ucChIID	Input	USINT	Normal in-position output channel number, with channel numbers 0~7 shared with HMC_NormalSwitch2DConfig
ucOutputID	Input	UDINT	Output position selection, 0~65535
ucTrigAxisX	Input	USINT	Sampling axis X, which is compared with the X target position value by the switch to determine the output level
ucTrigAxisY	Input	USINT	Sampling axis Y, which is compared with the Y target position value by the switch to determine the output level
ucTrigPosType	Input	USINT	Position comparison source mode: 0: Compare the Mpos of the sampling axis 1: Compare the Dpos of the sampling axis
ucOutputType	Input	USINT	Output mode: 0: Level flipping 1: Pulse with settable pulse width
bSwitchValue	Input	USINT	When ucOutputType=0, it means the output level value of the first position is 0 or 1. When ucOutputType= 1, it means the output pulse level value is 0 or 1.
uiPluseWidth	Input	UDINT	When ucOutputType= 1, this parameter is valid and represents the pulse width in ms (the set value shall be ≥ 1 servo cycle, and the resolution shall be plus or minus 1 servo cycle.)
pPosX	Input	POINTER TO LREAL	X target position storage address, and the array stores the target position set value. When comparing two-dimensional positions, this array describes the X-axis position value
pPosY	Input	POINTER TO LREAL	Y target position storage address. This parameter is valid when it is a two-dimensional position comparison. This array describes the Y-axis position value, and

			together with pMposX, describes the two-dimensional comparison target position
uiPosNum	Input	UDINT	Number of target Mpos sequences
dThreshold	Input	LREAL	Position comparison deviation threshold, which is used to determine whether the target position equal area (≥0) is reached
dDisturbValue	Input	LREAL	Anti-shake threshold. It is valid only when the position comparison source is Mpos and the feedback position comes from physical input. When the feedback position is compared with the target position, it is used for anti-shake processing. (≥0)
HMC_NormalSwitch1DConfig	Output	UDINT	0: Configured successfully others: Error code

5.5.2.2.3 Instruction type

It is a non-axis motion cache instruction.

5.5.2.3 HMC_GetNormalSwitchNum (Getting the Number of Outputs)

5.5.2.3.1 Function

Get the number of segments that the low speed in-position output with the specified ID has been executed.

5.5.2.3.2 Parameter

Parameter	Statement	Data Type	Description
ucChIID	Input	USINT	Low-speed output channel number, 0~7
HMC_GetNormalSwitchNum	Output	UDINT	Number of segments that have completed execution

5.5.2.3.3 Instruction type

It is a non-axis motion cache instruction.

5.5.2.4 HMC_GetNormalSwitchState (Getting low-speed output status)

5.5.2.4.1 Function

Get the current status of the low speed output with the specified ID.

5.5.2.4.2 Parameter

Parameter	Statement	Data Type	Description
ucChIID	Input	USINT	Low-speed output channel number, 0~7
HMC_GetNormalSwitchState	Output	UDINT	Return value 0: The current channel

			is working Return value 3: The current channel is idle, indicating that it is completed or not configured

5.5.2.4.3 Instruction type

It is a non-axis motion cache instruction.

5.5.2.5 HMC_CloseNormalSwitch (Closing Low-Speed Output)

5.5.2.5.1 Function

Close the low speed in-position output of this ID.

5.5.2.5.2 Parameter

Parameter	Statement	Data Type	Description
ucChIID	Input	USINT	Low-speed output channel number, 0~7
HMC_CloseNormalSwitch	Output	UDINT	Return value 0: Closed successfully Other values: Error code

5.5.2.5.3 Instruction type

It is a non-axis motion cache instruction.

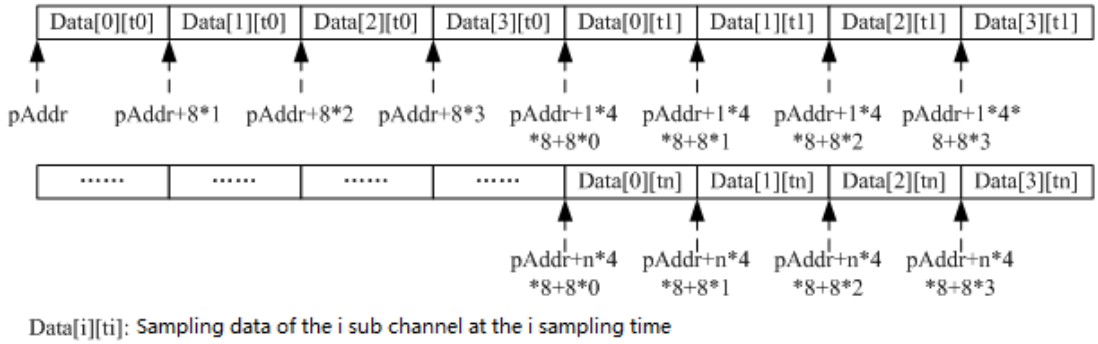
5.5.3 Oscilloscope Function Instructions

5.5.3.1 HMC_SetSampleVar (Setting Sampling Variables)

5.5.3.1.1 Function

The data sampling function can sample data from the system's axis parameters, I/O data, and intermediate/system variables according to the configured data sampling interval, and store the real-time data records obtained by sampling in the user-specified area, which must be located in the M area.

Regardless of whether the data type of the data to be sampled is integer or floating point, the sample data is ultimately stored in the user-specified area according to the double-precision floating-point LREAL type. The four sub-channel data of one channel are stored in the user-specified area in chronological order as follows:



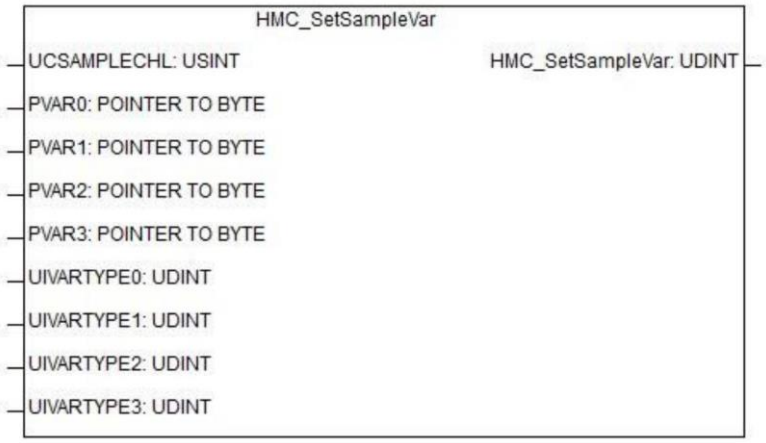
Data[i][ti]: Sample data of the ith sub-channel at the ith sampling time

Sample Storage Diagram

The recorded data can be parsed and viewed according to the above storage format. When only some sub-channels are used (for example, only channels 0 and 1 are used), the value of the next sampling time of sub-channel 0 will be stored immediately after the data of sub-channel 1, without any idle time in between.

The functional block is to configure the source of data to be sampled, and supports the collection of up to 4 variables.

The sampling function must be used in conjunction with other sampling-related functional blocks in the system. First call SetSampleVar to configure the data to be sampled, then call SetSamplePara to set the storage area, sampling interval, and total number of sampling points, and finally use HMC_TriggerSample to trigger sampling execution. During the sampling execution process, HMC_GetSampleNum can be used to check the current number of completed samples. After the sampling execution is completed, HMC_GetSampleData can be used to check the sample value of a sub-channel with the specified sampling time number. For other sampling-related functional blocks, please see the corresponding chapters.



HMC_SetSampleVar Functional Block

5.5.3.1.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel number

pVar0	Input	POINTER TO BYTE	Address of the 0th sub-channel data to be sampled
pVar1	Input	POINTER TO BYTE	Address of the first sub-channel data to be sampled
pVar2	Input	POINTER TO BYTE	Address of the second sub-channel data to be sampled
pVar3	Input	POINTER TO BYTE	Address of the third sub-channel data to be sampled
uiVarType0	Input	UDINT	Data type of the 0th sub-channel to be sampled, including USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, F32=7, LREAL=8
uiVarType1	Input	UDINT	Data type of the first sub-channel to be sampled, including USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, F32=7, LREAL=8
uiVarType2	Input	UDINT	Data type of the second sub-channel to be sampled, including USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, F32=7, LREAL=8
uiVarType3	Input	UDINT	Data type of the third sub-channel to be sampled, including USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, F32=7, LREAL=8
HMC_SetSampleVar	Output	UDINT	0: Succeeded Others: Error code

5.5.3.1.3 Instruction type

It is a non-axis motion cache instruction.

5.5.3.1.4 Example

Sample Mpos (LREAL) and VpSpeed (LREAL) of axis 0. The sampling interval is 1 times the servo cycle and 2 times the servo cycle respectively. The servo cycle is 1 ms, and the number of samples is 3000.

```
Axis0.Speed:=2000;

HMC_Forward(0);

HMC_SetSampleVar (0, ADR(axis0.MPos), ADR (axis0.VpSpeed), ADR(Temp), ADR(Temp), 8, 8, 7, 7); (*Use sub-channels 0 and 1 of channel 0 for sampling, use channels 2 and 3 to sample the variable Temp with a type of F32*)

HMC_SetSamplePara (0, 1,3000, ADR (RecordAry0)); (*Set the sampling interval of channel 0 to 1 servo cycle, the number of sampling points is 3000/sub-channel, and the sample data is stored in the array RecordAry0*)

HMC_SetSampleVar (1, ADR (axis0.MPos), ADR(axisO0. VpSpeed), ADR(Temp), ADR(Temp), 8, 8, 7, 7);
```

HMC_SetSamplePara (1, 10, 3000, ADR (RecordAry1)); (*Set the sampling interval of channel 1 to 10 servo cycles, the number of sampling points is 3000/sub-channel, and the sample data is stored in the array ARecordAry1*)

HMC_TriggerSample(0); (*Trigger sampling channel 0*)

HMC_TriggerSample(1); (*Trigger sampling channel 1*)

The sampling results stored in the arrays RecordAry0 and RecordAry1 are as follows:

Variable Name	Direct Address	Online Value
RecordAry0[0,0]	%MD0	5238
RecordAry0[0,1]	%MD8	2000
RecordAry0[1,0]	%MD16	5240
RecordAry0[1,1]	%MD24	2000
RecordAry0[2,0]	%MD32	5242
RecordAry0[2,1]	%MD40	2000
RecordAry0[3,0]	%MD48	5244
RecordAry0[3,1]	%MD56	2000
RecordAry0[4,0]	%MD64	5246
RecordAry0[4,1]	%MD72	2000
RecordAry0[5,0]	%MD80	5248
RecordAry0[5,1]	%MD88	2000
RecordAry0[6,0]	%MD96	5250
RecordAry0[6,1]	%MD104	2000
RecordAry0[7,0]	%MD112	5252
RecordAry0[7,1)]	%MD120	2000

Variable Name	Direct Address	Online Value
RecordAry1[0,0]	%MD5000	8320
RecordAry1[0,1]	%MD5008	2000
RecordAry1[1,0]	%MD5016	8340
RecordAry1[1,1]	%MD5024	2000
RecordAry1[2,0]	%MD5032	8360
RecordAry1[2,1]	%MD5040	2000
RecordAry1[3,0]	%MD5048	8380
RecordAry1[3,1]	%MD5056	2000
RecordAry1[4,0]	%MD5064	8400
RecordAry1[4,1]	%MD5072	2000
RecordAry1[5,0]	%MD5080	8420
RecordAry1[5,1]	%MD5088	2000
RecordAry1[6,0]	%MD5096	8440
RecordAry1[6,1]	%MD5104	2000
RecordAry1[7,0]	%MD5112	8460
RecordAry1[7,1)]	%MD5120	2000

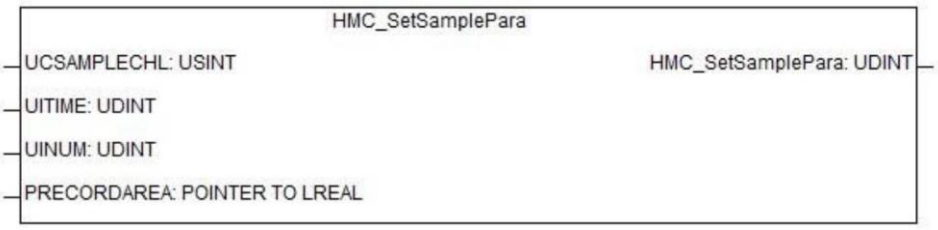
Storage Diagram

It can be seen that the difference between the Mpos sampling intervals stored in RecordAry0 is 2, which conforms to the requirements that the speed is 2000, the servo cycle is 1 ms, and the sampling interval is 1 servo cycle; the difference between the Mpos sampling intervals stored in RecordAry1 is 20, which conforms to the requirements that the speed is 2000, the servo cycle is 1 ms, and the sampling interval is 10 servo cycles.

5.5.3.2 HMC_SetSamplePara (Setting Sampling Parameters)

5.5.3.2.1 Function

Configure the sampling interval, number of sampling points, and sampling data storage area address of the specified sampling channel. It should be noted that the unit of sampling interval uiTime is servo cycle. uiTime=2 means that the sampling interval is 2 servo cycles. The LX controller supports servo cycles in 0.25, 0.5, 1, 2, and 4 ms.



HMC_SetSamplePara Functional Block

5.5.3.2.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel
uiTime	Input	USINT	It records the interval in servo cycles, which cannot be 0. Unit: servo cycle
uiNum	Input	UDINT	It records the number of points
pRecordArea	Input	POINTER TO BYTE	Starting address of the sampling record array (the sampling record array can only be defined in the M area, and the system does not allow the size of the recording area to exceed the M area; the user shall calculate the size when defining the record array to avoid out-of-bounds situations)
HMC_SetSamplePara	Output	UDINT	0: Succeeded Others: Error code

5.5.3.2.3 Instruction type

It is a non-axis motion cache instruction.

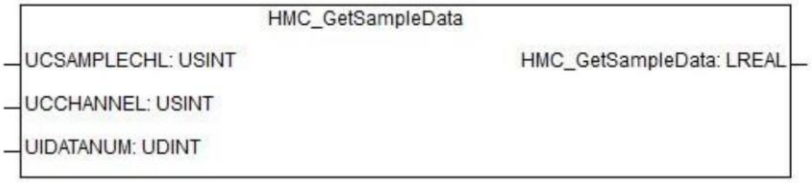
5.5.3.2.4 Example

Refer to HMC_SetSampleVar (Setting sampling variables).

5.5.3.3 HMC_GetSampleData (Getting sample data)

5.5.3.3.1 Function

Get the sample data of a sub-channel of the specified channel with the serial number uiDataNum.



HMC_SetSampleData Functional Block

5.5.3.3.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel
ucChannel	Input	USINT	Sub-channel number (0~3)
uiDataNum	Input	UDINT	Data number (uiDataNum-th data of this sub-channel), starting from 0
HMC_GetSampleData	Output	LREAL	Sample data Others: Error code

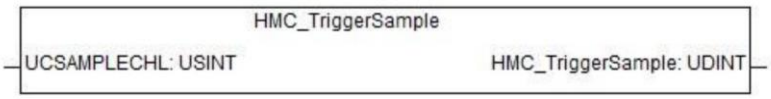
5.5.3.3.3 Instruction type

It is a non-axis motion cache instruction.

5.5.3.4 HMC_TriggerSample (*Trigger Sampling)

5.5.3.4.1 Function

Trigger the specified sampling channel to start sampling. After the relevant parameters of the sampling channel are configured, HMC_TriggerSample is used to trigger the execution of the sampling action.



HMC_TriggerSample Functional Block

5.5.3.4.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel
HMC_TriggerSample	Output	UDINT	0: Succeeded Others: error code

5.5.3.4.3 Instruction type

It is a non-axis motion cache instruction.

5.5.3.4.4 Example

Refer to HMC_SetSampleVar (Setting sampling variables)

5.5.3.5 HMC_DisableSample (Stopping Sampling)

5.5.3.5.1 Function

Disable the specified sampling channel from sampling. For the channel that is currently sampling, this function can be used to stop it from sampling. To trigger sampling again, sampling must be configured and triggered again.

HMC_DisableSample Functional Block

5.5.3.5.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel
HMC_DisableSample	Output	UDINT	0: Succeeded Others: error code

5.5.3.5.3 Instruction type

Non-axis motion cache instruction

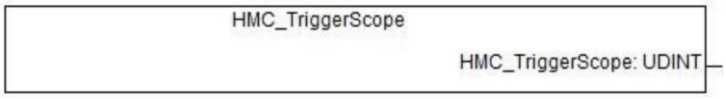
5.5.3.6 HMC_TriggerScope (Triggering Oscilloscope)

5.5.3.6.1 Function

It is used to trigger the oscilloscope to begin execution.

The oscilloscope function needs to be used with the oscilloscope software of AT. This function only works when the trigger mode is Program. In the oscilloscope window, select the trigger mode as Program, and switch the Stop/Run button to enter the run mode. By calling this functional block, the algorithm can be triggered for sampling, and the oscilloscope software draws the process curve based on the sample data.

Write an IEC program and call HMC_TriggerScope when the trigger condition is met, which can realize the function of sampling immediately after a certain condition is met.



HMC_TriggerScope Functional Block

5.5.3.6.2 Parameter

Parameter	Statement	Data Type	Description
HMC_TriggerScope	Output	UDINT	0: Succeeded Others: error code

5.5.3.6.3 Instruction type

Non-axis motion cache instruction

5.5.3.6.4 Example

Axis 0 sends a continuous forward rotation instruction. When the position of axis 0 reaches 5000, use an oscilloscope to get the speed and position information of axis 0.

Set the AT trigger mode to **Program** and switch the **Stop/Run** button to enter the run mode. Call the IEC task where the following program is located:

HMC_Forward (0);

WHILE axis0.Mpos < 5000 DO

;

END_WHILE

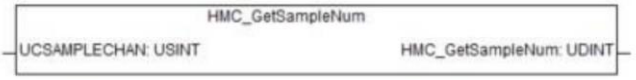
HMC_TriggerScope(); (Trigger oscilloscope sampling)

5.5.3.7 HMC_GetSampleNum (Getting the Sampled Number)

5.5.3.7.1 Function

Get the current number of completed samples of the specified sampling channel. Every time one sub-channel under this sampling channel is sampled, the number of samples accumulated will be accumulated by 1.

That is, if the number of sampling points set in HMC_SetSamplePara is N, and the sampling of four sub-channel samples set by HMC_SetSampleVar are all valid, then after the channel completes sampling, the number of samples obtained through HMC_GetSampleNum shall be 4*N.



HMC_GetSampleNum Functional Block

5.5.3.7.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel
HMC_GetSampleNum	Output	UDINT	Number of samples accumulated Others: error code

5.5.3.7.3 Instruction type

Non-axis motion cache instruction

5.5.3.8 HMC_SetSampleSingleVar (Setting a Single Sampling Variable)

5.5.3.8.1 Function

It is used to configure the source of data to be sampled for sub-channel 0 of the specified sampling channel, that is, only sampling a single variable. This function only uses sub-channel 0, and sets the data to be sampled in sub-channels 1 to 3 as empty, that is, no sampling is performed on the subsequent three sub-channels.

HMC_SetSampleSingleVar Functional Block

5.5.3.8.2 Parameter

Parameter	Statement	Data Type	Description
ucSampleChl	Input	USINT	Sampling channel
pVar	Input	POINTER TO BYTE	Address of the data to be collected
uiVarType	Input	UDINT	Data type to be sampled, including USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, F32=7, LREAL=8
HMC_SetSampleSingleVar	Output	UDINT	0: Succeeded Others: error code

5.5.3.8.3 Instruction type

It is a non-axis motion cache instruction.

5.5.4 Motion Hold Function Instructions

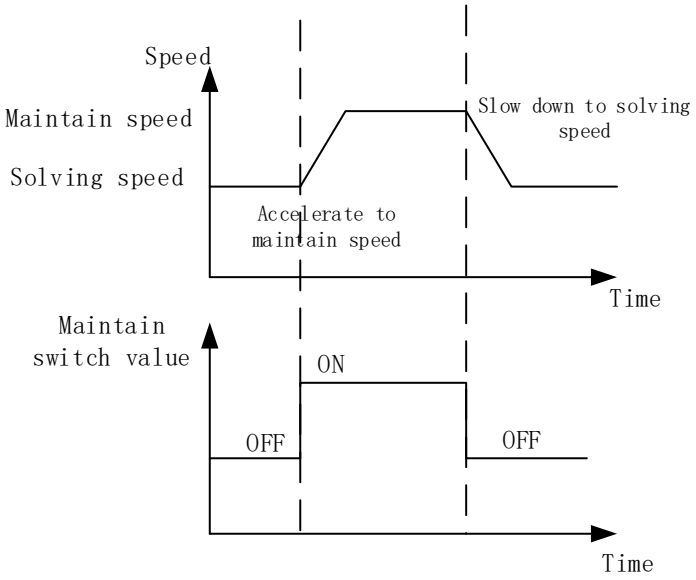
5.5.4.1 HMC_HoldConfig (Setting Hold Speed Switch)

5.5.4.1.1 Function

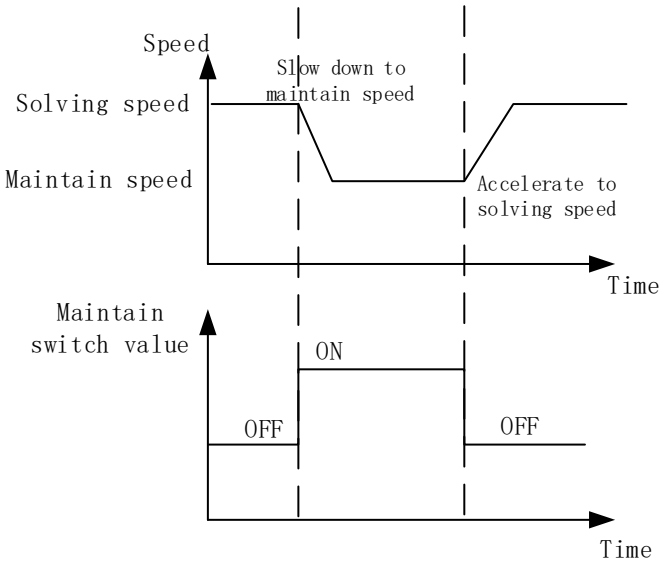
The hold speed function means that when the DI channel of the speed hold switch associated with an axis is triggered and the value is 1, the axis accelerates or decelerates from the current speed to the speed set by HoldSpeed according to Accel or Decel; when the DI channel is deactivated and the value is 0, the axis accelerates or decelerates to the instruction solving speed according to Accel or Decel.

This algorithm block is used to configure the hold speed switch of the axis to the specified DI, and the sDiChan parameter specifies the DI channel number. When the parameter sDiChan is equal to -1, it means that this function is turned off.

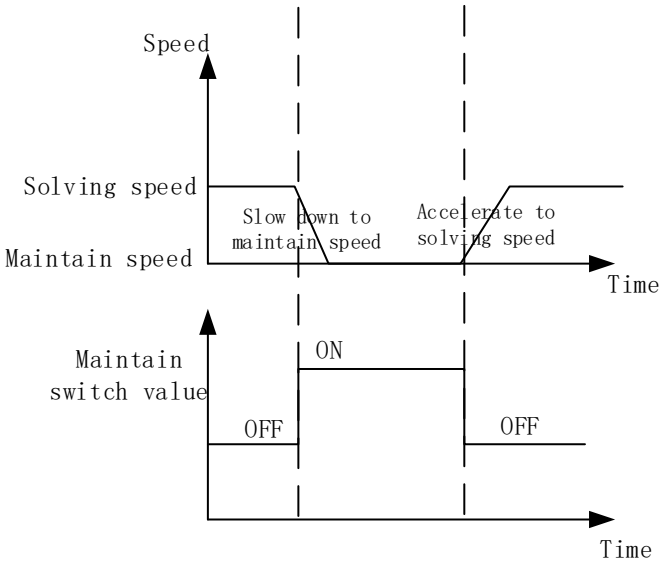
In the three common situations, including the hold speed as 0, the hold speed greater than the operation speed, and the hold speed smaller than the operation speed, the axis speed changes are as follows:



Keeping Speed Greater Than Solving Speed



Hold Speed Smaller Than Solving Speed



Keeping Speed Equal to 0

5.5.4.1.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucAxisID	Input	USINT	M or constant	Axis number
sDiChan	Input	UDINT	M or constant	DI channel number
HMC_HoldConfig	Output	UDINT	M or constant	Return value

5.5.4.1.3 Instruction type

It is a non-axis motion cache instruction.

5.5.4.1.4 Example

During the normal motion of an axis, when a certain condition is met, for example, when the pause button is pressed, the axis needs to pause its motion and remain stationary, and when the pause button pops up and the hold condition is lifted, the axis resumes its previous motion.

The button status can be detected through DI. Using the 10th DI, write a program as follows:

axis0.HoldSpeed:=0; (*Set the hold speed to 0*)

HMC_HoldConfig (0, 10); (*Configure the 10th DI as the hold speed switch of axis 0*)

HMC_Forward Ex (0, 10000); (*Send motion instruction, and axis 0 moves at a speed of 10000*)

(*When the on-site pause button is pressed, the 10th DI is detected as 1. When the algorithm detects that the DI is 1, the axis decelerates from the current speed to the hold speed of 0 to realize axis pause; when the on-site pause button pops up, the 10th DI is 0 and the algorithm detects that the DI is 0, the axis accelerates from the speed of 0 to the set speed of 10000*)

5.5.5 Motion Look-Ahead

This chapter describes the motion look-ahead function, which is used to continuously track, count, and analyze the axis motion, and obtain the solution results within the planned time. It takes a comprehensive consideration of the multiple influencing factors of the axis motion.

The controller provides an intra-segment trapezoidal or S-shaped acceleration and deceleration control function. In other words, it performs acceleration and deceleration processing for each segment. The speed between instructions is independent. Therefore, at the junction of each instruction segment, the axis will be sure to have a process that the speed first decelerates to zero, and then accelerates again after the next instruction is started. Macroscopically, it is a continuous acceleration and deceleration process. In actual application, it not only affects the processing efficiency, but also produces a huge acceleration shock on the machine tool when processing extremely short continuous micro-line segments, causing machine tool vibration and even overcutting of the workpiece. Therefore, more advanced speed control methods are needed.

The speed look-ahead function of the system can, on the one hand, make an overall plan for the instructions, that is, make an overall plan for the speed of each segment, and then cooperate with the acceleration and deceleration control within the instruction segment to keep the machine tool running at high speed, improve efficiency, and make the load motion smoother. Saying goodbye to the previous go-and-stop motion mode, the system realizes smooth motion through the Merge function. On the other hand, to limit mechanical shock and over-cutting at the same time of ensuring high-speed operation, deceleration recognition is also required to identify trajectory changes in advance to achieve a safe deceleration in advance, which can be realized by the system through the deceleration/stop merge function and the shock suppression function. Overall, the speed look-ahead function can not only improve the overall machine efficiency, but also reduce shock, increase flexibility, reduce component wear, and increase device service life.

The system's motion look-ahead function includes two categories of motion as follows: 1: merge function, which includes speed merge and deceleration/stop merge functions, and 2: shock suppression function.

1. Merge (Merge function)

The Merge function is to connect a series of motion instructions in the controller's motion cache to make the load motion smoother, thus saying goodbye to the stop-and-go mode. It is aimed at processing between instructions: after the Merge function is enabled, the speed connection is realized, and the speed between instructions no longer decelerates to 0; on the other hand, in order to avoid the shock caused by a large speed when the angle between instructions is too large, the deceleration/stop angle merge function is introduced.

Effective scope

To enable the Merge function successfully, it not only depends on the Merge switch, but also depends on the instruction type, the unification of the axes involved in the instruction, and other conditions, as follows:

Currently, it only supports the Merge processing of one-dimensional instructions (single-axis instructions), two-dimensional instructions (two-axis interpolation instructions), and N-axis interpolation instructions. It also supports S-shaped acceleration and deceleration merging.

Among single-axis instructions, only HMC_Move, HMC_MoveEx, HMC_MoveAbs, and HMC_MoveAbsEx

are supported.

Among two-axis interpolation instructions, only HMC_Move2D, HMC_Move2DEx, HMC_MoveAbs2D, HMC_MoveAbs2DEx, HMC_MoveCircle, and HMC_MoveCircleEx are supported.

Among N-axis interpolation instructions, only HMC_MoveND, HMC_MoveAbsND, HMC_MoveNDEx, and HMC_MoveAbsNDEx are supported.

In the two-axis/N-axis interpolation instructions, if the axis number sequence corresponding to the interpolated combined axis and split axis changes, Merge is invalid.

Merge is invalid when instructions with different dimensions (number of axes) are connected to each other.

If the current instruction is executing HMC_MoveModify, the HMC_SetMerge parameter setting fails.

In two-dimensional and N-dimensional instructions, the instruction rotation angle θ (the connection angle of two adjacent instructions connected by Merge) of the Nth segment and (N+1)th segment is not allowed to be too large. Merge is normal when the instruction rotation angle θ is smaller than the deceleration angle (the deceleration angle is set by the HMC_SetMergeAngle function). Merge is invalid when the deceleration angle θ is greater than or equal to the stop angle (the stop angle is set by the HMC_SetMergeAngle function). When the instruction rotation angle θ is greater than or equal to the deceleration angle and smaller than the stop angle, the current instruction ends at a speed smaller than Speed (proportionally).

If the Merge function is enabled, it will be automatically disabled after the HMC_MoveModify instruction is sent, and the user needs to manually re-enable the Merge function.

If the Merge function is enabled, it will be automatically disabled after the Cancel is used to cancel the instruction, and the user needs to manually re-enable the Merge function.

Enabling Merge function

The HMC_SetMerge function can be used to set whether and how the Merge function is enabled.

Merge=0

Merge is disabled and the speed at the end of the current instruction block is 0

Merge= 1

Merge on, in which case the target speed at the end of the current instruction block is the Speed of the instruction block

Merge=2

Merge on, in which case the target speed at the end of the current instruction block is the Speed of the next instruction block

Merge=3

Merge on, in which case the target speed at the end of the current instruction block is the minimum value of the Speed of the instruction block and the Speed of the next instruction block

Merge=4

Merge on, in which case the target speed at the end of the current instruction block is the maximum value of the Speed of the instruction block and the Speed of the next instruction block

The main design guidelines for Merge are as follows:

At the connection points of D instructions, if the remaining displacement of this instruction is insufficient, part of the displacement of the next instruction shall be output in advance to prevent the speed jump in the last cycle of each instruction.

At the connection points of 2D interpolation instructions, the interpolation trajectory is not required to pass through the junction point of the two instructions. If the remaining displacement of this instruction is insufficient, part of the displacement of the next instruction shall be output in advance.

For EX instructions to change the speed:

☐ Try to ensure that the preset speed of all EX instruction connection points can be achieved. If only the preset speed of some connection points can be guaranteed, priority shall be given to those with lower speeds. For connection points that cannot reach the preset speed, ensure that the actual operation speed is smaller than its preset speed.

☐ It must be ensured that the change process of VpSpeed always meets the preset trapezoidal/S-shaped law.

☐ Under the premise of meeting the constraints required by the above guidelines, each instruction shall be guaranteed to move at its speed as the target speed as much as possible during its life cycle.

For instructions with corner speed limit/Jerk speed limit:

☐ During the step-by-step look-ahead calculation process, it shall determine whether the current look-ahead connection point requires a corner speed limit, and if so, the speed limit shall be calculated.

☐ Try to ensure that the limit speed of all instruction connection points can be achieved. If only the limit speed of some connection points can be guaranteed, priority shall be given to those with lower limit speeds. For connection points that cannot reach the limit speed, ensure that the actual operation speed is smaller than its limit speed.

☐ During the motion process, it must be ensured that the change process of VpSpeed always meets the preset trapezoidal/S-shaped law;

☐ Under the premise of meeting the constraints required by the above guidelines, each instruction shall be guaranteed to move at its speed as the target speed as much as possible during its life cycle.

The following conditions can be used as the end criteria for look-ahead calculations:

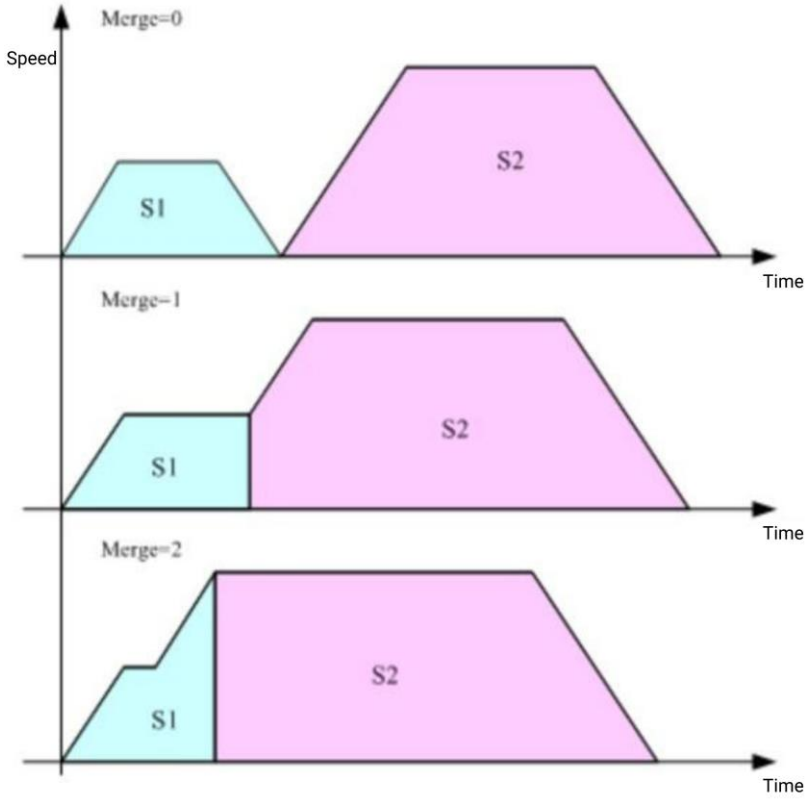
☐ If the look-ahead has reached the end of the instruction cache queue, the look-ahead ends.

☐ If the look-ahead calculation has not yet reached the end of the instruction cache queue, but, under current conditions, it can be guaranteed that even if instructions outside the look-ahead range are

subject to any low speed limit, they can be guaranteed to be able to follow the preset trapezoidal/S-shaped acceleration and deceleration rules to reach the speed provided that the displacement conditions are met, the current look-ahead calculation can be ended at this time.

This can effectively reduce the look-ahead calculation depth and save calculation time.

For example, when the current instruction speed is 1000 and the target speed of the next instruction is 2000, the calculation results in the five modes are as follows:



HMC_SetMerge (Setting merging function)

2. Deceleration/stop merge function

The deceleration/stop merge function can solve the following problem: when the angle between instructions is too large if it is still running at a large speed, a large mechanical shock will be generated at the angle and the trajectory will deviate.

The controller will identify the angle of trajectory change between instructions in advance, compare it with the deceleration/stop angle, and decide in advance whether to decelerate to ensure a smooth transition at the instruction connection.

The angle between instructions < deceleration angle

The deceleration/stop angle does not limit the speed, and the Merge function determines the speed at the end of the instruction.

Deceleration angle < angle between instructions < stop angle

The speed is limited, and then the limited speed is k times the end speed of the instruction set by the Merge function. The proportional coefficient k belongs to the range of [0, 1],

$$k = \frac{\text{Stop angle} - \text{Angle between instructions}}{\text{Stop angle} - \text{Deceleration angle}}$$

Stop angle < angle between instructions

The instruction end speed is 0.

As shown below, it walks the following trajectory on a two-dimensional plane and executes 4 instructions through points O->A->B->C->D to complete the entire trajectory. It is assumed that the parameter Speed of the combined axis is not modified in other ways among the four instructions.

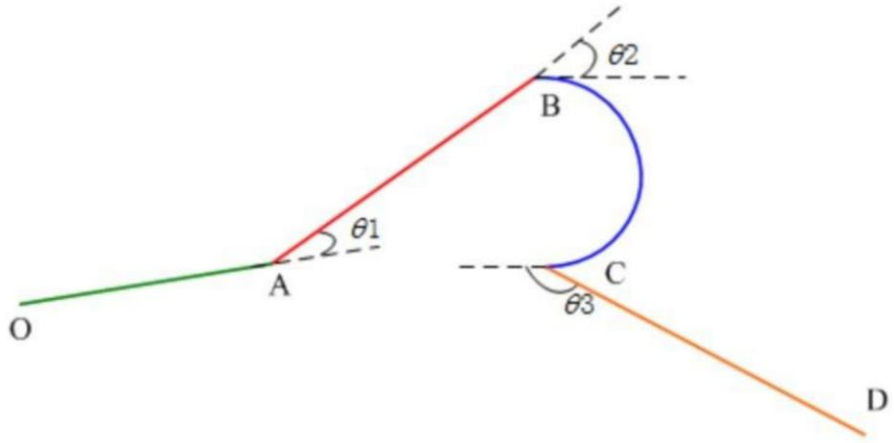
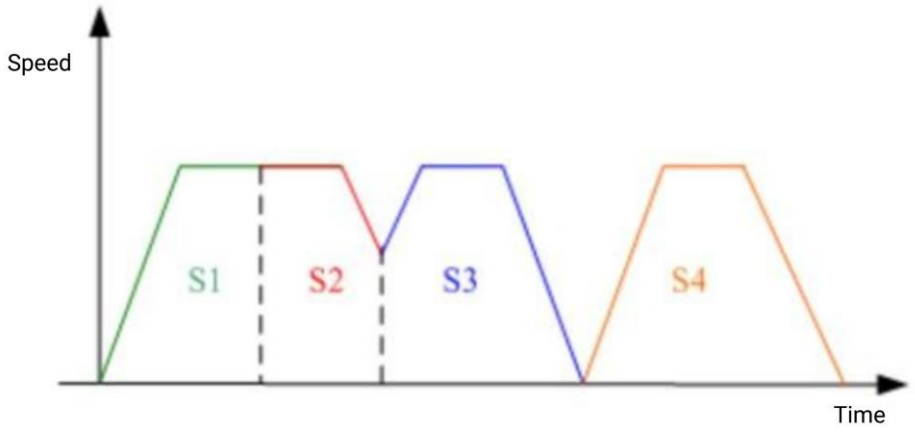


Diagram of Motion Trajectory

θ1 is the angle between instructions 1 and 2, and θ2 and θ3 are the angles between subsequent adjacent instructions. Assuming that the angle satisfies the relationship: θ1 < deceleration angle < θ2 < stop angle < θ3, then the speed change of the combined axis is as shown in the figure:



Combined Axis Speed Diagram

The shock suppression function mainly solves the following problem: when the curvature of the motion trajectory is large, if the speed is too high, it will bring about a large inertial centrifugal force, causing mechanical

shock to damage the workpiece, and even leading to trajectory deviation and misalignment.

Effective scope

Different from Merge which only works between multiple instructions, the shock suppression function not only supports multiple 2D instructions but also is valid for a single instruction. The details are as follows:

Single instructions, no need to enable Merge

- ☐ Planar arc interpolation instructions, spherical arc interpolation instructions;
- ☐ Helical interpolation instruction;
- ☐ Spline interpolation instructions;

Between multiple instructions, Merge shall be enabled

- ☐ For any combination of two-axis linear interpolation (HMC_Move2D, HMC_Move2DEx) and planar arc interpolation (HMC_MoveCircle, HMC_MoveCircleEx), the axis number sequence corresponding to the interpolated combined axis and split axis must be the same.

Shock suppression

Calculate the curvature ρ of the instruction trajectory, and then calculate the maximum speed based on the curvature and Jerk.

$V_{max}=f(\rho, Jerk)$

Compare it with the current target speed of the instruction to determine whether it needs to be limited for deceleration to ensure that the mechanical shock during the entire trajectory is limited within a reasonable range and to ensure the accuracy of the trajectory.

For example, to execute an arc instruction, set the same Speed=20000, but different Jerk values, the speed of the combined axis will change during actual operation.

At the default value of Jerk=5000000000, the combined axis speed reaches 20000, and the instruction takes a short time to complete.

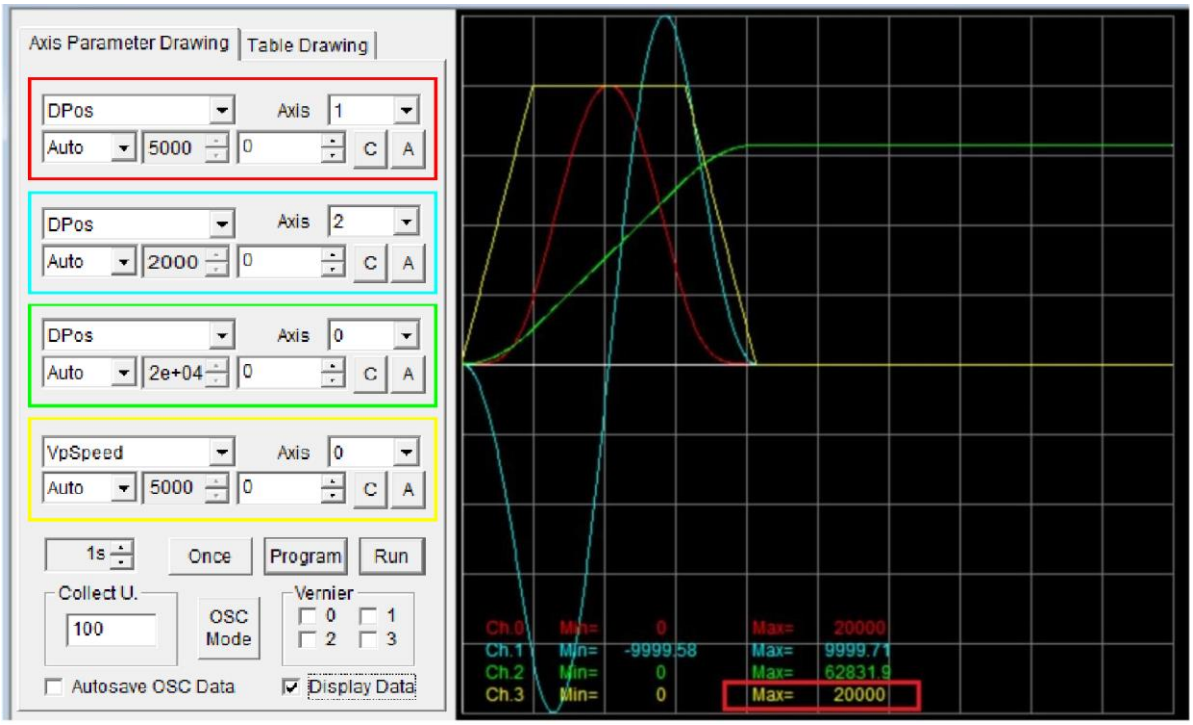


Diagram of Motion Trajectory

When it is modified to Jerk=5000, the maximum speed of the combined axis only reaches 7937.01, and the instruction takes a long time to complete. HMC_SetMerge (Setting merging function)

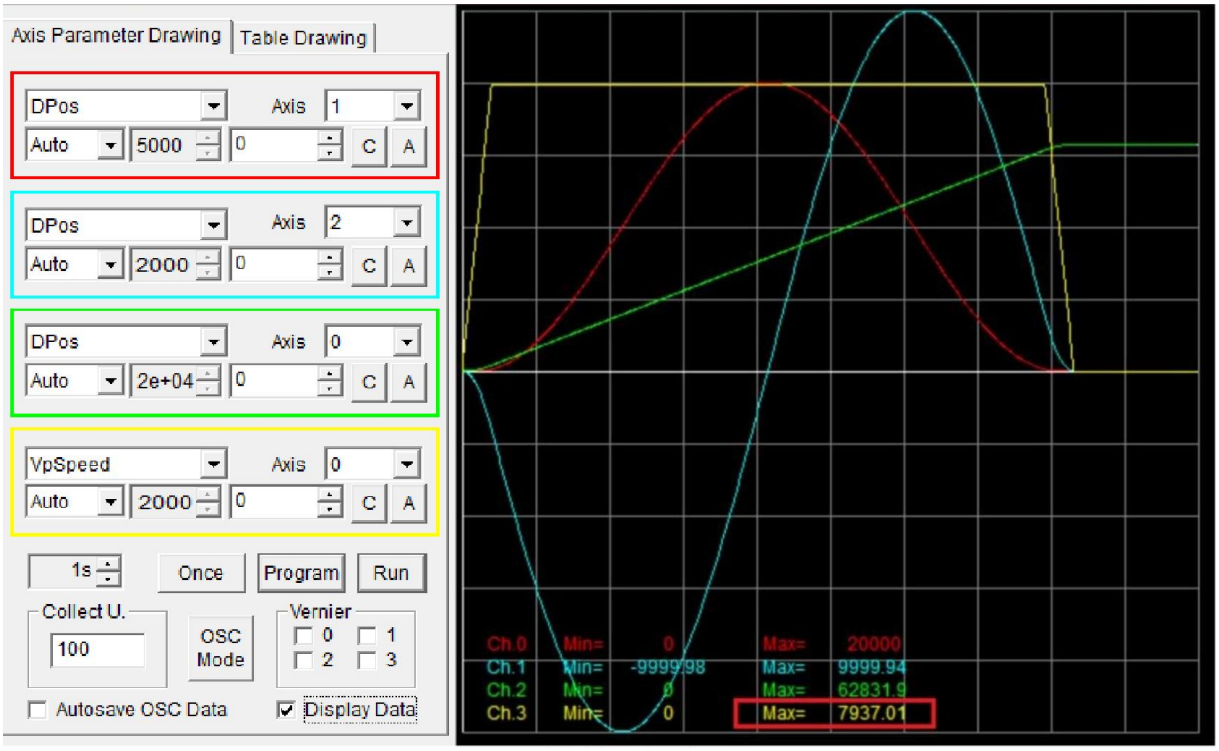
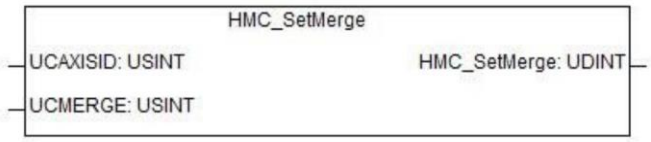


Diagram of Functional Motion Trajectory

5.5.5.1 HMC_SetMerge (Setting Merging Function)

5.5.5.1.1 Function

Set the Merge mode of axis parameters. For detailed descriptions of Merge, see the axis parameter Merge and the description of motion look-ahead function in this chapter.



HMC_SetMerge Functional Block

5.5.5.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucMerge	Input	USINT	Merge mode, which can only be 0, 1, 2, 3, 4
HMC_SetMerge	Output	UDINT	Return value

5.5.5.1.3 Instruction type

It is a non-axis motion cache instruction.

5.5.5.1.4 Example

The procedure is as follows:

axis0.Speed := 20000; (*Axis 0's speed is 20000 (user units/second)*)

HMC_Move(0, 60000); (*Instruction 0 moves forward by 60000 (user units)*)

HMC_Forward Ex(0, 40000); (*Instruction 1: set axis0.Speed to 40000 (user units/second) and continue to move forward*)

When axis0.Merge is 0, 1, or 2, the curve shapes are as follows (S0 is 60000 user units):

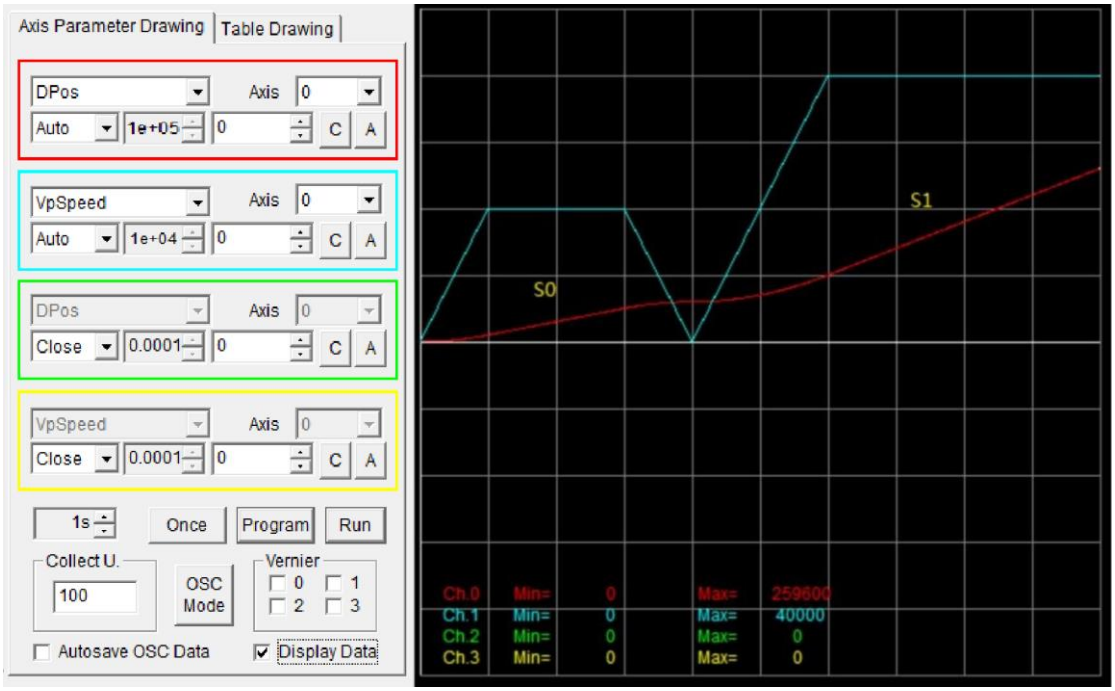


Diagram of Merge as 0

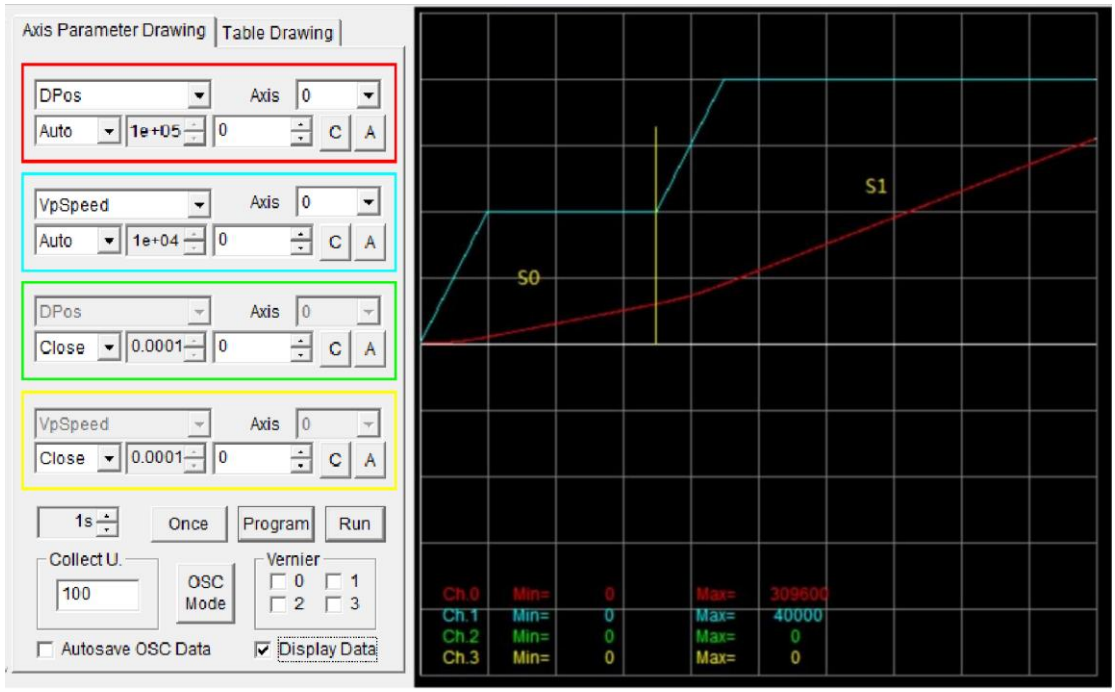


Diagram of Merge as 1

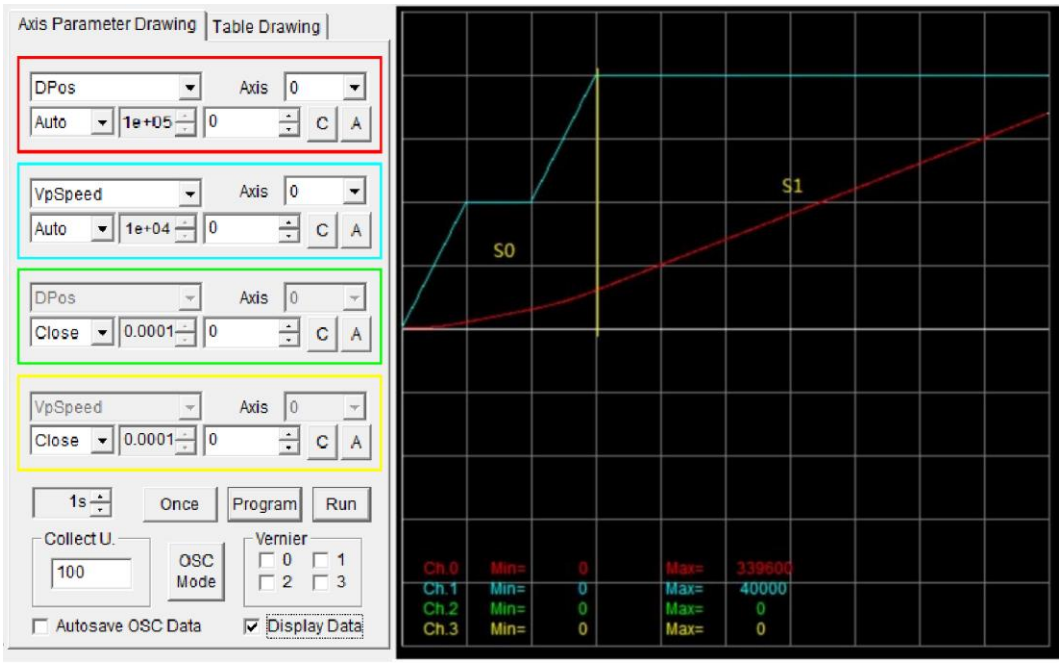


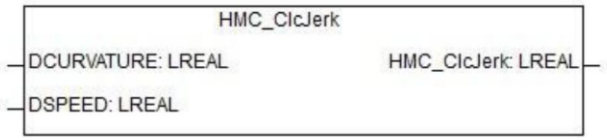
Diagram of Merge as 2

5.5.5.2 HMC_ClcJerk (Shock operation)

5.5.5.2.1 Function

Calculate Jerk based on Speed and curvature.

Jerk=dCurvature² * dSpeed³



HMC_ClcJerk Functional Block

5.5.5.2.2 Parameter

Parameter	Statement	Data Type	Description
dCurvature	Input	LREAL	Curvature (non-negative)
dSpeed	Input	LREAL	Speed (non-negative)
HMC_ClcJerk	Output	LREAL	Other values: Return Jerk value -1: Parameter error

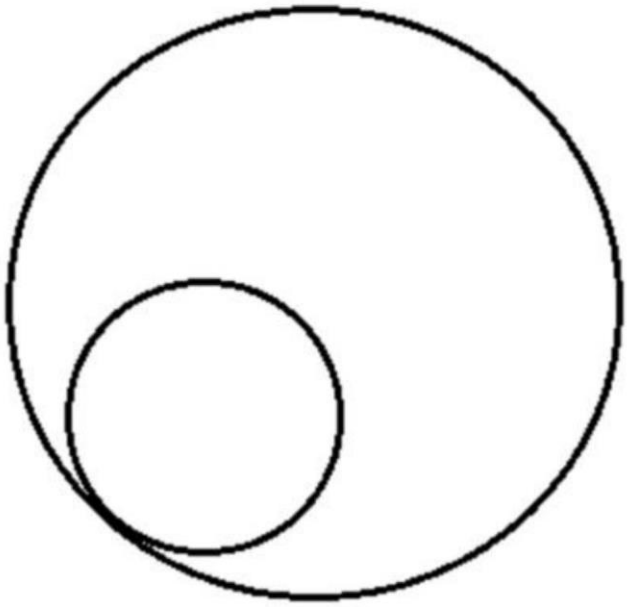
5.5.5.2.3 Instruction type

It is a non-axis motion cache instruction.

5.5.5.2.4 Example

To follow the trajectory below, start from the intersection of the two circles, first pass through the big circle to

the starting point, then pass through the small circle and finally reach the starting point. Assume that the maximum Jerk the device can withstand is 100000 and the desired speed is 1000.



Circles with Different Trajectories

Considering that Jerk is related to speed and curvature and the curvatures of the above two circles with different trajectories are different, if Jerk is set too small, the actual operation speed will be limited and the efficiency will be affected, but if Jerk is set too large and exceeds the maximum jerk that the device can withstand, it will affect the safety of the device. Therefore, after calculating the Jerk for the two circles respectively, compare it with the maximum Jerk. If it is greater than the maximum Jerk the system can withstand, set it to the maximum Jerk. Otherwise, use the calculated Jerk. Write the program as follows:

```
Axis7.Speed := 1000;

Axis7.Accel := 5000;

Axis7.Decel:= 5000;

HMC_MoveCircle(7, 0, 1, 1, 0, 0, 100, 100); (*First big circle*)

Jerk1 := HMC_ClcJerk( 1/(100*SQRT(2)), 1000); (*Shock operation function*)

MaxJerk := 100000;

IF Jerk1 > MaxJerk THEN (*Compared with the maximum shock the device can withstand in the principle of safety first and efficiency second*)

    axis7.Jerk := MaxJerk;

ELSE

    axis7.Jerk := Jerk1;

END_IF
```

HMC_MoveCircle(7, 0, 1, 1, 0, 0, 50, 50);

Jerk2 := HMC_ClcJerk(1/(50*SQRT(2)), 1000); (*Shock operation function*)

MaxJerk := 100000;

IF Jerk2 > MaxJerk THEN (*Compared with the maximum shock the device can withstand in the principle of safety first and efficiency second*)

axis7.Jerk := MaxJerk;

ELSE

axis7.Jerk := Jerk2;

END_IF

The result after execution is as follows:

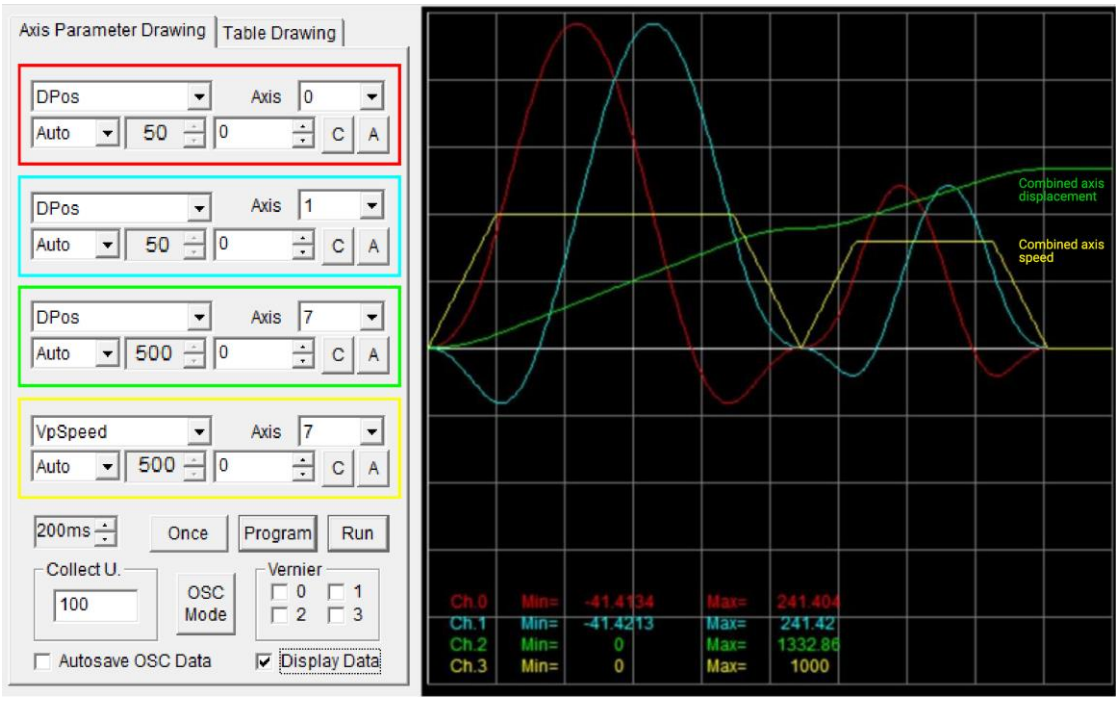


Diagram of Motion Trajectory

As can be seen from the above figure, although the same instruction speed Speed=1000 is set, the actual operation speed of the small circle trajectory does not reach 1000. This is because the curvature of the small circle is large, when it is running at a speed of 1000, the shock that the device needs to withstand exceeds the maximum shock of 100000 that the device can withstand. At this time, if Jerk = 100000 is set, it will limit the real-time settlement speed.

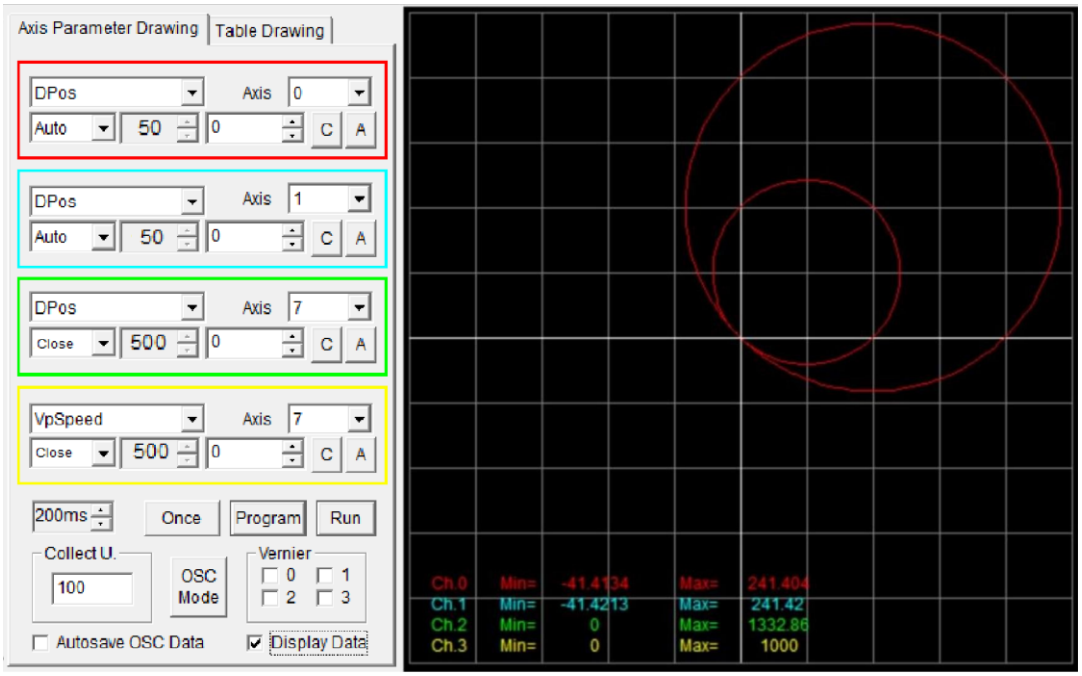
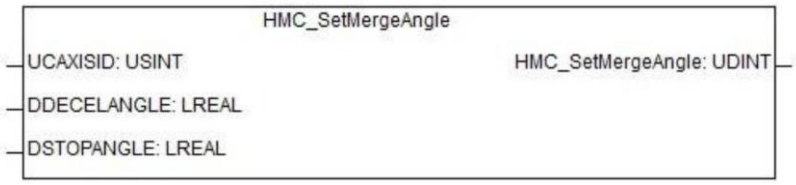


Diagram of Motion Trajectory

5.5.5.3 HMC_SetMergeAngle (Setting Merge Angle)

5.5.5.3.1 Function

Set the merging angle of the axis.



HMC_SetMergeAngle Functional Block

5.5.5.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
d DecelAngle	Input	LREAL	Set the deceleration angle of the two-dimensional instruction, which is valid for the combined axis setting and cannot be negative
dStopAngle	Input	LREAL	Set the stop angle of the two-dimensional instruction, which is valid for the combined axis setting and cannot be negative
HMC_SetMergeAngle	Output	UDINT	0: Set successfully Other values: Error code

5.5.5.3.3 Instruction type

It is a non-axis motion cache instruction.

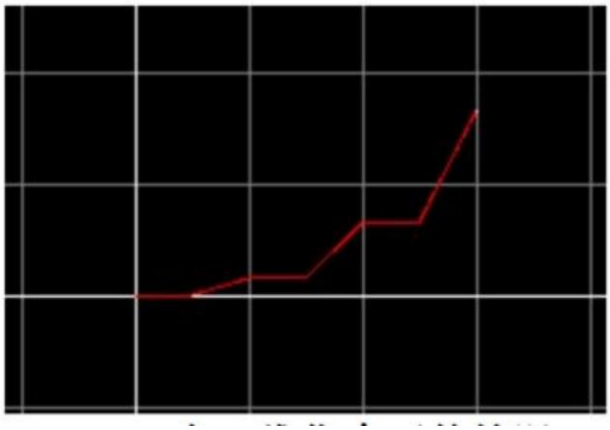
d DecelAngle≤π/6, dStopAngle≤π/2 and d DecelAngle≤dStopAngle.

The default value of d DecelAngle is π/6, and the default value of dStopAngle is π/2.

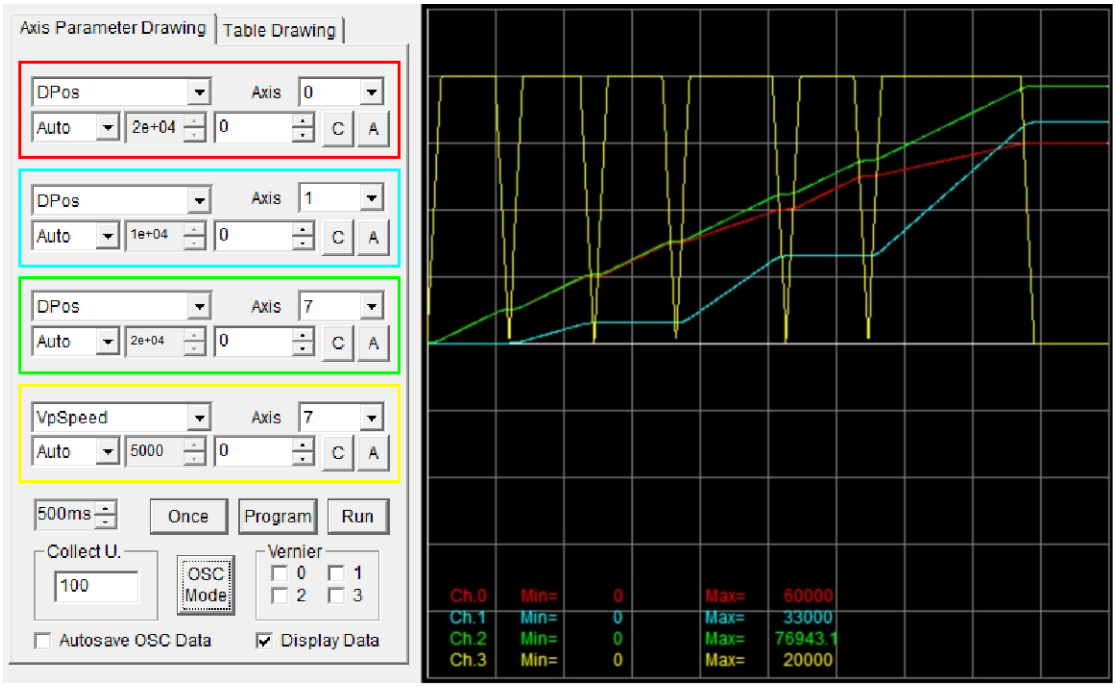
5.5.5.3.4 Example

For Merge for two-axis instructions, the program is as follows:

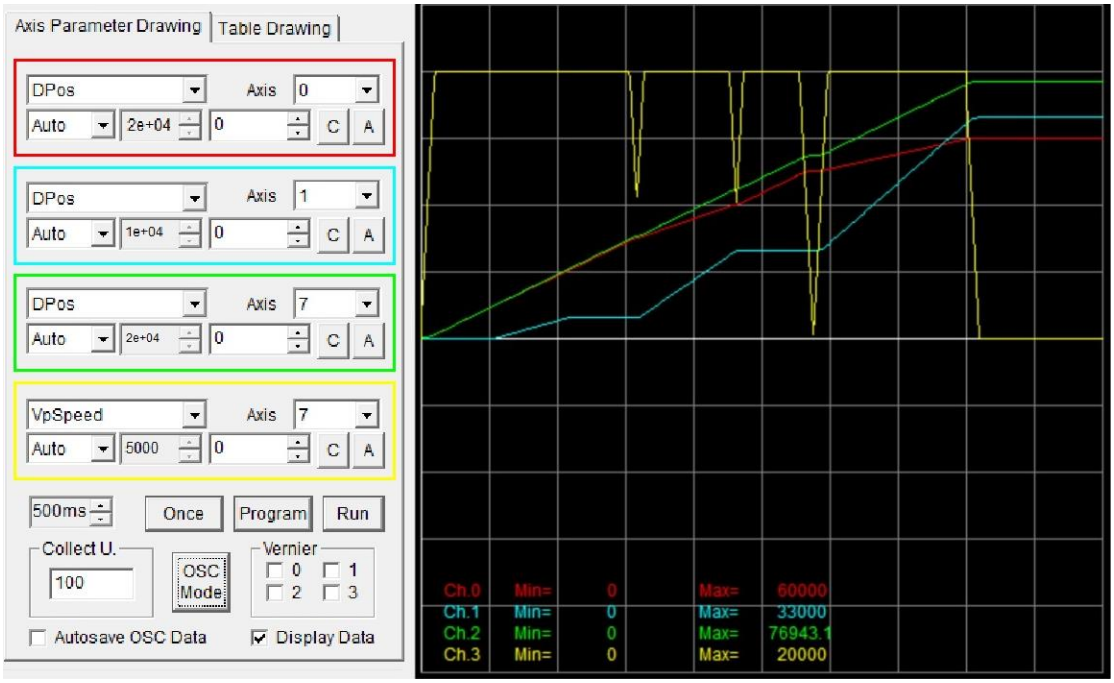
```
axis0.Speed := 20000; (*Axis 0's speed is 20,000 (user units/second)*)
axis0.Accel := 200000;
axis0.Decel := 200000;
HMC_Move2D(7, 0, 1, 10000, 0);
HMC_Move2D(7, 0, 1, 10000, 3000);
HMC_Move2D(7, 0, 1, 10000, 0);
HMC_Move2D(7, 0, 1, 10000, 10000);
HMC_Move2D(7, 0, 1, 10000, 0);
HMC_Move2D(7, 0, 1, 10000, 20000);
```



Curve When Merge in Two-dimensional Instructions



CUrve When Merge is 0

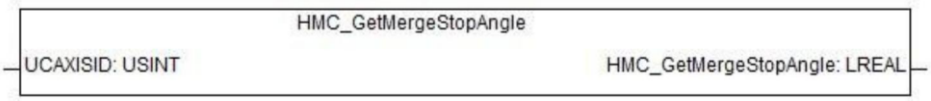


Curve When Merge is 1

5.5.5.4 HMC_GetMergeStopAngle (Getting Merge Stop Angle)

5.5.5.4.1 Function

Get the interpolation motion stop angle.



HMC_GetMergeStopAngle Functional Block

5.5.5.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_GetMergeStopAngel	Output	LREAL	Radian

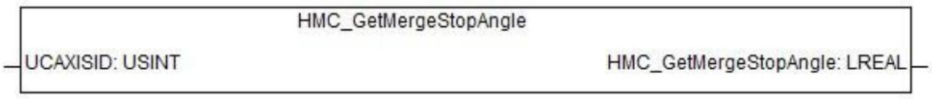
5.5.5.4.3 Instruction type

Non-axis motion cache instruction

5.5.5.5 HMC_GetMergeDecelAngle (Getting Merge Deceleration Angle)

5.5.5.5.1 Function

Get the interpolation motion deceleration angle.



HMC_GetMergeDeceAngle Functional Block

5.5.5.5.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	UINT	Axis number
HMC_GetMergeDecelAngel	Output	LREAL	Radian

5.5.5.5.3 Instruction type

It is a non-axis motion cache instruction.

5.5.5.6 HMC_SetMergeSpeedTolerRatio (Setting Speed Fluctuation Rejection Ratio)

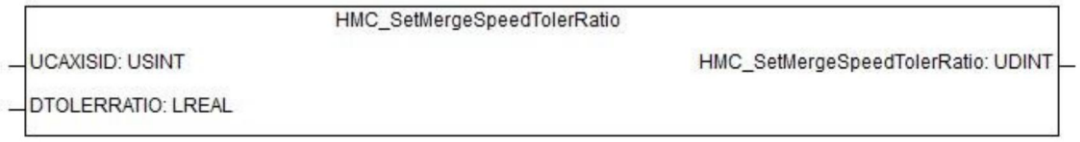
5.5.5.6.1 Function

This instruction sets the speed fluctuation rejection ratio parameter during the motion look-ahead process. The speed fluctuation rejection function allows the speed fluctuation to be suppressed within the error ratio set by the user.

During the motion look-ahead process, to reduce the shock, the operation speed is constrained by the curvature speed limit and corner speed limit mechanisms, so the speed often fluctuates to a certain extent. When small polylines are used to approximate curves, if the accuracy of the original point data provided by the user is too low, the constrained speed calculated by the curvature speed limit and corner speed limit mechanisms may frequently jump up and down, resulting in high-frequency fluctuations in speed.

The speed fluctuation rejection function can solve the above problems and reduce or even eliminate such fluctuations. The larger the speed fluctuation rejection ratio dTolerRatio, the stronger the rejection effect, but if it is

too large, it may increase the system shock. Therefore, within the acceptable speed fluctuation range, try to keep dTolerRatio as small as possible. If the S-shaped acceleration and deceleration are enabled and an appropriate jerk value is set, a better fluctuation rejection effect can be achieved under the same dTolerRatio.



HMC_SetMergeSpeedTolerRatio Functional Block

5.5.5.6.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dTolerRatio	Input	LREAL	Speed fluctuation rejection ratio [0, 1] The larger the value, the stronger the rejection effect, but if it is too large, it may increase the system shock.
HMC_SetMergeSpeedTolerRatio	Output	UDINT	0: Succeeded Error code: Parameter error

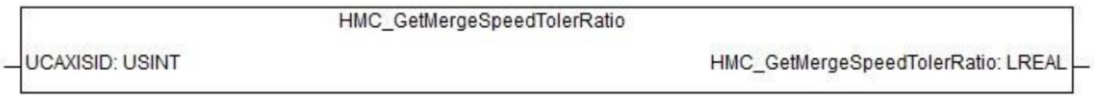
5.5.5.6.3 Instruction type

It is a non-axis motion cache instruction.

5.5.5.7 HMC_GetMergeSpeedTolerRatio (Reading Speed Fluctuation Rejection Ratio)

5.5.5.7.1 Function

Read the current speed fluctuation rejection ratio.



HMC_GetMergeSpeedTolerRatio Functional Block

5.5.5.7.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_GetMergeSpeedToleRatio	Output	LREAL	Current speed fluctuation rejection ratio

5.5.5.7.3 Instruction type

It is a non-axis motion cache instruction.

5.5.6 Curve Speed Limit

5.5.6.1 HMC_SetCurveJerkLimit (Setting the Upper Limit of Curve Shock)

5.5.6.1.1 Function

This instruction sets the upper limit of physical shock during curve interpolation, thereby limiting the curve interpolation speed.

By setting this parameter appropriately, during the interpolation process, the places with large curvature of the curve (such as small arcs) will have greater deceleration, while the places with small curvature (such as large arcs) will have less or no deceleration.

The shock parameter value that meets the specific device requirements can be found through online debugging of the device. The HMC_ClcJerk instruction can also be used to calculate the shock value at a specific curvature (reciprocal of radius) and specific speed as the upper limit value of shock.

5.5.6.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dLimitVal	Input	LREAL	Upper limit value set, non-negative
HMC_SetCurveJerkLimit	Output	UDINT	0: Succeeded Error code: Parameter error

5.5.6.1.3 Instruction type

It is a non-axis motion cache instruction.

5.5.6.2 HMC_GetCurveJerkLimit (Reading the Upper Limit of Curve Shock)

5.5.6.2.1 Function

Read the set upper limit of physical shock during curve interpolation.

5.5.6.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_GetCurveJerkLimit	Output	LREAL	0 is returned when there is an instruction parameter error in the upper limit of the physical shock in curve interpolation

5.5.6.2.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7 Tangential Tracking

5.5.7.1 HMC_MoveTang (Tangential Tracking)

5.5.7.1.1 Function

Based on the motion trajectories of the X-axis and Y-axis, this functional block calculates the angle between the combined motion trajectory and the positive X-axis and drives the cutter rotation axis C to track until the target angle. It can be used in material cutting scenarios to automatically adjust the tool direction in real time to keep consistent with the tangential direction of the cutting trajectory (cutting speed vector speed) and reach the optimal cutting effect. The offset dOffset of the tool inclination angle can be set to adjust the rake and relief angles of the tool during cutting.

During the execution of this instruction, the C-axis (tool rotation axis) works in cycle stroke mode, and the mode RepMode is 2 (symmetrical interval). Before using this instruction, the user must first set the cycle stroke length ([-Repdist, Repdist)) to the displacement amount in user units corresponding to one rotation of the axis (assumed to be RoundLength). That is,

Repdist = RoundLength/2

In this way, the absolute position of the C-axis [-Repdist, Repdist) corresponds to the tangential angle of [-π, π).

During the execution of the instruction, calculate in real time the angle TangAngle formed by the tangential direction of the combined motion trajectory of the X-axis (axis number: ucAxisX) and Y-axis (axis number: ucAxisY) and the forward direction of the X-axis, which is the target tangential angle of the C-axis; at the same time, combine it with the rate of change of the target tangential angle (angular velocity). Adjust the C-axis target position in real time and drive the C-axis (axis number: ucAxisC) to turn to the target tangential angle to achieve optimal tracking of the combined motion trajectory angle of the two axes.

When using the tangential tracking function, it is best for the user to first estimate the maximum acceleration and deceleration capacity of the C-axis of the tool rotation axis and the maximum operation speed. These parameters can be estimated based on the rated torque, rated speed, and other parameters of the motor in combination with the load conditions and the requirements of the device's mechanical structure on vibration and torque. Based on the above calculation, set the parameters of the C-axis such as acceleration, deceleration, and speed to the maximum values allowed by the device (or multiply by a certain safety margin factor). During the execution of tangential tracking, the algorithm will drive the C-axis to the target tangential angle according to the motion parameters that do not exceed the acceleration, deceleration, and speed set by the C-axis.

If the maximum acceleration and deceleration capability of the C-axis is smaller than the acceleration/deceleration of the actual tangential angle change, or the maximum operation speed is smaller than the maximum speed of the actual tangential angle change, the C-axis cannot follow the tangential angle without deviation during MoveTang operation.

During the execution of the instruction, when the target tangential angle has jump-changes or the change speed

(angular speed) of the target tangential angle has jump-changes, the tool rotation C-axis will move towards the target position with the acceleration, deceleration, and speed set in the axis parameters, without any shock caused by a jump-change in operation speed. However, there will be a certain following deviation between the actual angular position of the C-axis and the target cutting angle.

During device debugging, HMC_GetMoveTangDeviation can be used to obtain the following deviation of tangential angle in real time and view the actual operation effect of tangential tracking. If the following deviation exceeds the range required by the application, the HMC_GetMoveTangDestDirection and HMC_GetMoveTangDestVelocity instructions can be used to obtain the target tangential angle position and tangential angle change speed respectively. By comparing the DPOS change curve and the VpSpeed change curve of the C-axis, the specific reasons for the deviation can be analyzed, and generally can be divided into the following three categories:

(1) The acceleration/deceleration/Speed of the C-axis is too small and cannot follow the tangential angle change. Solution:

Increase the acceleration/deceleration/Speed of the C-axis;

Reduce the XY-axis interpolation speed to reduce the tangential angle change rate;

If the XY-axis interpolation speed cannot be reduced, and the acceleration/deceleration/Speed of the C-axis has reached the C-axis motor torque limit or mechanical structure limit, consider replacing a motor with a larger torque or changing the mechanical structure.

(2) There is a jump-change in the tangential angle position. (There are first-order non-differentiable points in the XY-axis interpolation trajectory, such as two polylines). Solution:

Optimize the XY-axis interpolation path. For example, for a polyline path, arcs can be inserted between the polylines.

If what is needed is a polyline trajectory, an auxiliary advance and retreat path can be inserted, so that the tool first cuts out from the extension line of one section of the polyline, and then cuts in from the extension line of the other section of the polyline.

Increasing the acceleration/deceleration/Speed of the C-axis or reducing the XY-axis interpolation speed at the transition point can reduce the influence range of this deviation, but it cannot be eliminated.

(3) There is a jump-change in the change speed of the tangential angle. Solution:

Optimize the XY-axis interpolation path. For example, by using spline interpolation trajectories.

If the interpolation path cannot be changed, you can increase the acceleration/deceleration/Speed of the C-axis or reduce the XY-axis interpolation speed at the transition point.

5.5.7.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number

ucAxisX	Input	USINT	Motion axis number in the X direction
ucAxisY	Input	USINT	Motion axis number in the Y direction
dOffset	Input	LREAL	Tool rotation axis offset (in user units)
JMC_MoveTang	Output	UDINT	Axis instruction ID: Success 0Xffffffff: Function parameter error

5.5.7.1.3 Instruction type

Axis motion cache instructions.

Additional information:

(1) Once the instruction is successfully sent, it will remain in effect until canceled by executing the cancel instruction.

(2) Tangential tracking supports any instruction type for the X and Y axes.

(3) The maximum speed, acceleration, and deceleration of the C-axis of the tool rotation axis are determined by its axis parameter settings. Appropriate values shall be set according to the motion conditions and the characteristics of the axis itself (make sure that the values are large enough within the range that the axis itself can withstand) to ensure that the C-axis can quickly follow the tangential angle.

(4) When a sharp angle change occurs at the connection of the two motion trajectories, the program will automatically select the shortest path to track and locate the tangential angle, and no spinning will occur.

(5) During the execution of the instruction, when the target tangential angle has jump-changes or the change speed (angular speed) of the target tangential angle has jump-changes, the tool rotation C-axis will move towards the target position with the acceleration, deceleration, and speed set in the axis parameters, without any shock caused by a jump-change in operation speed. However, there will be a certain following deviation between the actual angular position of the C-axis and the target cutting angle.

(6) When the X/Y axis encounters a limit, the tangential angle will be calculated according to the actual deceleration trajectory during deceleration and stop. After the limit stop action is completed, if the stroke over-limit status has not been released, and the subsequent instructions sent to the master axes X and Y do not run in the opposite direction of the X/Y limit (reducing the position over-limit direction), it will be processed as an instruction usage exception. For safety reasons, the C-axis of the tool rotation axis will remain in place at this time.

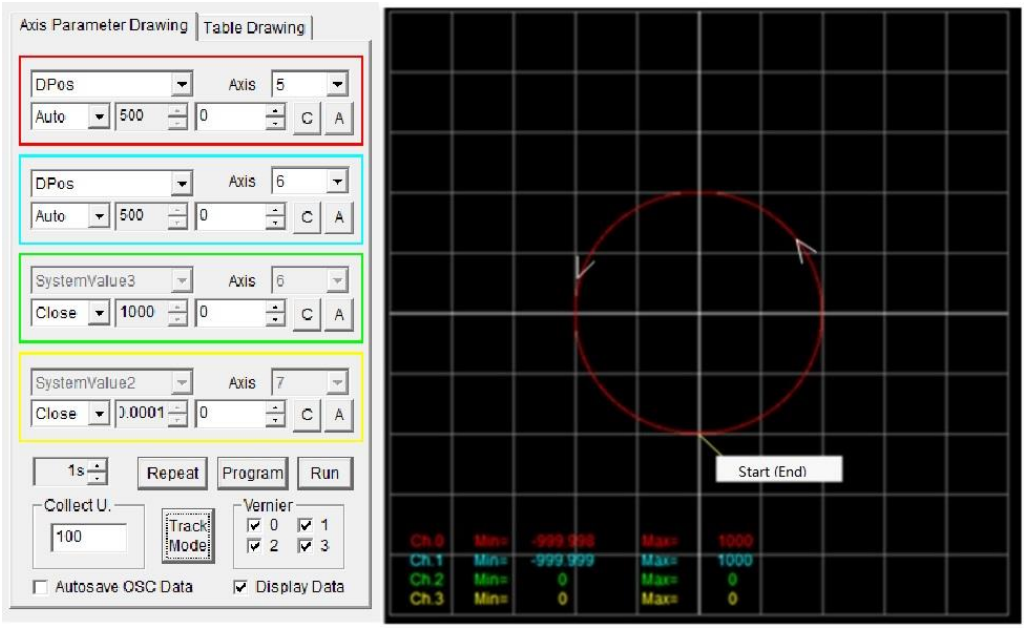
5.5.7.1.4 Example

To illustrate the impact of the parameter settings of the C-axis on the following deviation, some typical situations are described as follows. Assume that the C-axis of the tangential tracking tool rotation axis is axis 7, which follows the tangential angle of the interpolation trajectory of axes 5 and 6. Set axis 7 to cycle mode 2, and the cycle stroke interval to [-10000, 10000).

First, the instruction hmc_movetang(7,5,6,0) is sent, and the master axes 5 and 6 execute the following different instructions.

1. Master axes X and Y performing arc interpolation

Send hmc_movecircle(4,5,6, 1,0,0,0, 1000) to perform arc interpolation with a radius of 1000. Interpolation Speed = 1000, accel = decel = 2000. The arc trajectory is as follows:



XY-Axis Interpolation Trajectory

Some key motion parameters during the execution of arc interpolation can be calculated as follows (in C-axis units):

$$\text{Maximum speed of tangential angle change} = \frac{\text{Speed}}{\frac{\text{Radius}}{2 + \pi}} \times \text{Stroke length} = \frac{1000}{\frac{1000}{2 + \pi}} \times 20000 = 3{,}183.09886$$

$$\text{Acceleration of tangential angle change} = \frac{\text{Acceleration}}{\frac{\text{Radius}}{2 + \pi}} \times \text{Stroke length} = \frac{2000}{\frac{1000}{2 + \pi}} \times 20000 = 6{,}366.19772$$

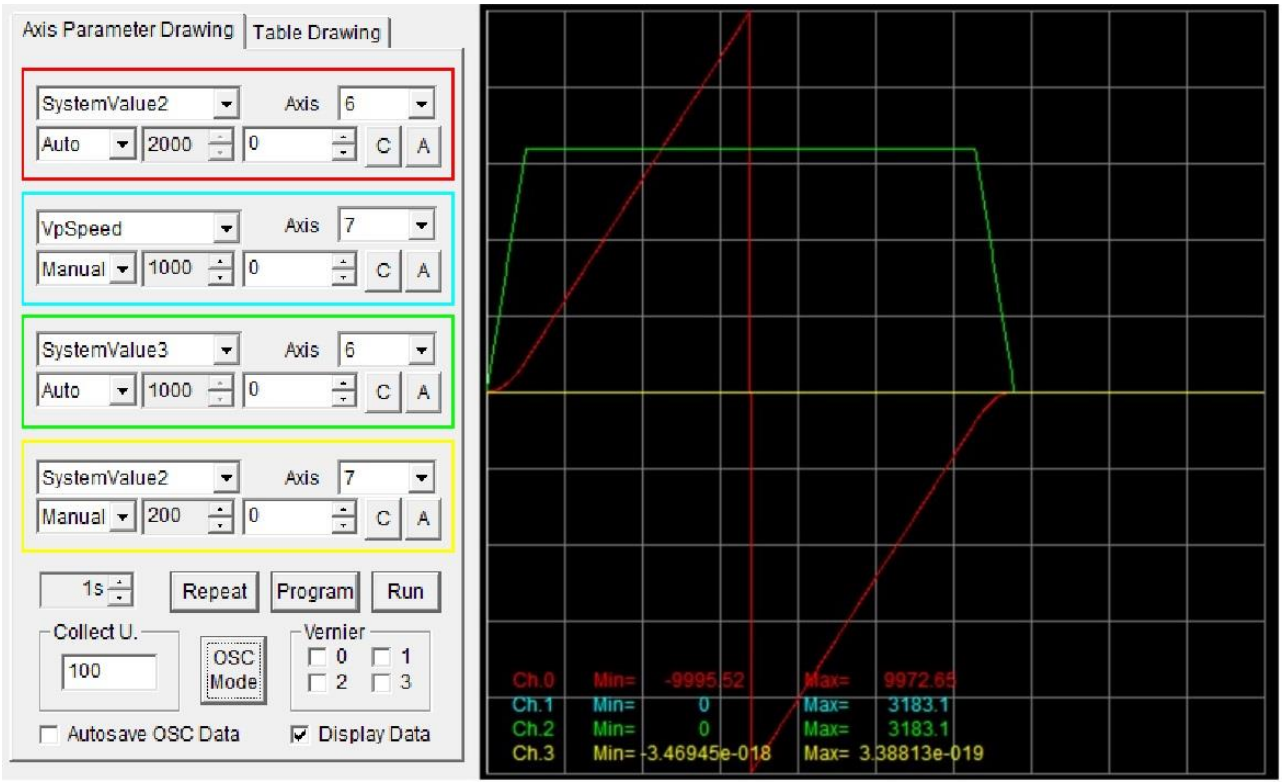
$$\text{Deceleration of tangential angle change} = \frac{\text{Deceleration}}{\frac{\text{Radius}}{2 + \pi}} \times \text{Stroke length} = \frac{2000}{\frac{1000}{2 + \pi}} \times 20000 = 6{,}366.19772$$

If the acceleration, deceleration, and speed parameters of the C-axis are greater than the above parameters calculated, seamless following can be achieved. The motion effects of different C-axis parameters are shown in the figure below, where the red curve is the real-time target angle position during the execution of moveTang, the cyan curve is the C-axis VpSpeed of the tool rotation axis, the green curve is the change speed of the target tangential angle, and the yellow curve is the difference between the instruction solution position and the theoretical target angular position during the execution of moveTang (following deviation).

(1) accel = decel = 10000, speed = 10000

It can be seen from the figure below that the C-axis can achieve seamless following at this time, and the VpSpeed curve of the C-axis completely coincides with the change speed curve of the tangential angle.

The jump in the red curve is due to the conversion at the cycle stroke boundary.



Tangential Tracking Real-time Data of Arc Interpolation (The C-axis Acceleration and Deceleration, and Maximum Speed Are Sufficient)

Red: Target tangential angle

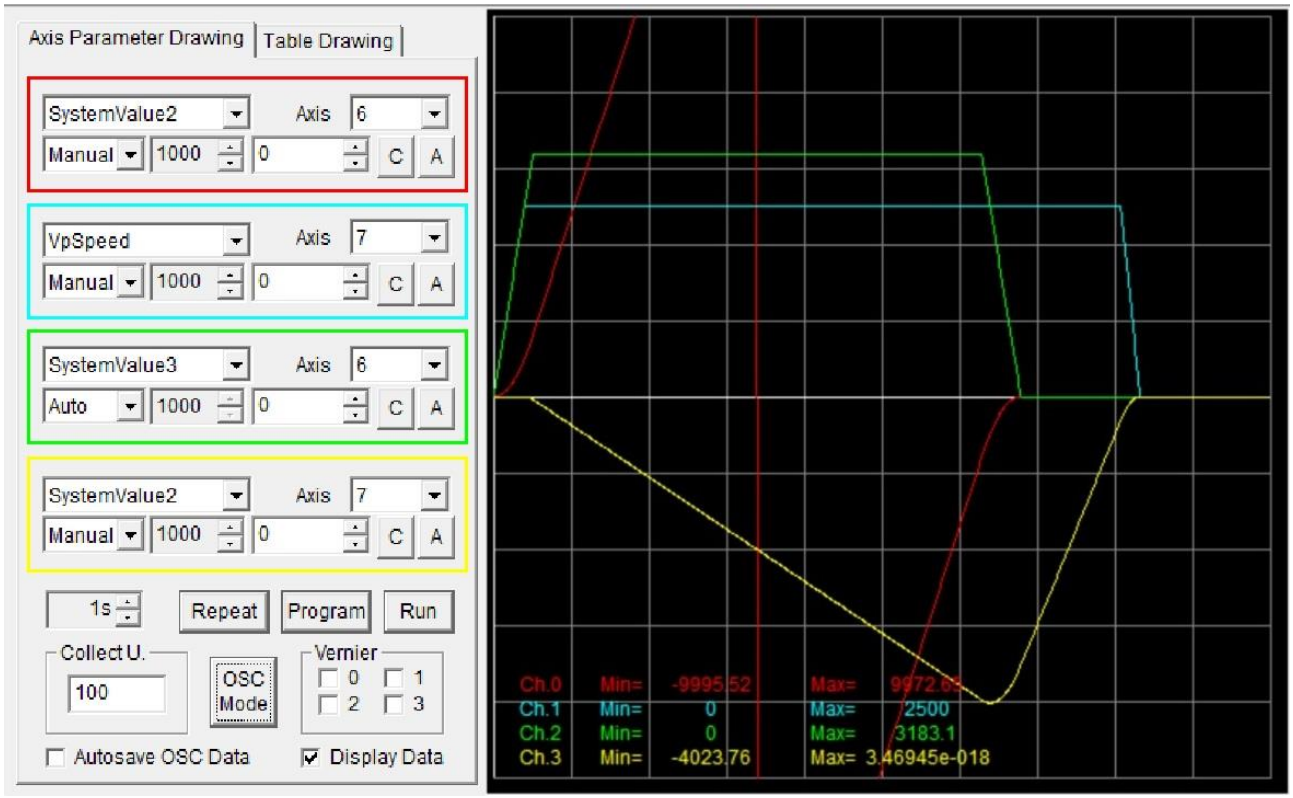
Cyan: Tool rotation axis VpSpeed

Green: Change speed of target tangential angle

Yellow: Difference between the solution position and the target tangential angle

(2) accel = decel = 10000, speed = 2500

At this time, the maximum speed of the C-axis is smaller than the maximum change speed of the tangential angle, and thus seamless following cannot be achieved.



Tangential Tracking Real-time Data of Arc Interpolation (The Maximum Speed of the C-axis is Too Small)

Red: Target tangential angle

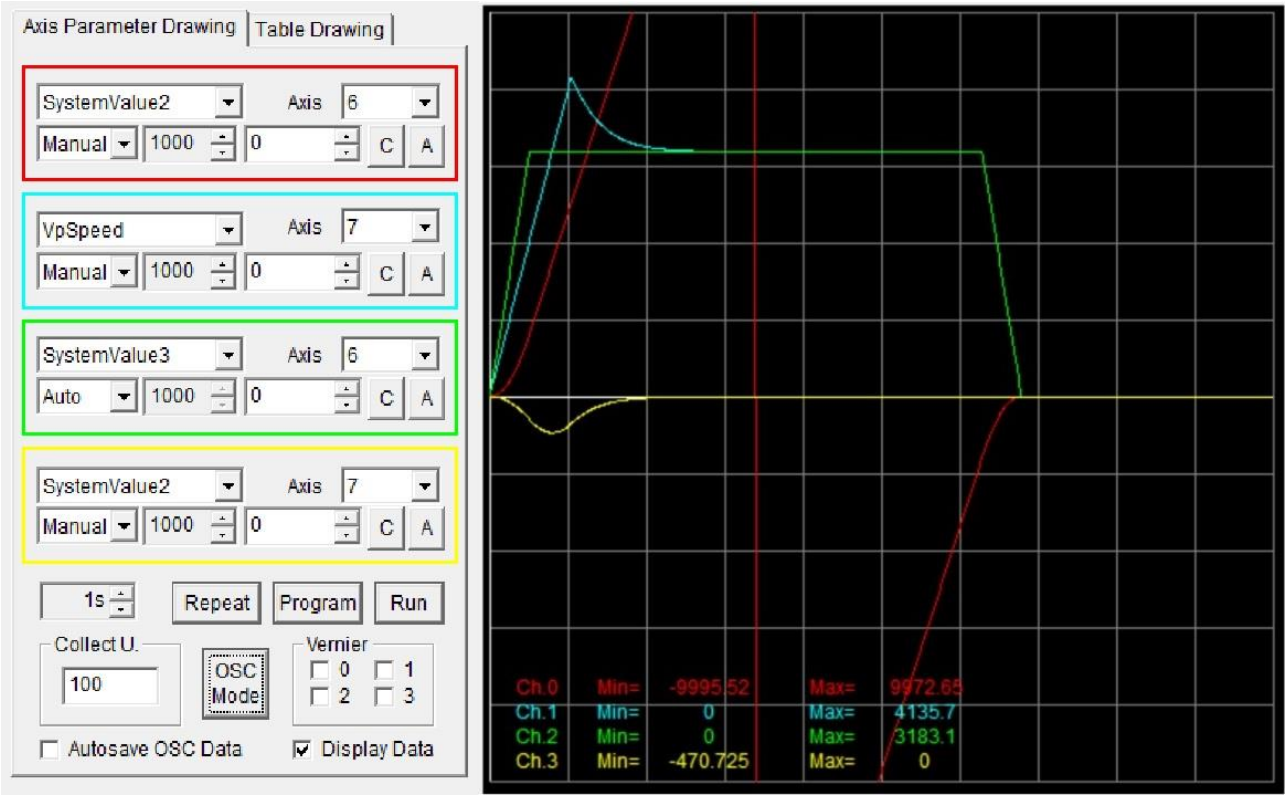
Cyan: Tool rotation axis VpSpeed

Green: Change speed of target tangential angle

Yellow: Difference between the solution position and the target tangential angle

(3) accel = 4000, decel = 10000, speed = 10000

At this time, the maximum acceleration of the C-axis is smaller than the maximum acceleration of the tangential angle change, and thus seamless following cannot be achieved during the acceleration stage.



Tangential Tracking Real-time Data of Arc Interpolation (The Acceleration Capability of the C-axis is Insufficient)

Red: Target tangential angle

Cyan: Tool rotation axis VpSpeed

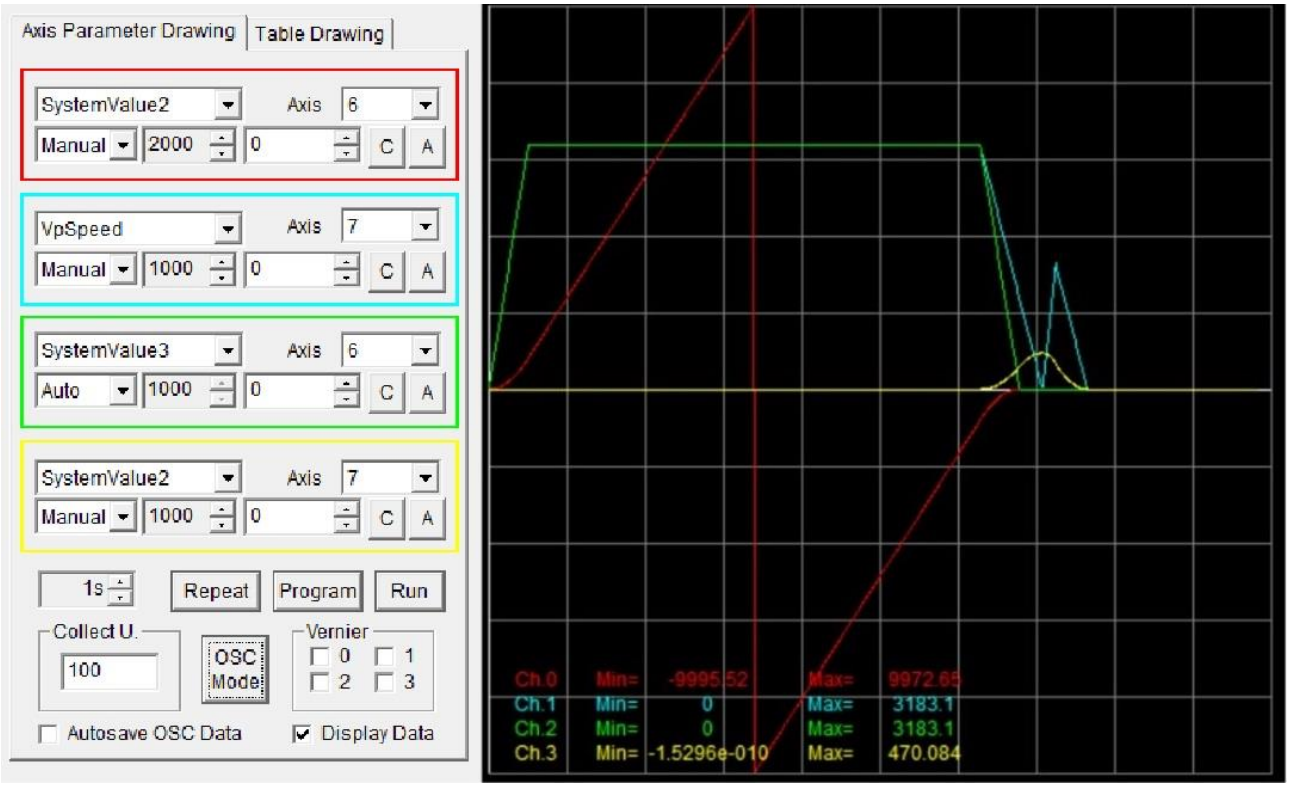
Green: Change speed of target tangential angle

Yellow: Difference between the solution position and the target tangential angle

(4) accel = 10000, decel =4000, speed = 10000

At this time, the maximum deceleration of the C-axis is smaller than the maximum deceleration of the tangential angle change, and thus seamless following cannot be achieved during the deceleration stage.

Therefore, in the final deceleration stage, the C-axis first decelerates to 0 (which has exceeded the target position), and then reverses positions to the target position.



Tangential Tracking Real-time Data of Arc Interpolation (The Deceleration Capability of the C-axis is Insufficient)

Red: Target tangential angle

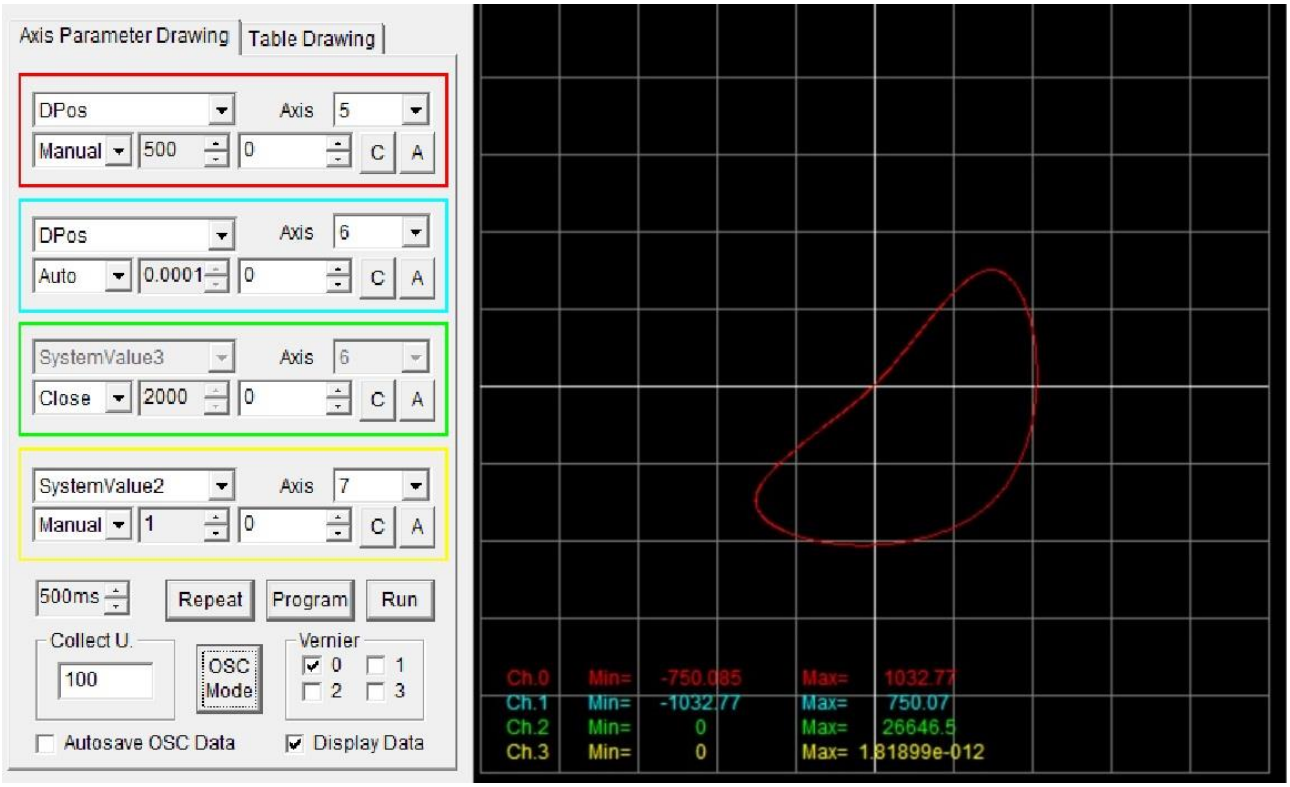
Cyan: Tool rotation axis VpSpeed

Green: Change speed of target tangential angle

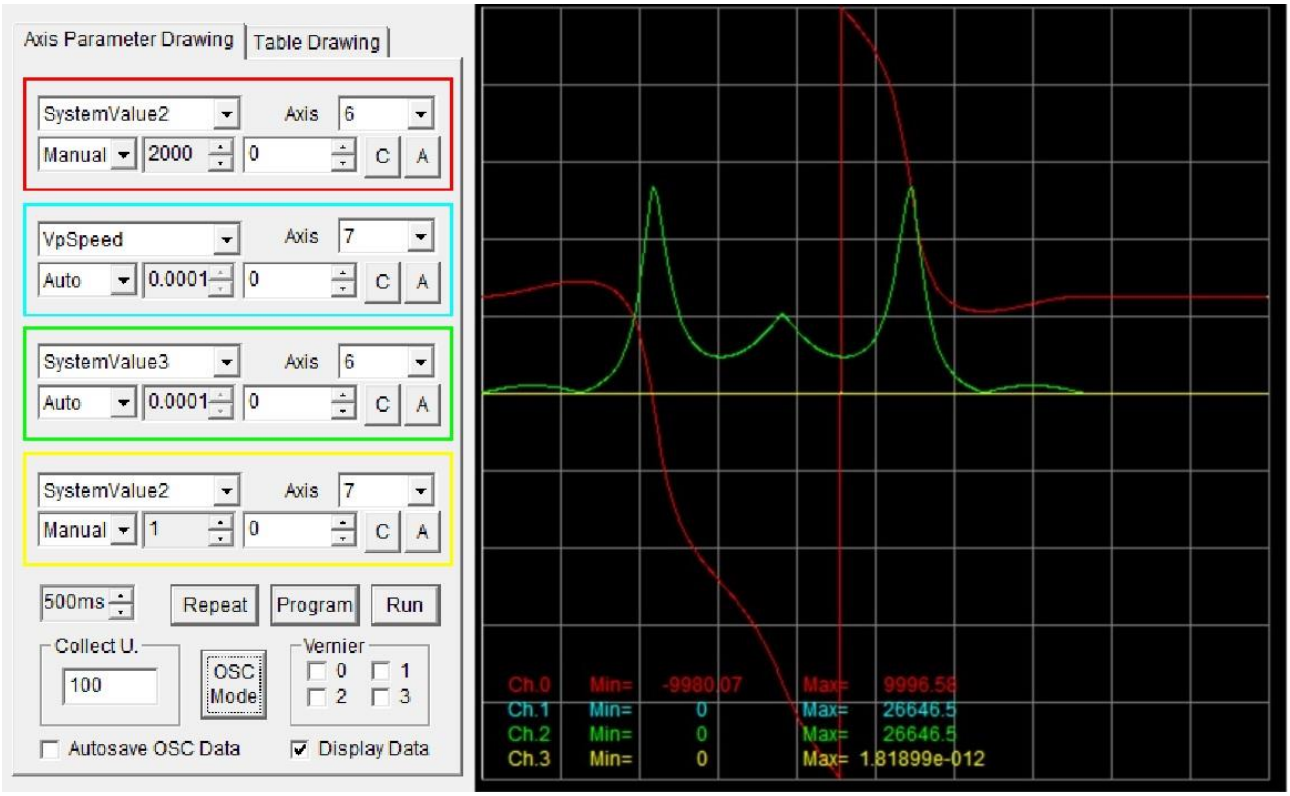
Yellow: Difference between the solution position and the target tangential angle

2. Master axes X and Y performing spline interpolation

The tangential angle change speed and acceleration and deceleration during the spline interpolation process are all variable values, and the maximum acceleration/deceleration and maximum speed cannot be accurately calculated. As long as the acceleration, deceleration, and speed parameters of the C-axis are greater than the tangential angle change speed and acceleration and deceleration during the spline interpolation process, seamless following can be achieved. as shown in the figure:



XY-Axis 2D Spline Interpolation Trajectory



Tangential Tracking Real-time Data of Spline Interpolation

Red: Target tangential angle

Cyan: Tool rotation axis VpSpeed

Green: Change speed of target tangential angle

Yellow: Difference between the solution position and the target tangential angle

3. Treatment of curve connections

At the connection of the curves, if there is a step change in the target tangential angle or the target tangential angle changes speed, then the C-axis will move toward the target tangential angle according to its set acceleration, deceleration, and speed, so there will be no motion shock. However, there will be a certain deviation between the actual position of the C-axis and the target tangential angle position, and this deviation is inevitable.

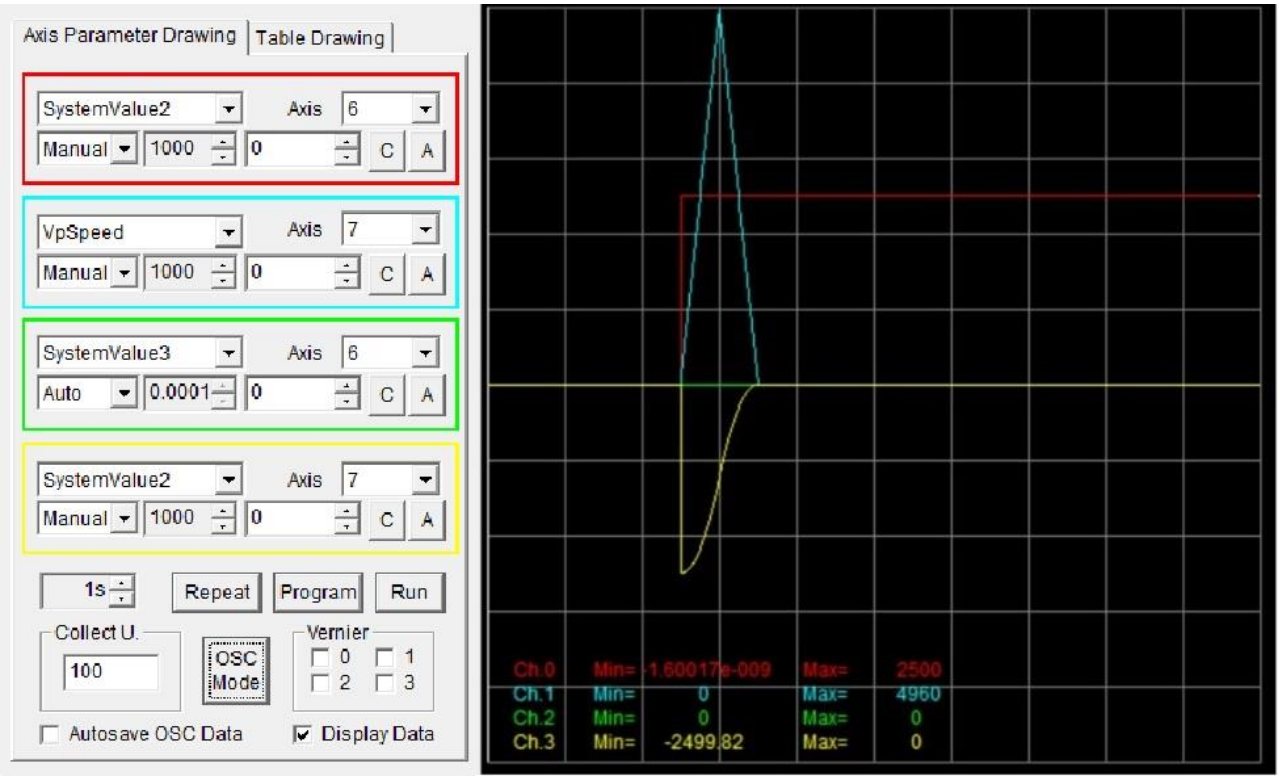
Assume that the C-axis of the tangential tracking tool rotation axis is axis 7, which follows the tangential angle of the combined displacement of axes 5 and 6. Set axis 7 to cycle mode 2, and the cycle stroke interval to [-10000, 10000). First, send the instruction hmc_movetang(7,5,6,0).

Here are just examples to illustrate the connection between straight lines. Due to the step change in the target angular position at the polyline, the C-axis will run towards the target position and target speed with its acceleration and deceleration, and the MoveTang tool rotation axis cannot perform seamless following.

Send instructions: hmc_move2d(4,5,6,2000,000); hmc_move2d(4,5,6,2000,2000). The trajectory is as follows:

XY-Axis Operation Trajectory

C-axis parameters: accel = decel = 10000, speed = 10000 The operation trajectory is as follows:



Real-time Data of Tangential Tracking (Polyline, Merge On)

Red: Target tangential angle

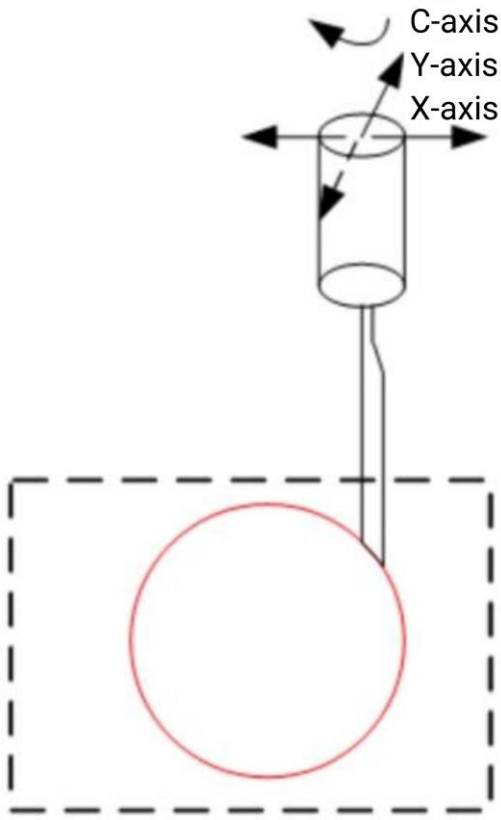
Cyan: Tool rotation axis VpSpeed

Yellow: Difference between the solution position and the target tangential angle

4. Examples of cutting application scenarios

When a cutter is used to cut a planar arc, to ensure the cutting effect and reduce the wear of the cutter, the cutter blade shall be adjusted in real-time during the entire cutting process to follow the tangential direction change of the arc trajectory.

The X-axis and Y-axis drive the cutter to perform arc interpolation motion in the XY plane to cut the material. The C-axis drives the cutter to rotate and then control the direction of the cutter to follow the cutting trajectory and adjust the angle in real time as follows:



Cutting Diagram

Then the program written in AT is as follows:

XAxisID :=1; (*Arc interpolating split axis X*)

YAxisID :=2; (*Arc interpolating split axis Y*)

CAxisID :=3; (*Cutter rotation axis C*)

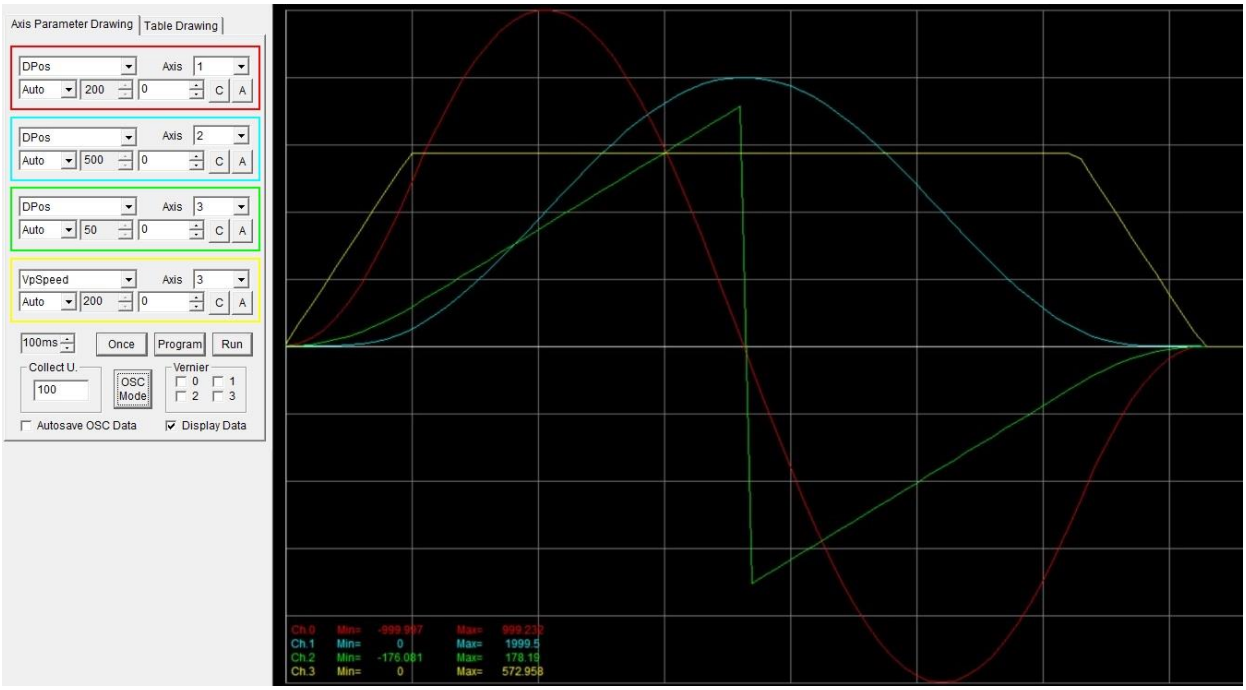
Axis3.RepMode:=2; (*Set the cutter rotation axis to the cycle stroke*)

Axis3. RepDist := 180; (*The cycle stroke range is -180~180, then the C-axis position is displayed in rotation angle*)

HMC_MoveCircle(0, XAxisID, YAxisID, 1,0,0, 0,1000); (*X-axis and Y-axis execute arc interpolation cutting*)

HMC_MoveTang (CAxisID, XAxisID, YAxisID, 0); (*C-axis rotates following the trajectories of the X-axis and Y-axis by adjusting the cutter angle*)

The actual execution result is as follows:



Motion Diagram

It can be seen that when the arc interpolation path is executed to 1/4 of the arc, the cutter angle is 90 degrees. When the arc interpolation continues to be executed to 1/2 of the arc, the cutter angle is 180 degrees. When the arc interpolation continues to be executed to 3/4 of the arc, the cutter angle is -90 degrees. When the arc interpolation ends the full circle, the cutter angle is 0 degrees. During the entire interpolation process, the cutter rotation angle follows the angle between the combined motion trajectory and the positive X-axis in real time, and the tool direction is consistent with the tangential direction of the cutting trajectory (cutting speed vector direction) to achieve the best cutting effect.

5.5.7.2 HMC_MoveTangConfigCornerMode (Tangential tracking corner retract settings)

5.5.7.2.1 Function

When there is a jump change in angle between the two instructions (such as a polyline), the corner retract function can be executed to protect the tool and ensure the processing effect.

The following three retract modes are provided for users to choose according to the degree of automation from high to low: automatic retract mode, manual retract mode, and no-retract mode.

5.5.7.2.1.1 Automatic retract mode ucCornerMode=2

1. Mode introduction

It is judged in advance that if the turning angle at the connection between instructions is greater than the preset retract angle threshold, the following retract action process will be automatically executed:

Wait for the interpolation master axes X and Y and tool lifting Z-axis to stop —> raise the Z-axis to a safe height —> the C-axis rotates to the starting tangential angle of the next instruction —> the Z-axis drops to its original position —> axes X, Y and Z resume their operation

It supports connections between any interpolation instructions. Since the starting tangential angle of non-interpolation instructions cannot be predicted, if there are non-interpolation instructions in the XY instruction queue, the connection between the instructions will be processed as if the tool is not retracted.

The following two scenarios are supported:

The master axes X and Y are the two split axes of the same interpolation instruction. A typical application is that the axes X and Y execute two-dimensional plane interpolation instructions.

The master axes X and Y and the tool lifting Z-axis are the three split axes of the same interpolation instruction. A typical application is that the X-axis, Y-axis, and Z-axis execute three-dimensional space interpolation instructions.

To ensure that the X-axis and Y-axis stop reliably at the connection of the corner retraction instruction when merge is enabled, it is necessary to ensure:

merge stop angle ≤ corner retract angle threshold

2. Configuration process

(1) Use HMC_MoveTang (ucAxisC, ucAxisX, ucAxisY, dOffset) to send tangential tracking instructions.

(2) Use HMC_MoveTangConfigCornerMode(ucAxisC,ucCornerMode,dCornerAngleLimit, ucAxisZ,dSafePosZ,ucSafePosZMode) to configure the corner retract mode as automatic retract mode, and also configure the corner angle threshold, tool lifting Z-axis number, Z-axis retract height, Z-xis retract height coordinate mode (incremental height/absolute height).

(3) Send two-dimensional plane interpolation instructions to the master axes X and Y, or send three-dimensional space interpolation instructions to the axes X, Y, and Z.

3. Additional information

(1) During the operation of the HMC_MoveTang instruction, HMC_GetMoveTangCornerState(ucAxisC) can be called to obtain the real-time operating status (normal operating status, corner retract status).

(2) If HMC_Cancel is used to cancel the HMC_MoveTang instruction in the corner retract status, the instruction will simultaneously control the C-axis and Z-axis to stop according to their own deceleration.

(3) Since the instructions in the XYZ instruction queue are in a suspended status when the tool is retracted at the corner, for safety reasons, the instructions in the XYZ instruction queue will not be automatically resumed after the cancellation is completed. To resume the subsequent instructions in the XYZ instruction queue for operation, the user needs to call HMC_SetCmdBufferLoad Mode(ucAxisID, ucCmdLoadMode) to set the instruction loading mode to normal loading mode (0);

(4) If the Z-axis encounters a limit in the corner retract status, the Z-axis will decelerate to stop at MAX{decle, FastDecel}. For safety reasons, HMC_MoveTang automatically switches to the manual retract mode.

5.5.7.2.1.2 Manual retract mode ucCornerMode=1

1. Mode introduction

When the turning angle at the connection between instructions is greater than the preset corner retract angle threshold, the X-axis and Y-axis automatically stop to wait for the user to process the Z-axis retract and C-axis direction adjustment actions in the IEC program. After the retract action process is completed, the X-axis and Y-axis continue to execute subsequent instructions. The processing flow of a complete corner retract in manual mode is as follows (the yellow part is the user IEC program):

LX controller:

If it is detected that the corner retract conditions are met, the interpolation master axes X and Y stop to wait.

In IEC periodic tasks (user-written program):

(1) Call HMC_GetMoveTangCornerState to query that MoveTang is in the corner retract waiting status in manual retract mode;

(2) Use HMC_Move to raise the Z-axis to a safe height;

(3) Wait for the Z-axis lifting action to be completed;

(4) Call HMC_MoveTangCornerFollowNextCmd to rotate the C-axis to the starting tangential angle of the next instruction;

(5) Wait for the C-axis rotation to complete;

(6) Use HMC_Move to drive the Z-axis down to its original position;

(7) Wait for the Z-axis dropping action to be completed;

(8) Call HMC_MoveTangCornerContinue to resume the X-axis and Y-axis to run subsequent instructions.

It supports connections between any interpolation instructions. Since the starting tangential angle of non-interpolation instructions cannot be predicted, if there are non-interpolation instructions in the XY instruction queue, the connection between the instructions will be processed as if the tool is not retracted.

Situation supported: The master axes X and Y are the two split axes of the same interpolation instruction. A typical application is that the axes X and Y execute two-dimensional plane interpolation instructions.

To ensure that the X-axis and Y-axis stop reliably at the connection of the corner retraction instruction when merge is enabled, it is necessary to ensure:

merge stop angle ≤ corner retract angle threshold

2. Configuration process

(1) Use HMC_MoveTang (ucAxisC, ucAxisX, ucAxisY, dOffset) to send tangential tracking instructions.

(2) Use HMC_MoveTangConfigCornerMode(ucAxisC,ucCornerMode,dCornerAngleLimit, ucAxisZ,dSafePosZ,ucSafePosZMode) to configure the corner retract mode to manual retract mode, and configure the corner angle threshold;

(3) In a periodic IEC task, call HMC_GetMoveTangCornerState(ucAxisC) to repeatedly query the tangential tracking execution status. If the return value is the corner stop status (2) in manual retract mode, execute the

following IEC program:

(4) Use HMC_Move(ucAxisZ,dSafePos) to drive the Z-axis to retract the tool to the safety plane;

(5) Wait for the Z-axis to reach the safe plane, and then call HMC_MoveTangCornerFollowNextCmd(AxisC), and the C-axis will move to the starting tangential angle position of the next instruction;

(6) Wait for the C-axis to reach the starting tangential angle position of the next instruction, and then use HMC_Move(ucAxisZ, -dSafePos) to drive the Z-axis to drop the tool to the original position;

(7) Wait for the Z-axis to reach the original position, and call HMC_MoveTangCornerContinue (AxisC) to resume the operation of the tangential tracking master axis;

(8) Send interpolation instructions to the master axes X and Y.

3. Additional information

(1) Different from the automatic retract mode, the manual retract mode does not support the situation where the master axes X and Y and the tool lifting Z-axis are used as three split axes of the same interpolation instruction;

(2) If HMC_Cancel is used to cancel the HMC_MoveTang instruction in the corner stop status, because the instructions in the XY instruction queue are in a suspended status in the corner retract status, for safety reasons, the instruction in the XY axis instruction queue will not be automatically resumed for execution after the cancellation is completed. To resume the execution of subsequent instructions in the XY instruction queue, the user needs to call HMC_SetCmdBufferLoad Mode (ucAxisID, ucCmdLoad Mode) to set the instruction loading mode to the normal loading mode (0).

5.5.7.2.1.3 No-retract mode ucCornerMode=0

In this mode, no processing will be performed, and the interpolation axes X and Y at the corner will not stop and wait.

5.5.7.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
ucCornerMode	Input	USINT	Corner retract mode (0: No retract; 1: Manual corner retract; 2: Automatic corner retract)
dCornerAngleLimit	Input	LREAL	Corner retract angle threshold (in radians)
ucAxisZ	Input	USINT	Corner automatic retract Z-axis number Note: It takes effect when ucCornerMode is 2.
dSafePosZ	Input	LREAL	Safe tool retract height for automatic corner retract. Note: It takes effect when ucCornerMode is 2.

ucSafePosZmode	Input	USINT	Safe tool retract height coordinate mode for automatic corner retract (0: incremental coordinates, 1: absolute coordinates). Note: It takes effect when ucCornerMode is 2.
HMC_MoveTangConfigConrnerMode	Output	USINT	0: Set successfully Others: Error code

5.5.7.2.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.2.4 Additional information

If the corner retract action cannot meet the user's requirements, the user can add a process path at the corner to meet the processing requirements.

(1) Arc process path

When the process allows, an arc auxiliary process path can be added at the corner to improve processing efficiency and avoid frequent retract, and meanwhile to improve cutting quality at corners. The black polyline is the processing path, and the purple part is the process path.

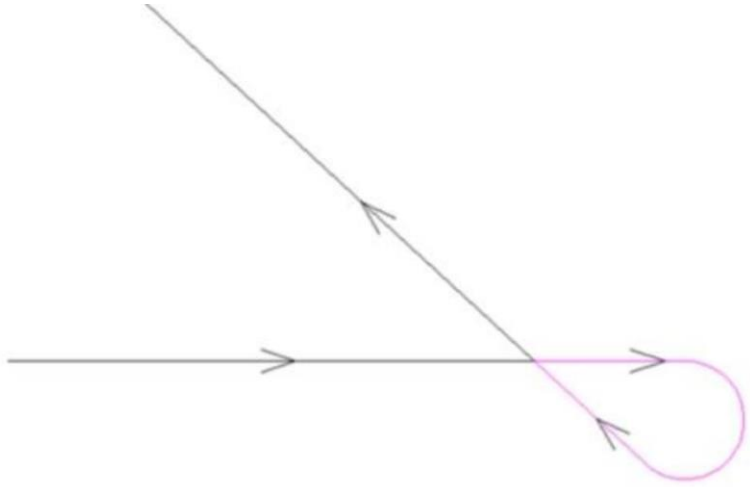


Diagram of Arc Process Path

(2) Process path with retract

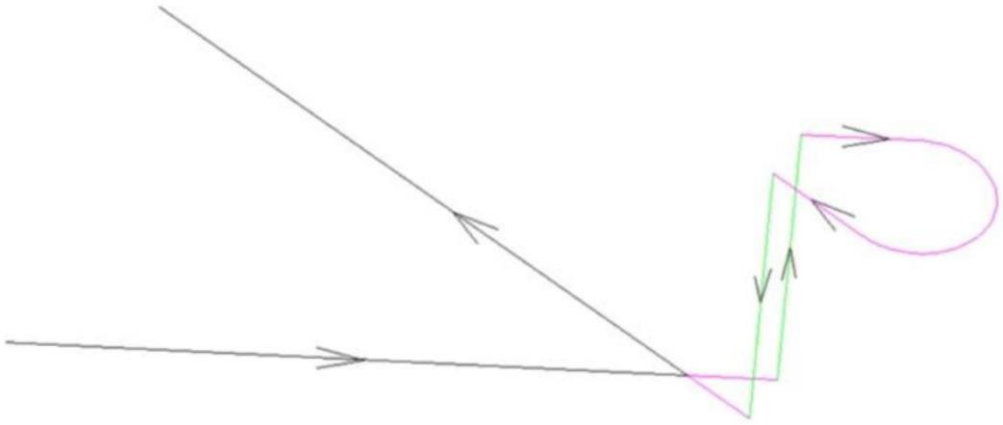


Diagram of Process Path With Retract

If the user wants to extend the cutter's processing path at the corner to ensure the cutting quality at the corner, and at the same time wants to reduce the waste in cutting raw materials caused by the introduction of the process path as much as possible, the user can insert a Z-direction retract path in the process path as shown in (1), such as the green part.

5.5.7.2.5 Example

(1) Example 1: Automatic retract

The interpolation instruction trajectory is as follows. To protect the device during the cutting process, the tool shall be retracted when the instruction corner is greater than or equal to π/4. Now let's take automatic retract and manual retract as examples to realize the retract protection function.

Send the HMC_MoveTang instruction to control the device to automatically retract the tool at a corner greater than or equal to π/4. Set the corner retract angle threshold to π/4-10-6. 10-6 is a safety margin, because there may be a small amount of deviation in the number of floating points. The user can choose the margin size according to the accuracy requirements. The ST language program is as follows. The interpolation trajectory of the master axes X and Y is shown in the figure below. The interpolation trajectory of the tangential tracking master axes X and Y is shown in the figure below. The motion process curves of each stage are shown in the second figure 2 below: Motion Process Curve of Tangential Tracking in Automatic Retract Mode.

AxisXY_ID:=4; (*Interpolation combined axis number*)

AxisX_ID:=5; (*X-axis number*)

AxisY_ID:=6; (*Y-axis number*)

AxisZ_ID:=7; (*Z-axis number*)

AxisC_ID:=8; (*C-axis number*)

tangentAngle0ffset:=0; (*Tangential angle offset*)

cornerMode :=2; (*Automatic corner retract mode*)

cornerAngleLimit := ATAN (1) - 1E-6; (*The corner retract angle threshold is 45 degrees, and 1E-6 is a safety

margin*)

safePosZ:= 4000; (*Safe height of corner retract*)

SafePosZMode:= 0; (*The safe height of corner retract is incremental coordinates*)

HMC_MoveTang (AxisC_ID, AxisX_ID, AxisY_ID, tangentAng1e0ffset); (*Send tangential tracking instructions*)

HMC_MoveTangConfigCornerMode (AxisC_ID, cornerMode, cornerAngleLimit, AxisZ_ID, safePosZ, SafePosZMode); (*Configure corner retract mode*)

Axis4.Merge := 1; (*Enable merge*)

HMC_SetMergeAngle (AxisXY_ID, ASIN (0.5), cornerAngleLimit); (*Configure merge deceleration angle and stop angle*)

(**Send interpolation instructions to master axes (arc+Move2d)**)

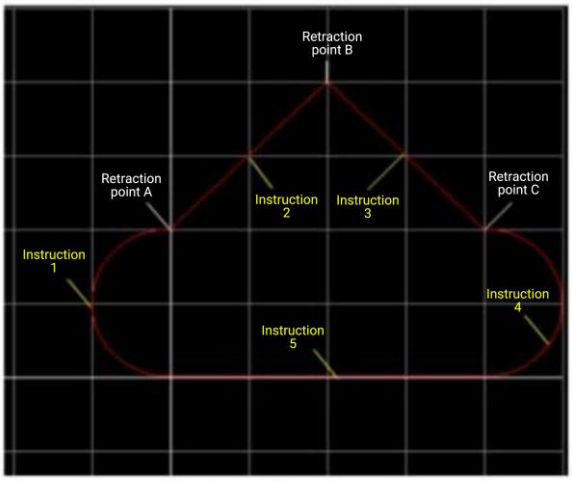
HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID, 3, 0, 2000, 0, 1000);

HMC_Move2d(AxisXY_ID, AxisX_ID, AxisY_ID, 2000, 2000);

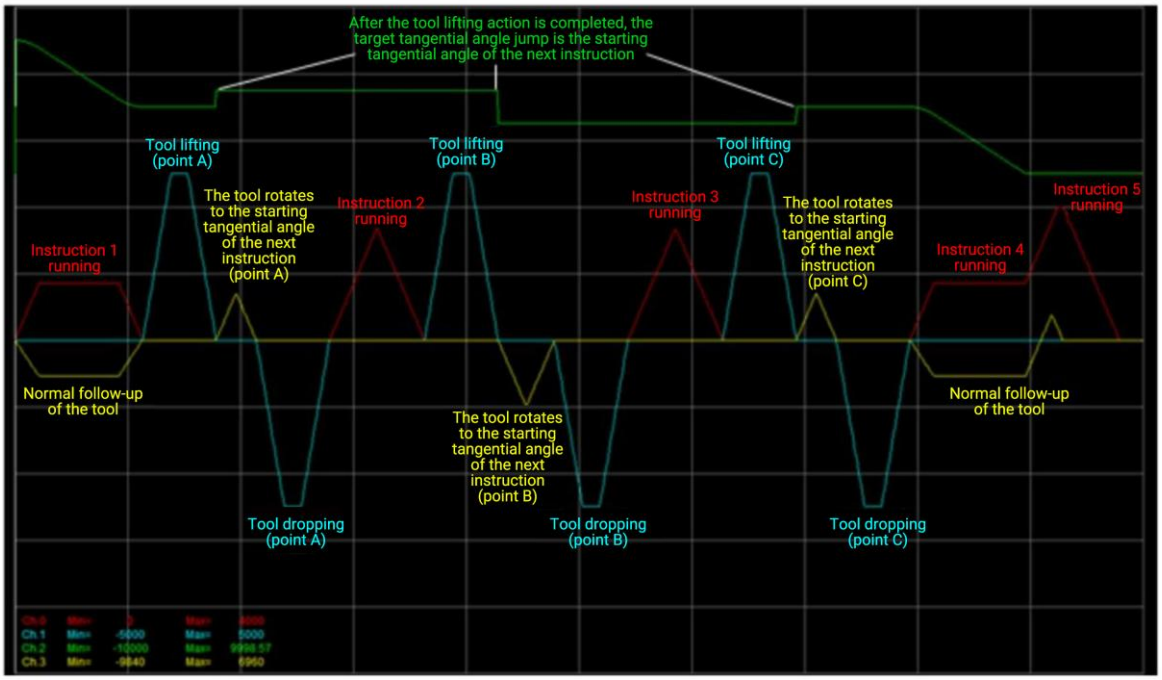
HMC_Move2d(AxisXY_ID, AxisX_ID, AxisY_ID, 2000, -2000);

HMC_MoveCircle (AxisXY_ID, AxisX_ID, AxisY_ID, 3, 0, -2000, 0, -1000);

HMC_Move2d(AxisXY_ID, AxisX_ID, AxisY_ID, -4000, 0);



Interpolation Trajectory of the Tangential Tracking Master Axes X and Y



Motion Process Curve of Tangential Tracking in Automatic Retract Mode (Master Axes X and Y Execute Arc +Move2d)

Green: Target tangential angle

Red: MpSpeed curve of XY interpolation combined axis

Yellow: MpSpeed curve of tool rotation C-axis

Cyan: MpSpeed curve of tool lifting Z-axis

As can be seen from the above figure, at the connection of instructions, when the change of the target tangential angle is greater than the set retract threshold (the corners between the first four instructions are all greater than the retract threshold), the interpolation axes X and Y automatically stop (the red curve is the speed curve of the interpolation combined axis), the retract lifting Z-axis begins to lift to the set position (the cyan curve is the speed curve of the tool lifting axis), then the cutter rotation axis rotates to the starting tangential angle of the next instruction, the tool lifting Z-axis drops to its original position, and the next interpolation instruction continues to be executed. When the target tangential angle change at the instructions connection points is smaller than the set retract threshold (the corner between the fourth and fifth instructions is 0 degrees), the retract action is not performed and the interpolation axis motion does not stop.

If the above move2d instruction is replaced by Move3d, that is, the axes X, Y, and Z axis execute the arc +3-axis interpolation instruction, the execution effect is as shown in the figure below. It can be seen from the figure that the interpolation motion with the Z-axis as a split axis for the 3-axis interpolation is performed in a different stage from the tool retract action of the Z-axis, resulting in no conflict.

(**Send interpolation instructions to master axes (arc+Move3d)**)

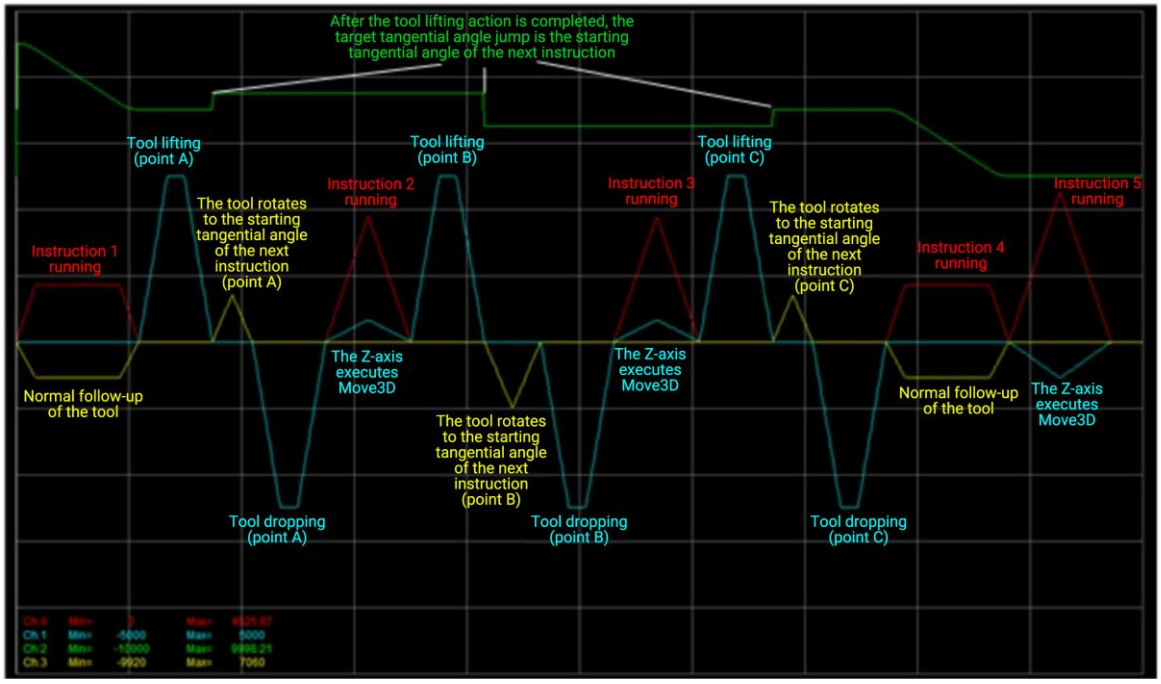
HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID, 3, 0, 2000, 0, 1000);

HMC_Move3d(AxisXY_ID, AxisX_ID, AxisY_ID, AxisZ_ID, 2000, 2000, 500);

HMC_Move3d(AxisXY_ID, AxisX_ID, AxisY_ID, AxisZ_ID, 2000, -2000, 500);

HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID, 3, 0, -2000, 0, -1000);

HMC_Move3d(AxisXY_ID, AxisX_ID, AxisY_ID, AxisZ_ID, -4000, 0, -1000);



Motion Process Curve of Tangential Tracking in Automatic Retract Mode (Axes X, Y, and Z Executing Arc +Move3d)

Green: Target tangential angle

Red: MpSpeed curve of XYZ interpolation combined axis

Yellow: MpSpeed curve of tool rotation C-axis

Cyan: MpSpeed curve of tool lifting Z-axis

(2) Example 2: Manual retract

The XY interpolation path is the same as (1). Send the HMC_MoveTang instruction to the C-axis and set it to the manual retract mode, with other parameters the same as those in (1). The ST program is as follows. The interpolation trajectory of master axes X and Y is shown in Figure 1 below, and the motion process curves of each stage are shown in Figure 2 below. It can be seen from the figure that the results are the same as those in the automatic retract mode in (1).

Task 1 program:

AxisXY_ID:=4; (*Interpolation combined axis number*)

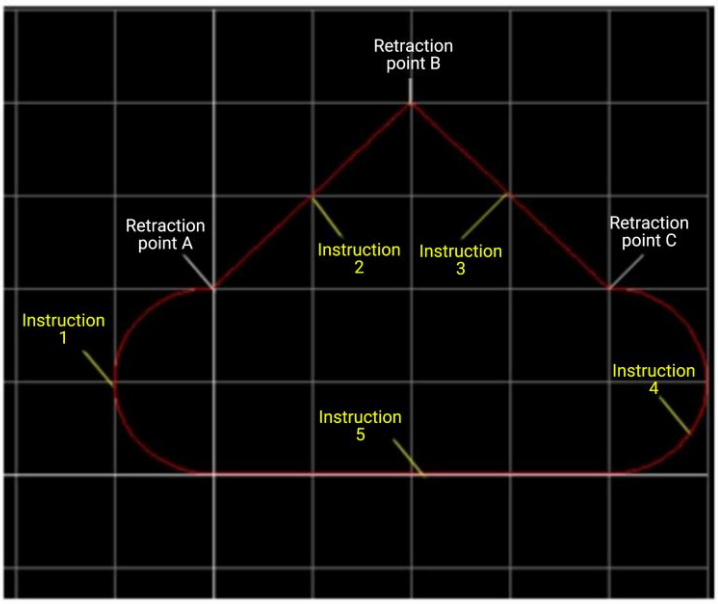
AxisX_ID:=5; (*X-axis number*)

AxisY_ID:=6; (*Y-axis number*)

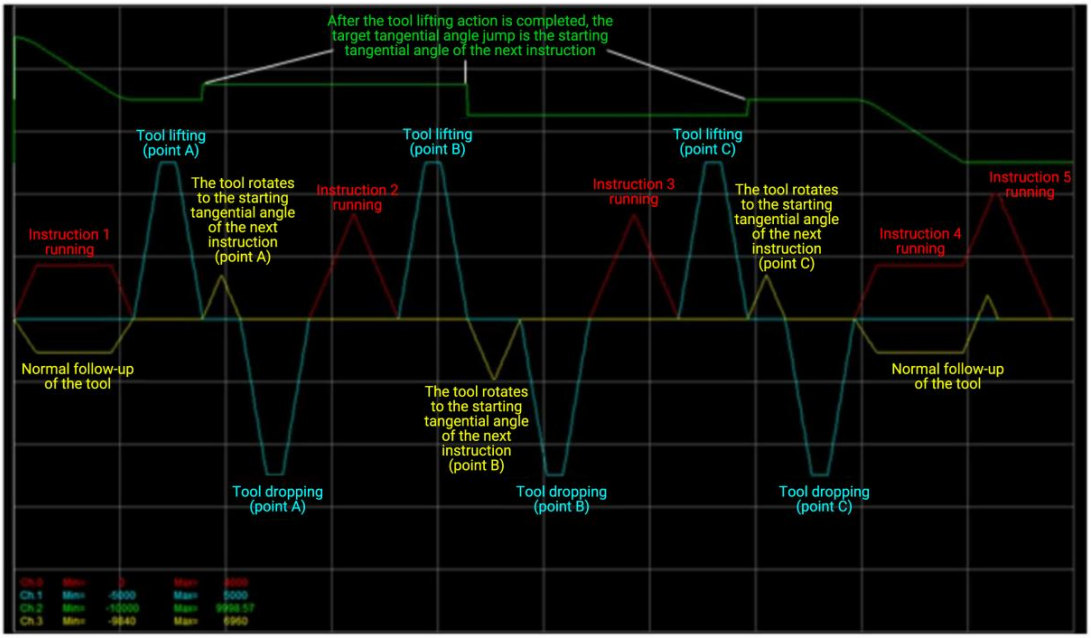
```
AxisZ_ID:=7; (*Z-axis number*)

AxisC_ID:=8; (*C-axis number*)

tangentAngle0ffset:=0; (*Tangential angle offset*)

cornerMode :=1; (*Manual corner retract mode*)

cornerAngleLimit := ATAN (1) - 1E-6; (*The corner retract angle threshold is 45 degrees, and 1E-6 is a safety margin*)

HMC_MoveTang (AxisC_ID, AxisX_ID, AxisY_ID, tangentAngle0ffset); (*Send tangential tracking instructions*)

HMC_MoveTangConfigCornerMode (AxisC_ID, cornerMode, cornerAngleLimit, 0, 0, 0); (*Configure corner retract mode*)

(**Send interpolation instructions to master axes**)

HMC_movecircle (AxisXY_ID, AxisX_ID, AxisY_ID, 3, 0, 2000, 0, 1000);

HMC_move2d(AxisXY_ID, AxisX_ID, AxisY_ID, 2000, 2000);

HMC_move2d(AxisXY_ID, AxisX_ID, AxisY_ID, 2000, -2000):

HMC_movecircle(AxisXY_ID, AxisX_ID, AxisY_ID, 3, 0, -2000, 0, -1000);

HMC_move2d(AxisXY_ID, AxisX_ID, AxisY_ID, -4000, 0);

Task 2 program (periodic task):

ManualModeState_Pause:=2; (*Corner retract waiting status*)

safePosZ:= 4000; (*Safe height of corner retract*)

IF (HMC_GetMoveTangCornerState (AxisC_ID) = ManualModeState_Pause ) THEN (*Master axes X and Y are in the corner stop status, waiting for tool retracting*)

CmdID_AxisZ := HMC_Move(AxisZ_ID, safePosZ); (*Z-axis tool retracting*)

HMC_WaitCmdFinish(CmdID_AxisZ); (*Wait for the Z-axis to complete tool retracting*)

HMC_MoveTangCornerFollowNextCmd(AxisC_ID); (*The C-axis moves to the starting tangential angle direction of the next instruction*)

WHILE (HMC_GetMoveTangDestDirection(AxisC_ID)<>Axis8.dPos) DO (*Wait for the C-axis action to complete*)

:

END_WHILE

CmdID_AxisZ := HMC_Move(AxisZ_ID, safePosZ); (*Z-axis tool dropping*)

HMC_WaitCmdFinish(CmdID_AxisZ); (*Z-axis tool dropping completed*)
```

HMC_MoveTangCornerContinue(AxisC_ID); (*The corner retract actions are completed and the interpolation axis resumes operation*)

END_IF



Interpolation Trajectory of the Tangential Tracking Master Axes X and Y



Motion Process Curve of Tangential Tracking in Manual Retract Mode (Master Axes X and Y Executing Arc +Move2d)

Green: Target tangential angle

Red: MpSpeed curve of XY interpolation combined axis

Yellow: MpSpeed curve of tool rotation C-axis

Cyan: MpSpeed curve of tool lifting Z-axis

5.5.7.3 HMC_GetMoveTangCornerState (Getting Tangential Tracking Corner Retracting Status)

5.5.7.3.1 Function

Get the tangential tracking corner retract status.

5.5.7.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_GetMoveTangCornerState	Output	USINT	0: No corner retract; 1: Manually retract the tool at the corner, and the master axes X and Y are in normal operation; 2: Manually retract the tool at the corner, and the master axes X and Y encounter the corner and pause to wait for the engineering program to process the tool retract; 3: Automatically retract the tool at the corner, and the master axes X and Y are in normal operation; 4: Automatically retract the tool at the corner, and the master axes X and Y encounter the corner and pause for automatic tool retract processing. 255: Error

5.5.7.3.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.3.4 Additional information

When the current instruction is not a tangential tracking instruction, an error will be returned.

5.5.7.4 HMC_MoveTangCornerFollowNextCmd (C-axis Positioning to the Starting Tangential Angle of the Next Instruction)

5.5.7.4.1 Function

It is used in the tangential tracking corner manual retract mode. When the axis pauses at the corner to wait for retracting, this instruction can be used to drive the C-axis to position to the starting tangential angle of the next instruction.

5.5.7.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_MoveTangCornerFollowNextCmd	Output	UDINT	0: Set successfully Others: Error code

5.5.7.4.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.4.4 Additional information

If the current instruction is not a tangential tracking instruction in manual retract mode, an error will be returned.

5.5.7.5 HMC_MoveTangCornerContinue (Continuing tangential tracking)

5.5.7.5.1 Function

It is used in the tangential tracking in corner manual retract mode. When it is paused in the corner to wait for the tool to be retracted, this instruction can be used to resume tangential tracking and continue running.

5.5.7.5.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_MoveTangCornerContinue	Output	UDINT	0: Set successfully Others: Error code

5.5.7.5.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.5.4 Additional information

If the current instruction is not a tangential tracking instruction in manual retract mode, an error will be returned.

5.5.7.6 HMC_GetMoveTangDestDirection (Getting Tangential Tracking Target Direction Position)

5.5.7.6.1 Function

This instruction obtains the target tangential angle of tangential tracking in the unit of the C-axis (ucAxisC, tool rotation axis). When this instruction is executed, the current instruction of the C-axis must be the HMC_MoveTang instruction.

At this time, the C-axis is working in cycle stroke mode in mode 2 (symmetrical interval), so the target tangential angle range returned by this instruction is [-Repdist, Repdist), corresponding to the tangential angle of [-π, π).

5.5.7.6.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_GetMoveTangDestdirection	Output	LREAL	Target tangential angle of the tangential tracking instruction. If there is an error in calling this instruction, 0 is returned and a user error is reported.

5.5.7.6.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.7 HMC_GetMoveTangDestVelocity (Getting Target Speed of Tangential Tracking)

5.5.7.7.1 Function

This instruction obtains the change speed of the target tangential angle of tangential tracking in C-axis (ucAxisC, tool rotation axis) user unit/S. When this instruction is executed, the current instruction of the C-axis must be the HMC_MoveTang instruction.

5.5.7.7.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_GetMoveTangDestVelocity	Output	LREAL	Change speed of the target tangential angle of the tangential tracking instruction. When the function fails, it returns 0 and reports a user error.

5.5.7.7.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.8 HMC_GetMoveTangDeviation (Getting Direction Deviation of Tangential Tracking)

5.5.7.8.1 Function

This instruction gets the deviation between the current instruction position of the tangential tracking C-axis (ucAxisC, tool rotation axis) and the target tangential angle in C-axis user unit. When this instruction is executed, the current instruction of the C-axis must be the HMC_MoveTang instruction.

5.5.7.8.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number

HMC_GetMoveTangDestViation	Output	LREAL	Deviation between the current position of the tangential tracking instruction and the target tangential angle. When the function fails, it returns 0 and reports a user error.

5.5.7.8.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.9 HMC_GetMoveTangMasterAxisX (Getting Tangential Tracking Master Axis X)

5.5.7.9.1 Function

Get the X-axis of the current tangential tracking instruction.

5.5.7.9.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_GetMoveTangMasterAxisX	Output	INT	X-axis number: Success -1: error

5.5.7.9.3 Instruction type

It is a non-axis motion cache instruction.

5.5.7.10 HMC_GetMoveTangMasterAxisY (Getting tangential tracking master axis Y)

5.5.7.10.1 Function

Get the Y-axis of the current tangential tracking instruction.

5.5.7.10.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisC	Input	USINT	Tool rotation axis number
HMC_GetMoveTangMasterAxisY	Output	INT	Y-axis number: Success -1: error

5.5.7.10.3 Instruction type

It is a non-axis motion cache instruction.

5.5.8 Multi-axis Synchronization Control

5.5.8.1 HMC_GantrySynchro (Gantry Synchronization Control)

5.5.8.1.1 Function

A gantry group supports up to 8 gantry axes, including a gantry master axis, with the remaining axes serving as slave axes, that is, it supports up to 7 slave axes. Gantry synchronization instructions are not displayed in the master axis motion cache. When a motion control instruction is sent to the master axis and a gantry synchronous control instruction is sent to the slave axis, with the master axis specified, the gantry synchronous control instruction dynamically synchronizes the master axis and the slave axis.

The gantry synchronization control instruction realizes dynamic synchronization control of a group of gantry axes through a combination of open-loop and closed-loop methods as follows:

Open-loop synchronization: synchronize the target position calculated by the gantry master axis to each gantry slave axis as the target position of the slave axis.It ensures position synchronization between axes by using front-end instructions;

Closed-loop synchronization: dynamically monitor the actual feedback position value of each gantry axis, dynamically compensate each gantry through the inter-axis cross-coupling algorithm, realize dynamic correction, and eliminate the deviation between gantry axes. For closed-loop synchronization, each gantry axis must have a position feedback value.

During the dynamic synchronization process, it also supports real-time deviation monitoring between axes, with two sets of thresholds provided:

Inter-axis deviation alarm threshold: It is used to monitor in real time the deviation between two axes in the gantry group. When the deviation exceeds the alarm threshold, bit5 of the corresponding axis parameter AxisStatus is set to 1. When the deviation recovers, bit5 of the axis parameter AxisStatus is automatically restored.

Inter-axis deviation stop protection threshold: It is used to monitor in real time the deviation between two axes in the gantry group. When the deviation exceeds the alarm threshold, bit6 of the corresponding axis parameter AxisStatus is set to 1 and the servo enable switch is turned off, regardless of whether bit6 of AxisErrMask is 1. When the deviation recovers, bit6 of the axis parameter AxisStatus does not recover automatically and needs to be cleared manually.

The HMC_Cancel instruction can be used to cancel the gantry synchronization instruction, and the axis number can be any axis number in the slave axis array;

The master axis number must be smaller than any axis number in the slave axis array;

The axis numbers of the slave axis array must be arranged from small to large.

5.5.8.1.2 Parameter

Parameter	Statement	Data Type	Description
UCMASTERAXIS	INPUT	USINT	Active synchronization axis number of gantry group (must be smaller than any slave axis number)
PSLAVEAXIS	INPUT	POINTER TO USINT	Slave synchronization axis number

			array of gantry group
UCSLAVEAXISNUM	INPUT	USINT	Number of slave synchronization axes in the gantry group (≤7)
UCMODE	INPUT	USINT	Gantry synchronization mode: 0: Inter-axis open-loop synchronization 1: Inter-axis closed-loop synchronization
LRPOSWARING	INPUT	LREAL	Inter-axis deviation alarm threshold
LRPOSERR	INPUT	LREAL	Inter-axis deviation stop protection threshold
HMC_GantrySynchro	OUTPUT	UDINT	Axis instruction ID: Success 0xFFFFFFFF: Function parameter error

5.5.8.1.3 Instruction type

Axis motion cache instructions.

5.5.8.1.4 Additional information

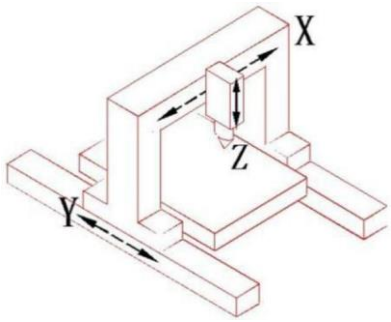
Within the same gantry group, all gantry axes must be of the same type;

When the gantry synchronization mode is set to open-loop mode, the axis type is the axis type with actual output, such as bus axis. When the closed-loop mode is selected, the axis type is the position closed-loop control axis type with actual output and actual feedback, such as bus speed control mode axis;

The cycle mode of the gantry axis must be limited stroke, that is, RepMode=0.

5.5.8.1.5 Example

For example, in the structure shown below, the Y-axis is a gantry dual drive, which can execute an arc trajectory on the work platform.



Structure Diagram

ucMaster := 1; (*Gantry master axis number*)

pSlaveAxisAry[0] := 2; (*Gantry slave axis number*)

ucSlaveNum := 1; (*A gantry slave axis*)

ucMode := 1; (*Inter-axis closed-loop synchronization*)

lrPosWaring := 100;

lrPosErr := 500;

HMC_GantrySynchro

(ucMaster,ADR(pSlaveAxisAry),ucSlaveNum,ucMode,lrPosWaring,lrPosErr); (*Send the gantry instruction to establish a gantry relationship between axis 0 and axis 1*)

HMC_MoveCircle (10,0, 1, 1,0,0, 100, 100); (*Axis 0 is the X-axis, axis 1 is the Y-axis, and axis 2 is in a gantry relationship with axis 1*)

The information after the instruction is sent is as shown in the figure:

Basic				
AxisID	USINT	0	1	2
AxisType	EmAxisType	Servo_AO	Servo_AO	Servo_AO
Units	LREAL	1	1	1
Status				
MCmdType	EmCmdType	HMC__MoveCircle	HMC__MoveCircle	HMC__GantrySynchro
NCmdType	EmCmdType	HMC__McIdle	HMC__McIdle	HMC__McIdle
AxisStatus	UDINT	0	0	0
AxisErrMask	UDINT	1	1	1

Parameter Diagram

5.5.8.2 HMC_GantryInit (Alignment Initialization of Gantry Synchronized Back to Reference Point Position)

5.5.8.2.1 Function

When the gantry starts to run, it must complete the alignment and return to the origin. Alignment needs to be completed by the alignment detector installed on each axis side of the gantry. The alignment detector must be installed accurately in its mechanical position, this is because this is the only place where the gantry can correct the parallel alignment. The entire gantry initialization process is as follows:

1. Each axis returns to the reference point independently in turn. During the process of one axis returning to the reference point, another gantry axis follows synchronously;

2. After all axes return to the reference point, redefine the positions of all gantry axes at the same time;

3. Calculate the deviation between axes. When the deviation exceeds the threshold, the instruction ends. When the deviation is within the threshold range, compensate and shift each axis to eliminate the deviation and

achieve alignment.

The back-to-reference-point function of gantry axes is similar to the back-to-origin function. It supports two modes: Home low-speed and HomeEx high-speed. When the HomeEx high-speed mode (not support) is selected, high-speed DI must be used.

5.5.8.2.2 Parameter

Input Parameters	Statement	Type	Parameter Description
PAXIS	INPUT	POINTER TO USINT	Synchronization axis number array of gantry group
UCAXISNUM	INPUT	USINT	Number of synchronization axes in the gantry group (≤8)
UCHOMEMODE	INPUT	USINT	For the origin regression mode of gantry synchronization back to the reference point, please refer to the origin regression mode description in the MC_Home instruction (gantry synchronization initialization supports modes 1/2/3/5/7/11/17/18/19/21/23/27)
PDICHL	INPUT	POINTER TO UDINT	DI signal used to return to the reference point (non-repeatable)
DREFERPOINTPOS	INPUT	LREAL	Reference point absolute position value
DPOSERR	INPUT	LREAL	Inter-axis deviation threshold
HMC_GantryInit	OUTPUT	UDINT	Axis instruction ID: Success 0xFFFFFFFF: Function parameter error

5.5.8.2.3 Instruction type

Axis motion cache instructions.

5.5.8.2.4 Additional information

The origin signal source used for the gantry axis to return to the reference point cannot be repeated;

Within the same gantry group, all gantry axes must be of the same type;

When the gantry synchronization mode is set to open-loop mode, the axis type is the axis type with actual output, such as bus axis. When the closed-loop mode is selected, the axis type is the position closed-loop control axis type with actual output and actual feedback, such as bus speed control mode axis;

The cycle mode of the gantry axis must be limited stroke, that is, RepMode=0；

The HMC_Cancel instruction can be used to cancel the gantry initialization instruction, and the axis number must be the smallest one in the axis array;

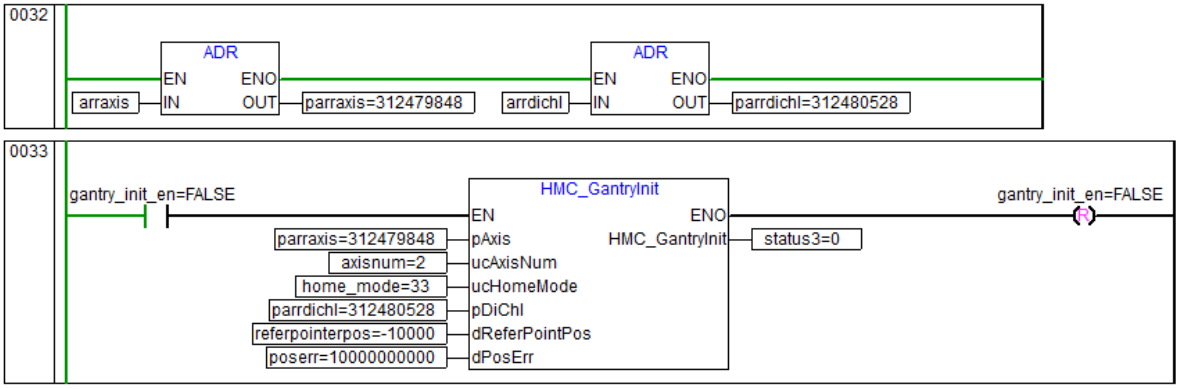
There is no requirement for the order of axis numbers in the axis array.

5.5.8.2.5 Example

After the gantry dual-drive structure is shut down and restarted, the two gantry axes are in a natural connection status, and there is a deviation. At this time, use the gantry initialization alignment instruction to make the two axes

return to the alignment reference point in sequence and initialize the axis positions. After obtaining the deviation between the gantry axes, eliminate the deviation between the two axes and complete the position initialization and alignment of the gantry axis. After that, send the gantry synchronization control instruction to the two axes to establish the gantry relationship, and then send the motion control instruction to the gantry master axis to realize the dynamic synchronous following control of the gantry axis.

Variable Name	Address	Variable Description	Variable Type	Initial Value
gantry_init_en			BOOL	FALSE
arraxis			ARRAY[0..1] OF USINT	
arrdichl			ARRAY[0..1] OF UDINT	
referpointerpos			LREAL	-10000
poserr			LREAL	10000000000
home_mode			USINT	33
status3			UDINT	0
parraxis			POINTER TO USINT	0
parrdichl			POINTER TO UDINT	0




```
(*Longmen synchronization initialization*)
IF gantry_init_en = TRUE THEN
    status3 := HMC_GantryInit(ADR(arraxis), 2, home_mode, ADR(arrdichl), referpointerpos, poserr);
    gantry_init_en := FALSE;
END_IF
```

Synchronization axis number array of gantry group:

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	arraxis[0]		Longmen synchronous axis 0	USINT	0	FALSE	Not Public	
0002	arraxis[1]		Longmen synchronous axis 1	USINT	1	FALSE	Not Public	

DI signal used to return to the reference point:

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	arrdichl[0]		Axis 0: DI signal	UDINT	38	FALSE	Not Public	
0002	arrdichl[1]		Axis 1: DI signal	UDINT	39	FALSE	Not Public	

5.5.8.3 HMC_SetGantryAxisPID (Setting Gantry Synchronization Axis Compensation PID Parameters)

5.5.8.3.1 Function

It is used in conjunction with the gantry synchronization control instruction HMC_GantrySynchro. Set the

dynamic correction PID parameters of the specified gantry axis in a gantry group

Parameter. If the specified axis number is not currently a gantry synchronization instruction, the setting will be unsuccessful.

5.5.8.3.2 Parameter

Input parameters	Description	Type	Parameter Description
UCAXISID	INPUT	USINT	Gantry group axis number
DKPGAIN	INPUT	LREAL	Dynamic compensation PID proportional coefficient (default value: 1)
DKIGAIN	INPUT	LREAL	Dynamic compensation PID differential coefficient (default value: 0)
DKDGAIN	INPUT	LREAL	Dynamic compensation PID integral coefficient (default value: 0)
HMC_SetGantryAxisPID	OUTPUT	UDINT	0: Success 0xFFFFFFFF: Function parameter error

5.5.8.3.3 Instruction type

It is a non-axis motion cache instruction.

5.5.8.3.4 Additional information

When this instruction is not called to set the PID, the dynamic compensation PID of each axis uses the default value.

5.5.8.4 HMC_GetGantryAxisPID (Reading gantry synchronization axis compensation PID parameters)

5.5.8.4.1 Function

It is used in conjunction with the gantry synchronization control instruction HMC_GantrySynchro. Read the dynamic correction PID parameters of the specified gantry axis in a gantry group.

5.5.8.4.2 Parameter

Input parameters	Description	Type	Parameter Description
UCAXISID	INPUT	USINT	Gantry group axis number
DKPGAIN	INPUT/OUTPUT	LREAL	Dynamic compensation PID proportional coefficient
DKIGAIN	INPUT/OUTPUT	LREAL	Dynamic compensation PID differential coefficient
DKDGAIN	INPUT/OUTPUT	LREAL	Dynamic compensation

			PID integral coefficient
HMC_GetGantryAxisPID	OUTPUT	UDINT	0: Success 0xFFFFFFFF: Function parameter error

5.5.8.4.3 Instruction type

It is a non-axis motion cache instruction.

5.5.8.4.4 Additional information

When the axis number is not a gantry axis, an error will be returned.

5.5.8.5 HMC_GantryGroupDefPos (Setting the current position of a gantry synchronization axis group)

5.5.8.5.1 Function

It is used in conjunction with the gantry synchronization control instruction HMC_GantrySynchro. Redefine the position of all gantry axes in a gantry group.

5.5.8.5.2 Parameter

Input parameters	Description	Type	Parameter Description
PAXIS	INPUT	POINTER TO USINT	Gantry group synchronization axis
UCAXISNUM	INPUT	USINT	Number of synchronization axes in the gantry group (≤8)
DMPOS	INPUT	LREAL	Predefined position
HMC_GantryGroupDefPos	OUTPUT	UDINT	0: Success 0xFFFFFFFF: Function parameter error

5.5.8.5.3 Instruction type

It is a non-axis motion cache instruction.

5.5.8.5.4 Additional information

When the current instruction is not a gantry synchronization instruction, an error will be returned.

5.5.8.6 HMC_GetGantryInitState (Reading the Current Status of a Gantry Synchronization Control Function)

5.5.8.6.1 Function

It is used in conjunction with the gantry synchronization alignment reference point initialization alignment instruction HMC_GantryInit. Read the current execution status of the instruction.

5.5.8.6.2 Parameter

Input parameters	Description	Type	Parameter Description
PAXIS	INPUT	POINTER TO USINT	Gantry group synchronization axis
UCAXISNUM	INPUT	USINT	Number of synchronization axes in the gantry group
HMC_GetGantryInitState	OUTPUT	UDINT	Gantry initialization instruction execution status: 1: Homing 2: ReturnBack 3: Defpos 4: Compsate 5: Finish 0xFFFFFFFF: Function parameter error

5.5.8.6. 3 Instruction type

It is a non-axis motion cache instruction.

5.5.9 Axis Position Compensation Function

5.5.9.1 HMC_SetAxisCompenPos (Setting Axis Position Compensation Quantity)

5.5.9.1.1 Function

Set axis position compensation quantity.

Note: The compensation amount shall be set before the HMC_TrigAxisCompens instruction so that it can take effect when the next compensation action is triggered. Changes in this value do not affect ongoing compensation actions.

5.5.9.1.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
dCompenPos	INPUT	LREAL	Position compensation value
HMC_SetAxisCompenPos	OUTPUT	UDINT	0: Execution succeeded 0xFFFFFFFF: Execution failed

5.5.9.1.3 Instruction type

It is a non-axis motion cache instruction.

5.5.9.2 HMC_SetAxisCompenPara (Setting Axis Position Compensation Motion Parameters)

5.5.9.2.1 Function

Set the kinematic parameters of the axis position compensation motion, which takes effect in real time.

5.5.9.2.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
dJerkMode	INPUT	USINT	Acceleration and deceleration type used in axis position compensation action: 0: Trapezoidal acceleration and deceleration 1: S-shaped acceleration and deceleration
dVmax	INPUT	LREAL	Maximum speed of position compensation action
dAccel	INPUT	LREAL	Acceleration of position compensation action
dDecel	INPUT	LREAL	Deceleration of position compensation action
dJerk	INPUT	LREAL	Jerk of S-shaped acceleration and deceleration
HMC_SetAxisCompenPara	OUTPUT	UDINT	0: Execution succeeded 0xFFFFFFFF: Execution failed

5.5.9.2.3 Instruction type

It is a non-axis motion cache instruction.

5.5.9.3 HMC_TrigAxisCompens (Triggering Position Compensation)

5.5.9.3.1 Function

Trigger an axis position compensation action. Each time this instruction is run, a position compensation action is performed using the currently set axis position compensation amount. If the current compensation action has not ended when this instruction is called, the remaining unfinished compensation is abandoned and a new compensation action is started immediately. The HMC_GetAxisCompenState instruction can be called to query the axis's compensation status to confirm whether the current compensation action has been completed.

5.5.9.3.2 Parameter

Parameters	Staetement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
HMC_TrigAxisCompens	OUTPUT	UDINT	0: Execution succeeded 0xFFFFFFFF: Execution failed

5.5.9.3.3 Instruction type

It is a non-axis motion cache instruction.

5.5.9.4 HMC_StopAxisCompens (Stopping Position Compensation)

5.5.9.4.1 Function

Immediately stop the axis position compensation action in progress.

5.5.9.4.2 Parameter

Parameters	Staetement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
HMC_StopAxisCompens	OUTPUT	UDINT	0: Execution succeeded 0xFFFFFFFF: Execution failed

5.5.9.4.3 Instruction type

It is a non-axis motion cache instruction.

5.5.9.5 HMC_GetAxisCompenState (Getting Axis Position Compensation Status)

5.5.9.5.1 Function

Get the axis position compensation status.

5.5.9.5.2 Parameter

Parameters	Staetement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
HMC_GetAxisCompenState	OUTPUT	UDINT	0: Compensation action completed 1: Compensation action in progress 0xFFFFFFFF: Axis parameter error

5.5.9.5.3 Instruction type

It is a non-axis motion cache instruction.

5.6 Robot Control Instructions

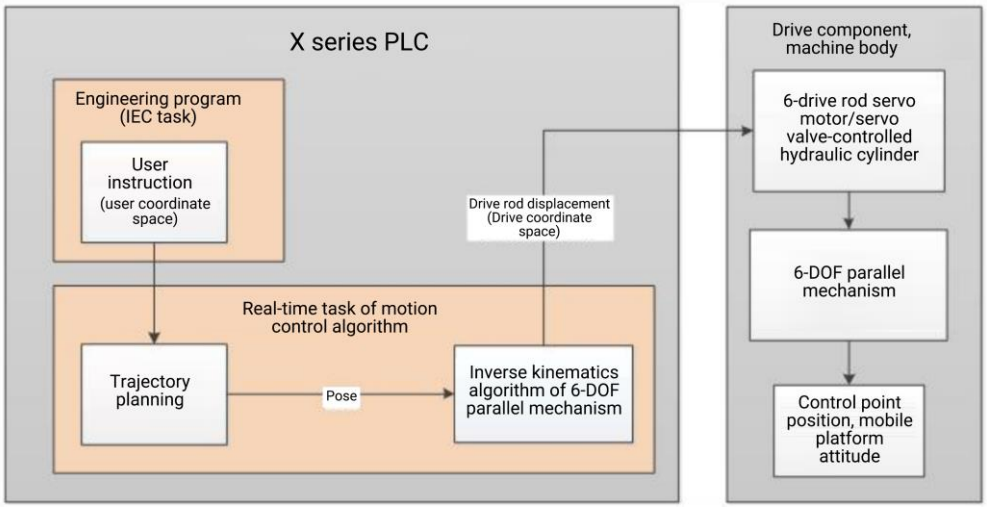
5.6.1 Stewart 6-DOF Parallel Mechanism

5.6.1.1 HMC_StewartControl (Stewart Mechanism Control)

5.6.1.1.1 Function

This instruction completes the kinematic transformation of the Gough-Stewart 6-DOF parallel mechanism (from user coordinate space to drive coordinate space). After the instruction takes effect, the user engineering

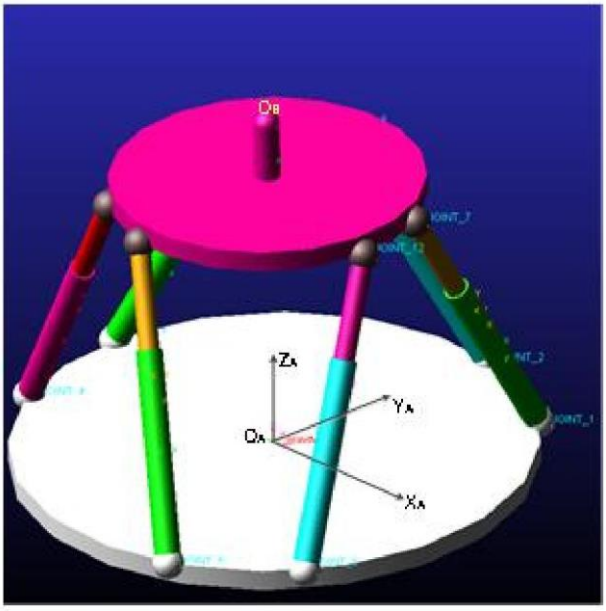
program can be written in the user coordinate space (Cartesian coordinate system). The controller automatically completes the inverse kinematic solution transformation of the mechanism and controls the six drive rods for motion in the drive coordinate space. The process is shown in the figure below. For details about the Gough-Stewart platform, please refer to the additional information.



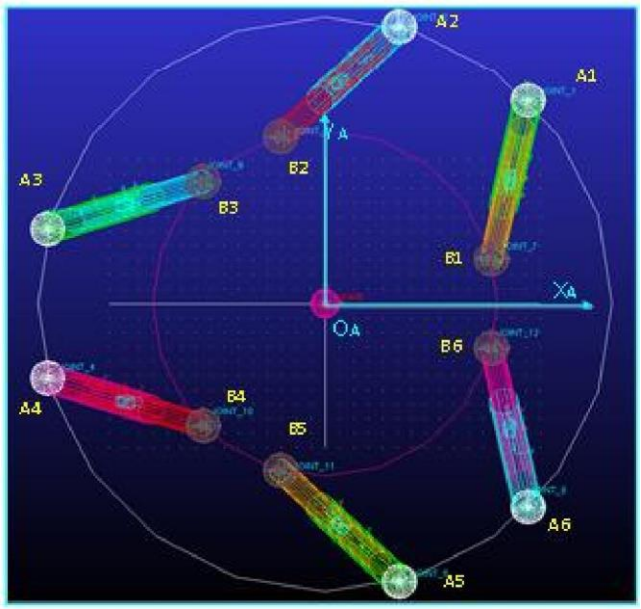
Block Diagram of 6-DOF Parallel Mechanism Control System

The attitude description uses the RPY angle, which is the definition of rotation around a fixed coordinate system, with the rotation sequence of XYZ.

After the return-to-zero operation of the six drive rods is completed on the Gough-Stewart platform, and the actual rod lengths of the six drive rods are equal to the parameter StewartParalinit, call this instruction to complete the initialization of the 6-DOF parallel platform and configure the mechanism parameters of the 6-DOF parallel mechanism, the axis numbers of the drive coordinate space (6), and the axis numbers of the user coordinate space (6).



(a)



(b)

Diagram of Coordinates and Mechanism Parameters of Parallel 6-DOF Platform

As shown in the figure above, the target control point is located on a straight line that passes through the center of the circumcircle of the mobile platform and is perpendicular to the plane of the circle, and the distance from the plane of the moving platform is H.

The absolute coordinate system $O_A - X_A Y_A Z_A$ fixed on the static platform is defined as follows:

The coordinate origin is located at the center of the circumcircle of the hinge point of the static platform;

The axis is perpendicular to the plane of the static platform and upward;

The axis is located on the bottom surface and is perpendicular to the line connecting lower hinge points 1 and 6;

The axis direction can be determined by the right-hand rule.

Let x, y, and z be the position coordinates of the target control point O_B in $O_A - X_A Y_A Z_A$, and φX, φY, and φZ be the attitude coordinates of the mobile platform in $O_B - X_B Y_B Z_B$. Then the pose coordinates of the mobile platform in the absolute coordinate system $O_A - X_A Y_A Z_A$ are:

$X=[x, y, z, \varphi_X, \varphi_Y, \varphi_Z]^T$

After the initialization is completed by calling this instruction, the coordinates of each axis in the user coordinate space are automatically initialized to the initial pose:

$X_{init}=[0, 0, zinit, 0, 0, 0]^T$

The coordinates of each axis in the drive coordinate space are initialized to the initial rod length parameters. Please note the following points when using this instruction:

This function can only be called when the actual lengths of the six drive rods are equal to the parameter linit in the StewartPara array, or else the initial pose will deviate;

HMC_Cancel(AxisID,CancelMode) can be used to cancel this instruction, and the axis number passed in during canceling must be the smallest axis number among the six drive axes;

After the instruction is called successfully, when it is executed for the first time, the displacement of each axis in the user coordinate space will be automatically set as the pose of the mobile platform in the absolute coordinate system OA -XAYAZA, and the displacement of each axis in the drive coordinate space will be set as the initial rod length linit. Based on the initial pose, the motion is performed;

When the instruction is executed for the first time, the RepMode of each axis in the drive coordinate space and user coordinate space will be set to limited stroke, and the soft limit parameters of each drive axis will be automatically set according to the drive rod length parameters lmin and lmax. The axis parameter RepMode is not allowed to be modified during the execution of the instruction, or else coordinate transformation errors may occur;

After this instruction is called and the initialization is successful, the user can program in the absolute coordinate system $O_A-X_AY_AZ_A$, and the instruction is not allowed to be called;

The HMC_DefPos or HMC_DefOrigin instruction shall be used to reset the coordinates of each axis in the user coordinate space or drive coordinate space, or else serious deviations may occur during coordinate transformation. If the user needs to use the relative user coordinate system, he can perform coordinate offset conversion by himself in the engineering program;

Once this instruction is called successfully, the HOME instruction is not allowed to be used in the user coordinate space because the HOME instruction will have a jump in DPOS and Mpos when the back-to-zero operation is completed, resulting in a jump in the output of the drive axis after coordinate transformation;

This parallel mechanism has a singular configuration within the accessible working space. Near the singular configuration, the mechanism has poor mechanical performance and even loses its load-bearing capacity. Please plan the working space reasonably according to the mechanism parameters and avoid singular configurations and mechanical interference configurations.

5.6.1.1.2 Parameter

Parameters	Staetement	Data Type	Description
UserSpaceAxisID	INPUT	POINTER TO USINT	User coordinate space axis number array, consisting of six elements, corresponding to the translation axes X, Y, and Z and the rotation axes x, y, and z. Unit in degrees. Generally, they are virtual axes. The user can program these six axes and plan the pose of the mobile platform.
DriveSpaceAxisID	INPUT	POINTER TO USINT	Drive axis number array, consisting of six elements, corresponding to drive rods 1~6. As shown in the figure, AiBi corresponds to the physical output axis of the drive rod i, and drives the six drive rods of the mechanism to move.
StewartPara	INPUT	POINTER TO LREAL	Gough-Stewart mechanism parameter array, consisting of eight elements as follows: 1. Radius R0 of the circumcircle of the hinge point on the static platform

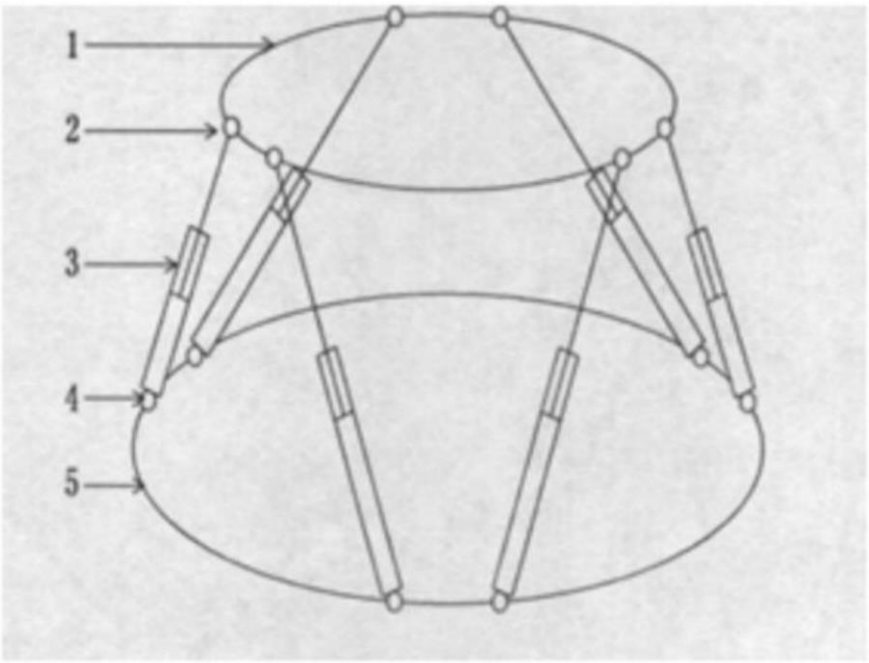
Parameters	Staetement	Data Type	Description
			2. The center angle of the circle corresponding to the short edge of the hinge point on the static platform (0° ≤ 0 6°) 3. Radius R1 of the circumcircle of the hinge point on the mobile platform 4. Center angle of the circle corresponding to the short edge of the hinge point on the mobile platform (1° ≤ 16°) 5. Distance H between the target control point and the plane where each hinge point of the mobile platform is located (the target control point is located on a straight line that passes through the center of the circumcircle of the mobile platform and is perpendicular to the plane of the circle). H≥0 6. Initial rod length of the drive rod linit 7. The shortest rod length of the drive rod l in 8. The longest rod length of the drive rod l x
HMC_StewartControl	OUTPUT	UDINT	Instruction ID: Success 16#FFFFFFFF: Error

The input parameter length is the user unit, and the angle unit adopts the degree measure (°).

5.6.1.1.3 Instruction type

Axis motion cache instructions.

5.6.1.1.4 Additional information



Mechanism Principle of Parallel 6-DOF Platform

1: Mobile platform

2: Upper ball hinge

3: Hydraulic cylinder/electric cylinder

4: Lower ball hinge

5: Static platform

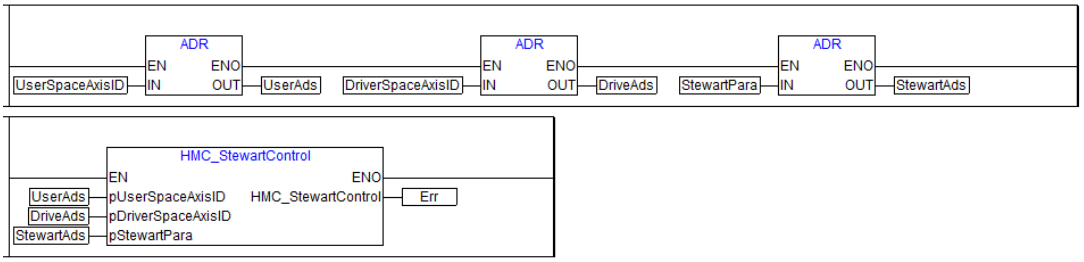
The mechanism principle of the 6-DOF parallel platform is shown in the figure above. The system consists of a mobile platform, a static platform, and six servo electric cylinders (or servo valve-controlled hydraulic cylinders). Each branch chain electric cylinder is connected to the mobile and static platforms by ball hinges respectively (in actual use, to eliminate local degrees of freedom, the upper hinge points of the platform are usually connected using Hooke hinges). Generally, the static platform is fixed on the base. By controlling the telescopic motion of the branch chain electric cylinder, the mobile platform can realize spatial motion and complete position and attitude adjustment with six degrees of freedom in space. The parallel mechanism itself has high stiffness. When driven by a hydraulic cylinder, it is especially suitable for simulating various spatial motions under heavy loads.

5.6.1.1.5 Example

Task 1: Send the StewartControl instruction.

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
UserSpaceAxisID		User axis number array	ARRAY[0..5] OF USINT		FALSE	Not Public	
DriverSpaceAxisID		Drive shaft number array	ARRAY[0..5] OF USINT		FALSE	Not Public	
StewartPara		Institutional parameters	ARRAY[0..7] OF LREAL		FALSE	Not Public	
Err		Return value	DWORD	0	FALSE	Not Public	

LD program:



ST program:

```
(*Send StewartControl command*)
Err := HMC_StewartControl(ADR(UserSpaceAxisID), ADR(DriverSpaceAxisID), ADR(StewartPara));
```

Variable definition:

(1) User axis number array: (Duplicate axis numbers or duplicate drive axis numbers are not allowed)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
UserSpaceAxisID[0]		X-axis position	USINT	10	FALSE	Not Public	
UserSpaceAxisID[1]		Y-axis position	USINT	11	FALSE	Not Public	
UserSpaceAxisID[2]		Z-axis position	USINT	12	FALSE	Not Public	
UserSpaceAxisID[3]		X-axis gesture	USINT	13	FALSE	Not Public	
UserSpaceAxisID[4]		Y-axis gesture	USINT	14	FALSE	Not Public	
UserSpaceAxisID[5]		Z-axis gesture	USINT	15	FALSE	Not Public	

(2) Drive axis number array: (Bus encoder axis type or unused axis type is not allowed)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
DriverSpaceAxisID[0]		Drive lever 1	USINT	0	FALSE	Not Public	
DriverSpaceAxisID[1]		Drive lever 2	USINT	1	FALSE	Not Public	
DriverSpaceAxisID[2]		Drive lever 3	USINT	2	FALSE	Not Public	
DriverSpaceAxisID[3]		Drive lever 4	USINT	3	FALSE	Not Public	
DriverSpaceAxisID[4]		Drive lever 5	USINT	4	FALSE	Not Public	
DriverSpaceAxisID[5]		Drive lever 6	USINT	5	FALSE	Not Public	

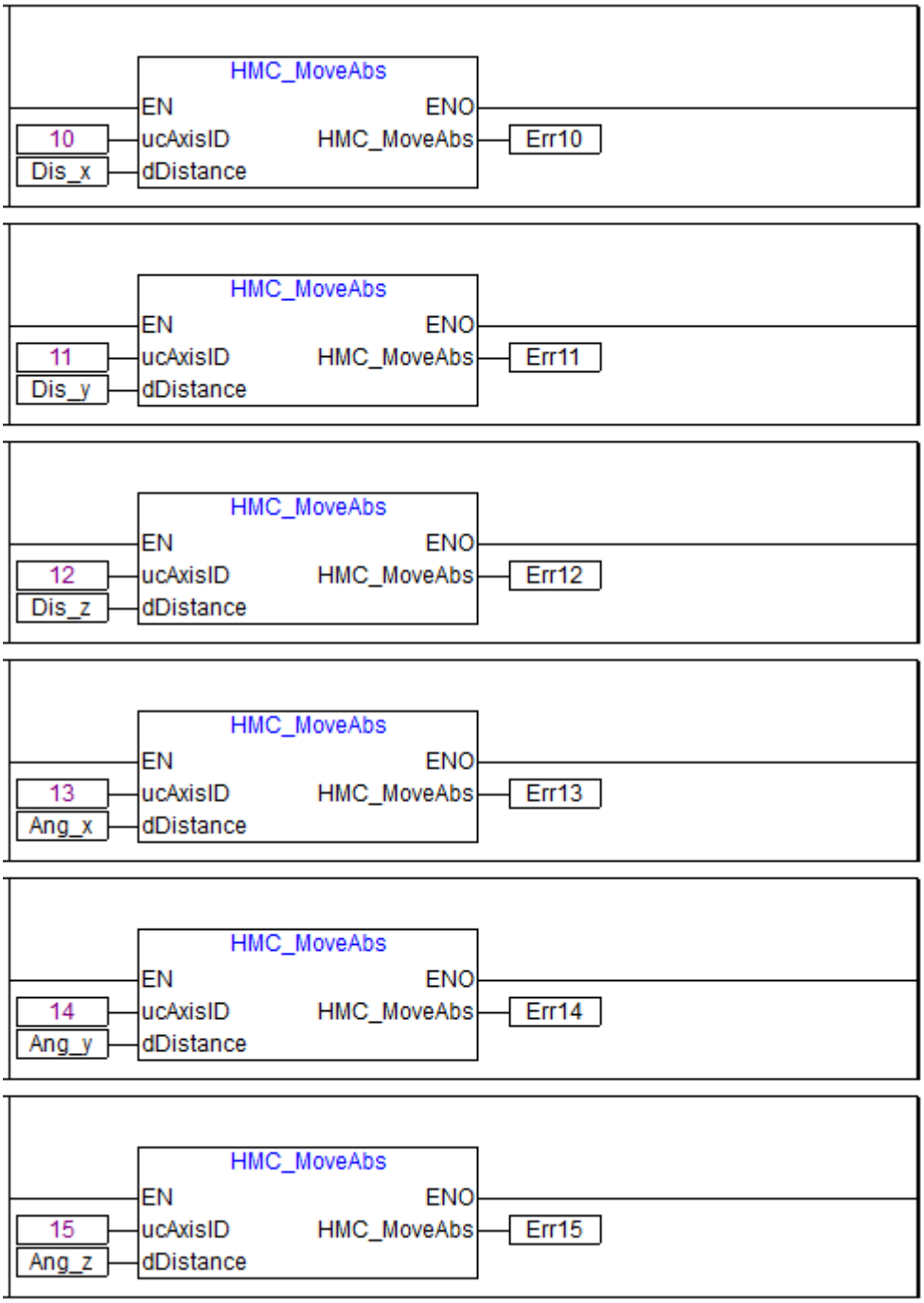
(3) Mechanism parameters: (When the mechanism parameters are set unreasonably, the instruction will be sent unsuccessfully)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
StewartPara[0]		Moving platform radius	LREAL	10000	FALSE	Not Public	
StewartPara[1]		Angle of the short side of the moving platform	LREAL	30	FALSE	Not Public	
StewartPara[2]		Static platform radius	LREAL	20000	FALSE	Not Public	
StewartPara[3]		Short edge angle of static platform	LREAL	30	FALSE	Not Public	
StewartPara[4]		Control point distance to the height of the moving platform	LREAL	1000	FALSE	Not Public	
StewartPara[5]		Initial rod length	LREAL	20000	FALSE	Not Public	
StewartPara[6]		Minimum rod length	LREAL	10000	FALSE	Not Public	
StewartPara[7]		Maximum pole length	LREAL	30000	FALSE	Not Public	

Example 2: When motion control instructions are sent to the user axis (single-axis absolute motion, 6-axis absolute linear interpolation (N-axis interpolation), arc, spherical arc, spline, etc.), the drive axis will adjust the position and attitude.

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
Dis_x		X-axis coordinate position	LREAL	1000	FALSE	Not Public	
Dis_y		Y-axis coordinate position	LREAL	2000	FALSE	Not Public	
Dis_z		Z-axis coordinate position	LREAL	3000	FALSE	Not Public	
Ang_x		X-axis gesture_angle	LREAL	10	FALSE	Not Public	
Ang_y		Y-axis gesture_angle	LREAL	-20	FALSE	Not Public	
Ang_z		Z-axis gesture_angle	LREAL	30	FALSE	Not Public	

LD program:



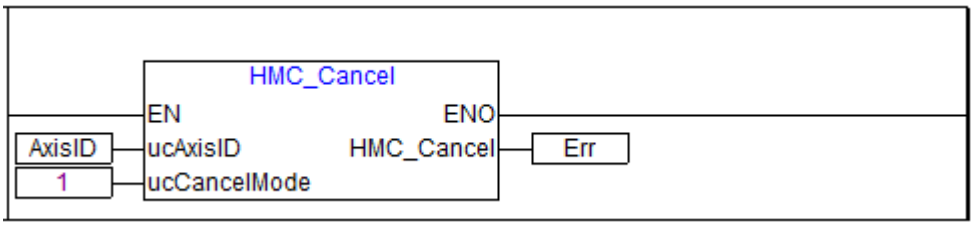
ST program:

```
(*Adjustment of x-axis position*)
HMC_MoveAbs(10, Dis_x);
(*Adjustment of y-axis position*)
HMC_MoveAbs(11, Dis_y);
(*Adjustment of z-axis position*)
HMC_MoveAbs(12, Dis_z);
(*Adjustment of x-axis angle*)
HMC_MoveAbs(13, Ang_x);
(*Adjustment of y-axis angle*)
HMC_MoveAbs(14, Ang_y);
(*Adjustment of z-axis angle*)
HMC_MoveAbs(15, Ang_z);
(*You can also throw ball arcs or spline commands to plan the position and posture of the user axis*)
```

Example 3: Cancel the StewartControl instruction (AxisID must be the smallest axis number in the drive axis number array)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
AxisID		The first axis number in the drive s...	BYTE	0	FALSE	Not Public	

LD program:



ST program:

```
(*Revoke StewartControl directive*)
HMC_Cancel(AxisID, 1);
```

5.6.2 5-DOF SCARA Robot Arm

5.6.2.1 HMC_Scara5Control (5-DOF SCARA Robot Arm Control)

5.6.2.1.1 Function

This instruction completes the kinematic transformation of the 5-DOF SCARA mechanism (from the base coordinate system to the joint coordinate system). After the instruction takes effect, the user can use base coordinates (Cartesian coordinate system) to write user engineering programs, and the controller automatically completes the inverse kinematic solution transformation of the mechanism and drives the motion of each joint. The process is shown in Figure 205.

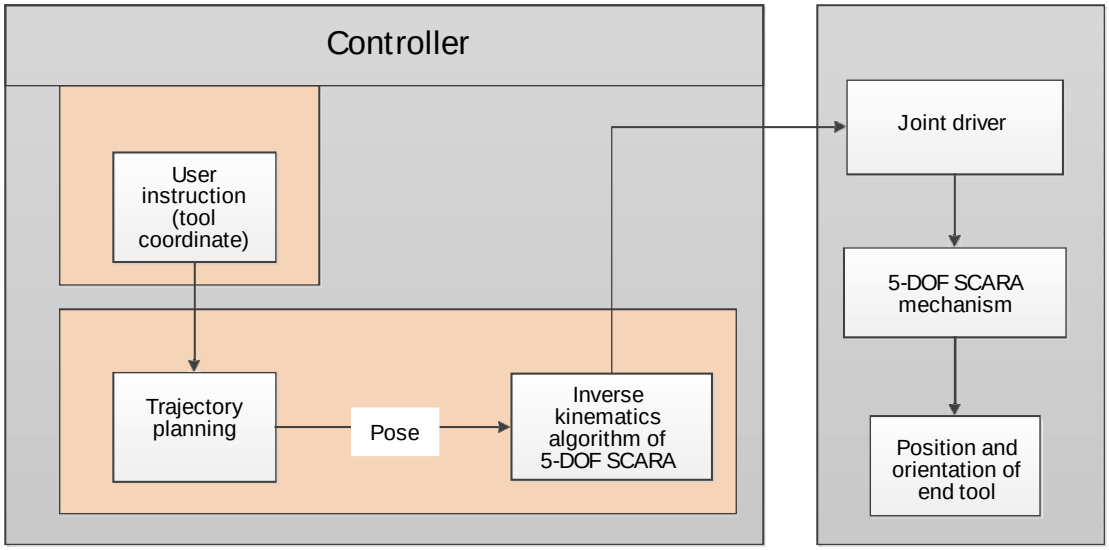


Figure 205 Block Diagram of 5-DOF SCARA Serial Mechanism Control System

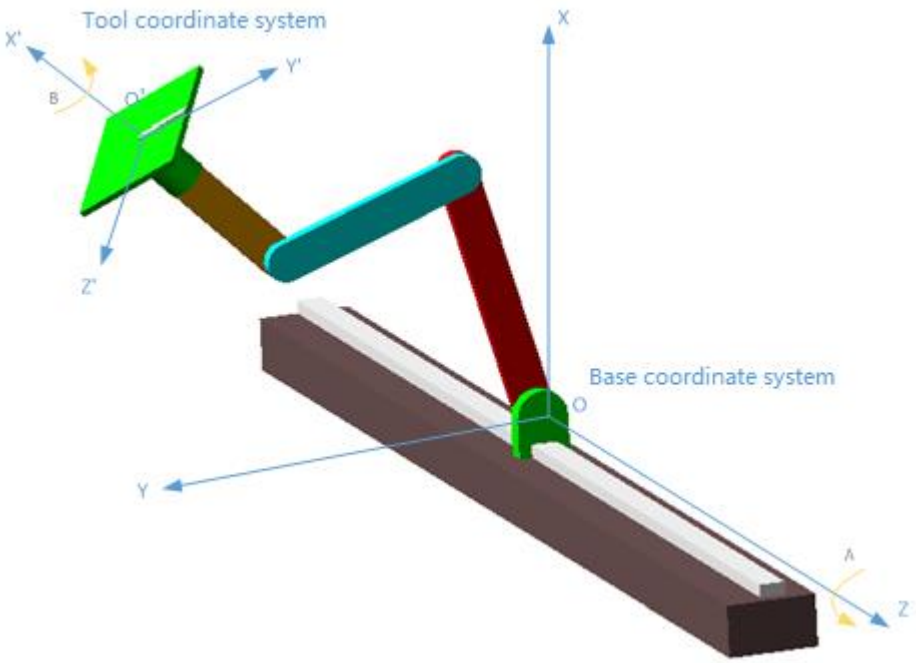


Figure 206 Definitions of Base Coordinate System and Tool Coordinate System of 5-DOF SCARA Manipulator (Cartesian Coordinates)

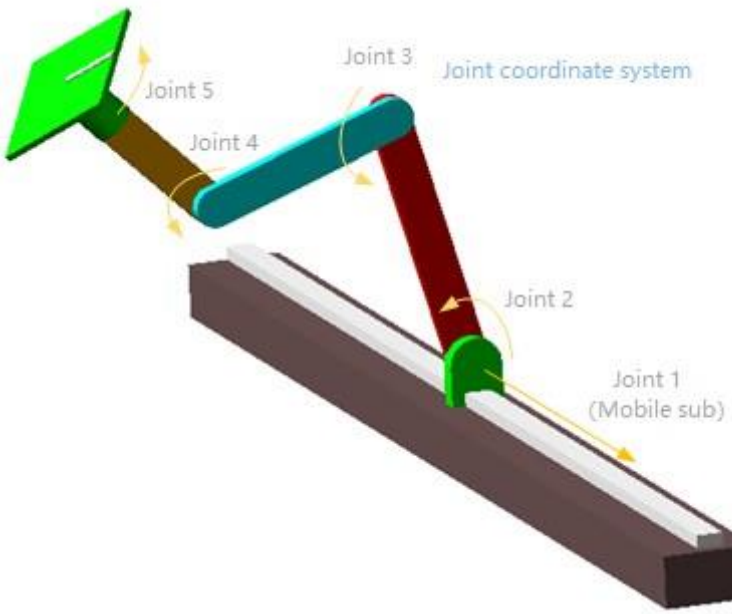


Figure 207 Definition of Joint Coordinate System of 5-DOF SCARA Manipulator (Generalized Coordinates)

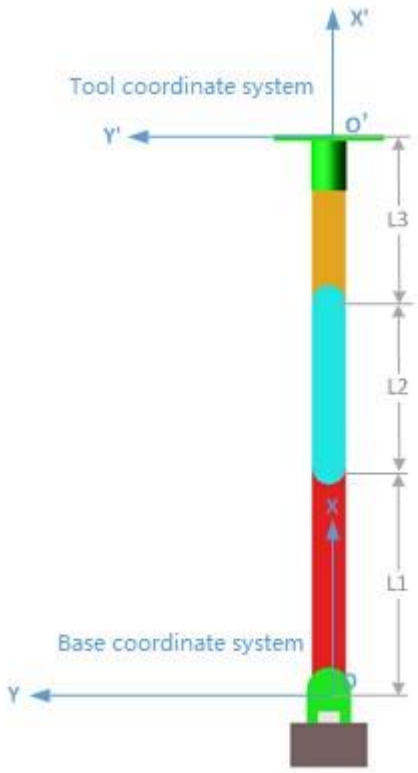


Figure 208 Zero-point Pose of Joint Angle of 5-DOF SCARA Manipulator

The definitions of the base coordinate system, tool coordinate system, and joint coordinate system are shown in Figure 206 and Figure 207. The joint angle zero point position is shown in Figure 208.

Definition of tool coordinates: The position and attitude of the tool coordinate system in the base coordinate system describe the target pose of the mechanism, that is, the tool coordinates of the mechanism. There are five coordinate components in total, namely (x, y, z, Φ_A, Φ_B).

The attitude description uses Euler angles, which is the rotation definition method of the orbiting coordinate system. The rotation sequence is Z-X', which corresponds to the A-axis and B-axis in Figure 206 respectively.

The zero point position of the joint angle is shown in Figure 208. After the back-to-zero operation is completed, as long as the zero point position has not changed, this instruction can be called at any position.

Please note the following points when using this instruction:

(1) The length unit uses user unit, and the angle unit adopts the degree measure (°);

(2) The zero point position of the joint angle is shown in Figure 208;

(3) HMC_Cancel(AxisID,CancelMode) can be used to cancel this instruction, and the axis number passed in during canceling must be the smallest axis number among the five drive axes;

(4) After the instruction is called successfully, when it is executed for the first time, the tool coordinates will be initialized based on the current joint angle, and then it will move based on this initial pose;

(5) When the instruction is executed for the first time, the RepMode of each axis in the drive coordinate space and user coordinate space will be set to limited stroke. The user can set the soft limit parameters of each joint angle

of the mechanism according to the specific structure of the mechanism. RepMode is not allowed to be modified during the execution of the instruction, or else coordinate transformation errors may occur;

(6) The limit value of each joint angle of the mechanism can be set according to the actual situation of the mechanism;

(7) After this instruction is called and the initialization is successful, the user can program in the base coordinate system. It is not allowed to call the HMC_DefPos or HMC_DefOrigin instruction to reset the coordinate values of each axis in the base coordinate system or joint coordinate system, or else serious deviations may occur during coordinate transformation. If the user needs to use the relative coordinate system, he can perform coordinate offset conversion by himself in the engineering program;

(8) The HOME instruction is not allowed to be used on the axes in the base coordinate system when this instruction is running, because the HOME instruction will have a jump in DPOS and Mpos when the back-to-zero operation is completed, resulting in a jump in the output of the joint axis after coordinate transformation;

(9) The accessible working space of the mechanism is limited. If the planned path exceeds its accessible working space, the instruction will be canceled for safety reasons. Please plan the working space reasonably according to the mechanism parameters, and ensure that the working space is within the accessible working space of the mechanism (the HMC_Scara5ReverseKinematics instruction can be used to assist in verifying whether the planned tool coordinates are valid).

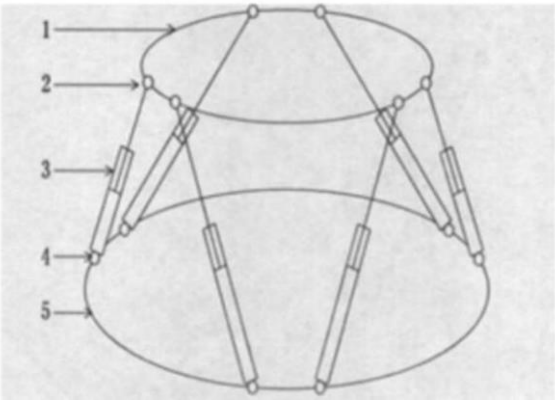
5.6.2.1.2 Parameter

Parameters	Statement	Data Type	Description
arrBaseCoordAxisID	INPUT	POINTER TO USINT	Axis number array in the base coordinate system, consisting of five elements, corresponding to the translation axes X, Y, and Z, and the rotation axes A and B, as shown in Figure 206 and Figure 207. Unit in degrees. Generally, they are virtual axes. The user can program these five axes and plan the pose of the tool coordinate system
arrJointCoordAxisID	INPUT	POINTER TO USINT	Axis number array of each joint axis in the joint coordinate system, consisting of five elements, corresponding to joints 1~5 in sequence. As shown in Figure 206 and Figure 207, the physical output axis drives the five joints of the drive mechanism to move.
arrScaraLinkLength	INPUT	POINTER TO LREAL	Connecting rod length array, consisting of three elements, corresponding to L1, L2, and L3 in Figure 208.
HMC_Scara5Control	OUTPUT	UDINT	Instruction ID: Success 16#FFFFFFFF: Error

5.6.2.1.3 Instruction type

Axis motion cache instructions.

5.6.2.1.4 Additional information



- 1: Mobile platform
- 2: Upper ball hinge
- 3: Hydraulic cylinder/electric cylinder
- 4: Lower ball hinge
- 5: Static platform

The mechanism principle of the 6-DOF parallel platform is shown in the figure above. The system consists of a mobile platform, a static platform, and six servo electric cylinders (or servo valve-controlled hydraulic cylinders). Each branch chain electric cylinder is connected to the mobile and static platforms by ball hinges respectively (in actual use, to eliminate local degrees of freedom, the upper hinge points of the platform are usually connected using Hooke hinges). Generally, the static platform is fixed on the base. By controlling the telescopic motion of the branch chain electric cylinder, the mobile platform can realize spatial motion and complete position and attitude adjustment with six degrees of freedom in space. The parallel mechanism itself has high stiffness. When driven by a hydraulic cylinder, it is especially suitable for simulating various spatial motions under heavy loads.

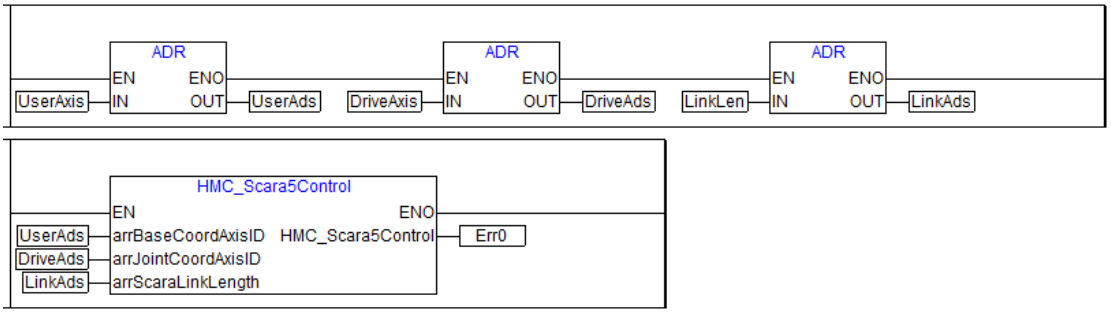
5.6.2.1.5 Example

Example 1: Send the Scara5Control instruction.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
UserAxis		User base coordinate axis array	ARRAY[0..4] OF BYTE		FALSE	Not Public	
DriveAxis		Joint drive shaft array	ARRAY[0..4] OF BYTE		FALSE	Not Public	
LinkLen		Rod length array	ARRAY[0..2] OF LREAL		FALSE	Not Public	
Err0		Return value	UDINT	0	FALSE	Not Public	

LD program:



ST program:

```
(*Send Scara5Control command*)
Err0 := HMC_Scara5Control(ADR(UserAxis), ADR(DriveAxis), ADR(LinkLen));
```

Variable definition:

(1) User base coordinate axis number array: (Duplicate axis numbers or duplicate drive axis numbers are not allowed)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
UserAxis[0]		User base coordinate X	BYTE	6	FALSE	Not Public	
UserAxis[1]		User base coordinate Y	BYTE	7	FALSE	Not Public	
UserAxis[2]		User base coordinate Z	BYTE	8	FALSE	Not Public	
UserAxis[3]		User rotates coordinate A	BYTE	9	FALSE	Not Public	
UserAxis[4]		User rotates coordinate B	BYTE	10	FALSE	Not Public	

(2) Joint drive axis number array: (Bus encoder axis type or unused axis type is not allowed)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
DriveAxis[0]		Joint 1 drive shaft	BYTE	1	FALSE	Not Public	
DriveAxis[1]		Joint 2 drive shaft	BYTE	2	FALSE	Not Public	
DriveAxis[2]		Joint 3 drive shaft	BYTE	3	FALSE	Not Public	
DriveAxis[3]		Joint 4 drive shaft	BYTE	4	FALSE	Not Public	
DriveAxis[4]		Joint 5 drive shaft	BYTE	5	FALSE	Not Public	

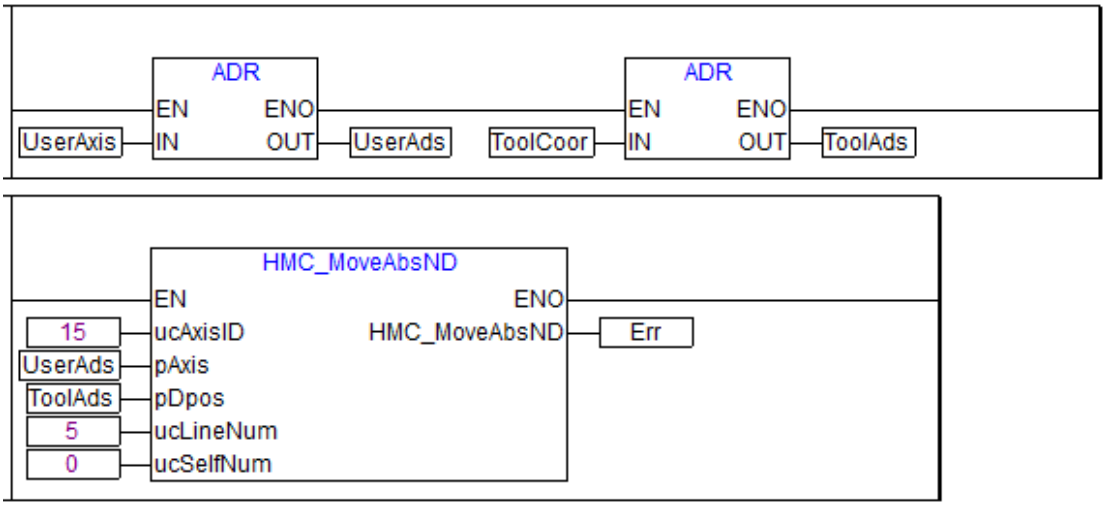
(3) Rod length parameters: (When the rod length is set unreasonably, the instruction will be sent unsuccessfully)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
LinkLen[0]		Rod 1 length	LREAL	10000	FALSE	Not Public	
LinkLen[1]		Rod 2 length	LREAL	11000	FALSE	Not Public	
LinkLen[2]		Rod 3 length	LREAL	12000	FALSE	Not Public	

Example 2: Send the motion control instruction (MoveAbsND) to the user base coordinate axis, and set the tool coordinates in the user base coordinate system.

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
UserAxis		User base coordinate axis array	ARRAY[0..4] OF BYTE		FALSE	Not Public	
ToolCoor		Tool coordinates	ARRAY[0..4] OF LREAL		FALSE	Not Public	

LD program:



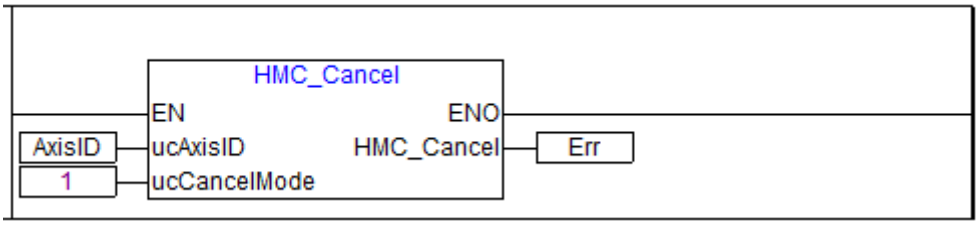
ST program:

```
(*Send the MoveAbsND command to the user's base coordinates, giving the tool coordinates in the user's base coordinates*)
HMC_MoveAbsND(15, ADR(UserAxis), ADR(ToolCoor), 5, 0);
```

Example 3: Cancel the Scara5Control instruction (AxisID must be the smallest axis number in the drive axis number array)

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
AxisID		The first axis number in the drive s...	BYTE	0	FALSE	Not Public	

LD program:



ST program:

```
(*Revoke StewartControl directive*)
HMC_Cancel(AxisID, 1);
```

5.6.2.2 HMC_Scara5Forward Kinematics (5-DOF SCARA Forward Kinematics)

5.6.2.2.1 Function

This instruction completes the forward kinematics of the 5-DOF SCARA mechanism (from the joint coordinate system to the base coordinate system).

5.6.2.2.2 Parameter

Parameters	Statement	Data Type	Description
arrScaraLinkLength	INPUT	POINTER TO LREAL	Connecting rod length array, with a total of three elements, corresponding to L1, L2, and L3 in the

			zero point pose of the joint angle of the SCARA manipulator with 208-DOF in Figure 208
arrToolCoordinate	INPUT	POINTER TO LREAL	Tool coordinate array, consisting of five elements, to save the forward solution results
arrJointAngle	INPUT	POINTER TO LREAL	Joint coordinate array, consisting of five elements
HMC_Scara5ForwardKinematics	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.6.2.2.3 Instruction type

It is a non-axis motion cache instruction.

5.6.2.3 HMC_Scara5ReverseKinematics (5-DOF SCARA Inverse Kinematics)

5.6.2.3.1 Function

This instruction completes the inverse kinematics of the 5-DOF SCARA mechanism (from the base coordinate system to the joint coordinate system). When the input tool coordinates exceed the accessible working space of the mechanism, an error is returned.

This function can be used in the following ways:

Check whether the tool coordinates of the key points are within the accessible working space;

Implement joint coordinate space programming.

Joint coordinate space programming can avoid the problem that the motion trajectories between key points may pass through the inaccessible working space of the mechanism when programming in the Cartesian coordinate system.

If only the positioning motion from point to point is required but the intermediate motion trajectories are not required, you can plan the tool coordinates of the key position points in the base coordinate system, and then use this instruction to check the validity of the key points and transform the tool coordinates of valid key points into joint coordinates, after which you can program in the joint coordinate system.

5.6.2.3.2 Parameter

Parameters	Statement	Data Type	Description
arrScaraLinkLength	INPUT	POINTER TO LREAL	Connecting rod length array, with a total of three elements, corresponding to L1, L2, and L3 in the zero point pose of the joint angle of the SCARA manipulator with 208-DOF in Figure 208
arrToolCoordinate	INPUT	POINTER TO LREAL	Tool coordinate array, consisting of five elements, namely (x, y, z, φA, φB)
arrReferenceJointAngle	INPUT	POINTER TO	Joint coordinate array (reference value), consisting of five elements, which are the

		LREAL	rotation angles of joints 1 to 5 in sequence Since there is more than one inverse solution result, a group of solutions that are closer to the reference position is selected as the inverse solution result.
arrJointAngle	INPUT	POINTER TO LREAL	Joint coordinate array, with a total of five elements to store the inverse kinematics result, which are the rotation angles of joints 1 to 5 in sequence
HMC_Scara5ReverseKinematics	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.6.2.3.3 Instruction type

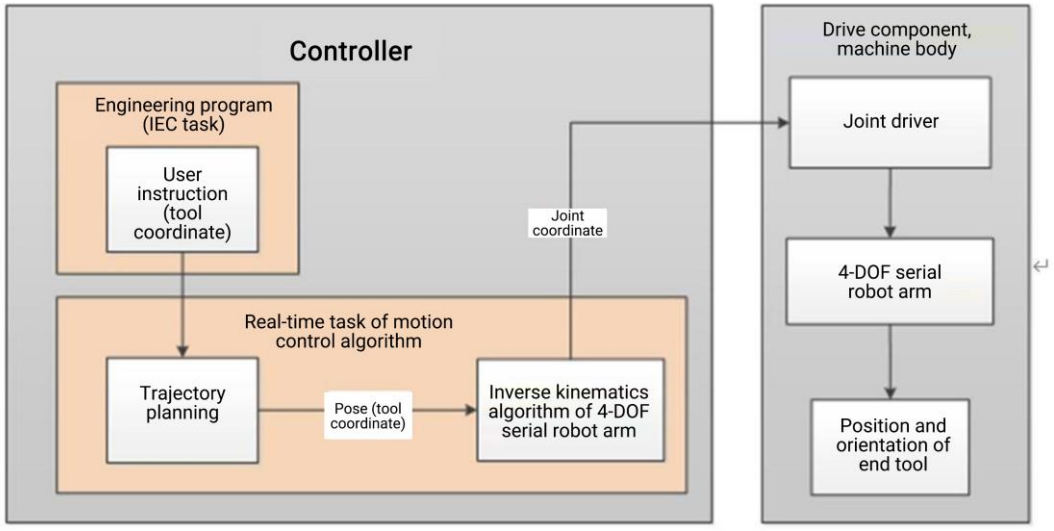
It is a non-axis motion cache instruction.

5.6.3 4-DOF Serial Robot Arm

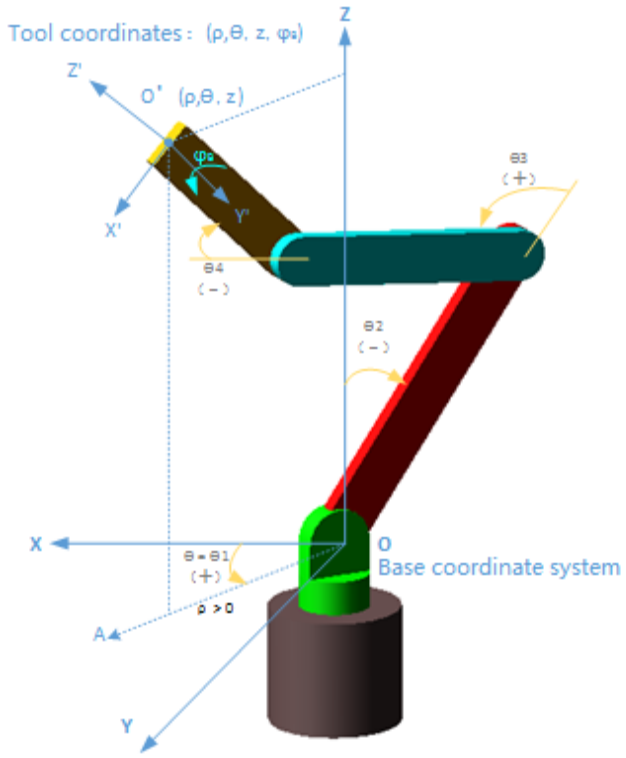
5.6.3.1 HMC_Serial4Control (4-DOF Serial Robot Arm Control)

5.6.3.1.1 Description of Functions

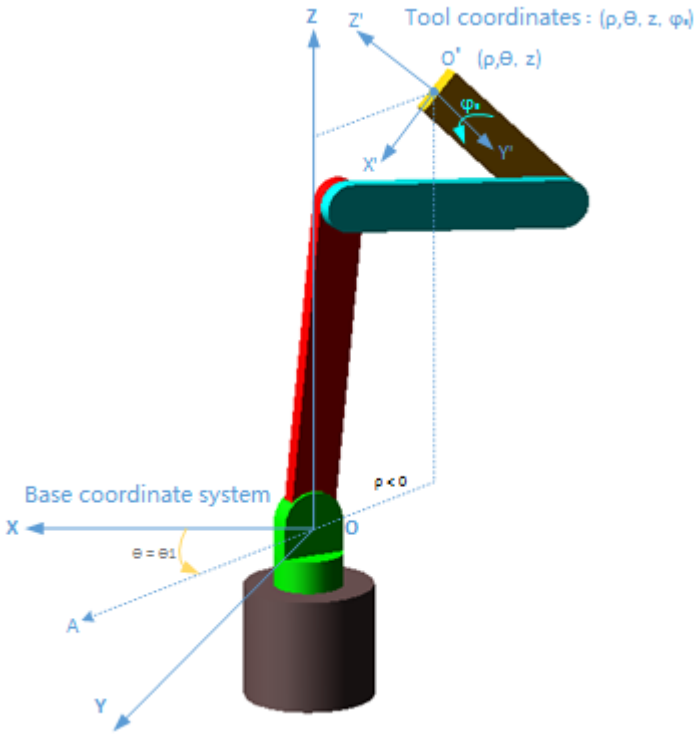
This instruction is used to complete the kinematic transformation of the 4-DOF serial robot arm (from tool coordinates to joint coordinates). After the instruction takes effect, the user can use tool coordinates (cylindrical coordinates) to write engineering programs, and the controller automatically completes the inverse kinematic solution transformation of the mechanism and drives the motion of each joint. The process is shown in the figure below.



Block Diagram of 4-DOF Serial Robot Arm Control System



(a)

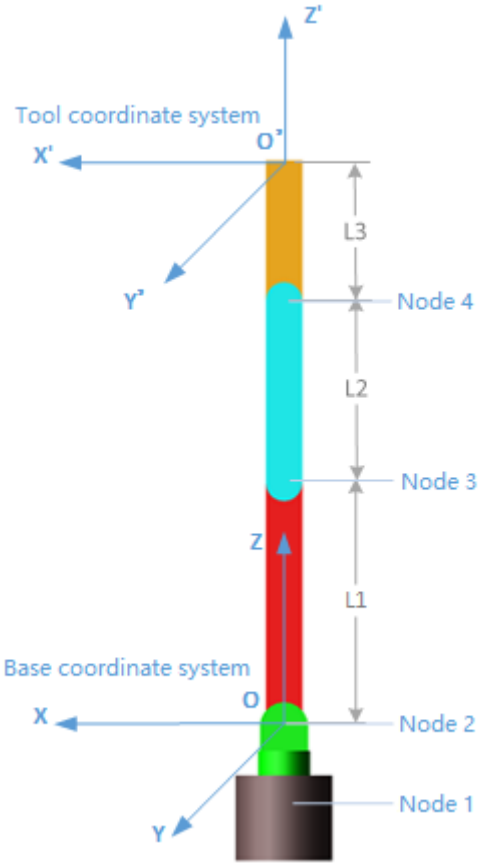


(b)

4-DOF Manipulator Tool Coordinates

(1) Figure (a) shows the diagram of tool coordinates (cylindrical coordinates, $\rho > 0$) and joint coordinates of a 4-DOF manipulator;

(2) Figure (b) shows the diagram of tool coordinates (cylindrical coordinates, ρ < 0) and joint coordinates of a 4-DOF manipulator.



DOF Serial Robot Arm Joint Angle Zero-point Pose

Define the tool coordinate system O'-X'Y'Z', as shown in figures 4-DOF Manipulator_Tool Coordinates and DOF Serial Robot Arm Joint Angle Zero-point Pose.

Define the base coordinate system as a cylindrical coordinate system, and record it as (ρ, θ, z, φ_B) because the tool coordinates are described by cylindrical coordinates. Among them, φ_B is the rotation angle of the end execution tool around the Y' axis, and for the forward direction of rotation, refer to the right-hand rule. θ is the rotation angle of joint 1, and ρ is the polar coordinate radius. To comprehensively describe the pose of the end tool, ρ is expanded here. It can be positive or negative. The corresponding poses are as shown in Figure (a) 4-DOF Manipulator_Tool Coordinates and Figure (b) 4-DOF Manipulator_ Tool Coordinates.

The rotation angles of joints 1, 2, 3, and 4 are called joint coordinates, and recorded as (θ_1, θ_2, θ_3, θ_4) in degrees, as shown in figure 4-DOF Manipulator_Tool Coordinates. The zero point position of each joint angle is shown in the figure DOF Serial Robot Arm Joint Angle Zero-point Pose.

After the back-to-zero operation is completed, as long as the joint angle zero point position has not changed, this instruction can be called at any position without having to adjust it to the position in the DOF Serial Robot Arm Joint Angle Zero-point Pose in the figure.

Please note the following points when using this instruction:

(3) The length unit uses user unit, and the angle unit adopts the degree measure (°);

(4) The zero point position of each joint angle is shown in the figure DOF Serial Robot ArmJoint Angle Zero-point Pose;

(5) HMC_Cancel(AxisID,CancelMode) can be used to cancel this instruction, and the axis number passed in during canceling must be the smallest axis number among the four joint axes;

(6) After the instruction is called successfully, when it is executed for the first time, the tool coordinates will be initialized based on the current joint angle, and then it will move based on this initial pose;

(7) When the instruction is executed for the first time, the RepMode of the joint coordinate axis and the tool coordinate axis will be set to limited stroke. RepMode is not allowed to be modified during the execution of the instruction, or else coordinate transformation errors may occur;

(8) The soft limit value of each joint angle of the mechanism can be set according to the actual situation of the mechanism, and the instruction is automatically canceled after the limit is triggered;

(9) After this instruction is called and the initialization is successful, the user uses tool coordinates (cylindrical coordinates) to program;

(10) It is not allowed to call the HMC_DefPos or HMC_DefOrigin instruction to reset the tool coordinate value, or use the HOME instruction for the tool coordinate axis, or else serious deviations may occur during coordinate transformation. If the user needs to use the self-defined relative user coordinate system, he can perform coordinate offset conversion by himself in the engineering program;

(11) The accessible working space of the mechanism is limited. If the planned path exceeds its accessible working space, the instruction will be canceled for safety reasons. Please plan the working space reasonably according to the mechanism parameters, and ensure that the working space is within the accessible working space of the mechanism (the HMC_Serial4ReverseKinematics instruction can be used to verify whether the planned tool coordinates are valid).

5.6.3.1.2 Parameter

Parameters	Statement	Data Type	Description
arrToolCoordAxisID	INPUT	POINTER TO USINT	The axis number array of the tool coordinate system, consisting of four elements, which correspond to (ρ, Θ, z, φB) in the cylindrical coordinate system. The coordinate axes used are generally virtual axes. The user uses cylindrical coordinate programming in these four axes to plan tool coordinates.
arrJointCoordAxisID	INPUT	POINTER TO USINT	Joint axis number array, with a total of four elements, corresponding to joints 1~4 in sequence. As shown in the figure DOF Serial Robot Arm Joint Angle Zero-point Pose, the physical output axis drives four joints to move.

Parameters	Statement	Data Type	Description
arrLinkLength	INPUT	POINTER TO LREAL	Connecting rod length array, with a total of three elements, corresponding to L1, L2, and L3 as shown in DOF Serial Robot Arm Joint Angle Zero-point Pose
HMC_Serial4Control	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.6.3.1.3 Instruction type

Axis motion cache instructions.

5.6.3.1.4 Application example

The steps to use the instruction are as follows:

(1) Parameter definition

(*Rod length parameter array*)

arrLinkLength[0] := 200; (*L1*)

arrLinkLength[1] := 150; (*L2*)

arrLinkLength[2] := 100; (*L3*)

(*Joint axis number array*)

arrJointCoordAxisID[0] := 0; (*Joint 1*)

arrJointCoordAxisID[1] := 1; (*Joint 2*)

arrJointCoordAxisID[2] := 2; (*Joint 3*)

arrJointCoordAxisID[3] := 3; (*Joint 4*)

(*Tool coordinate axis number array*)

arrToolCoordAxisID[0] := 10;(*Axis number corresponding to the tool coordinate ρ*)

arrToolCoordAxisID[1] := 11;(*Axis number corresponding to the tool coordinate θ*)

arrToolCoordAxisID[2] := 12;(*Axis number corresponding to the tool coordinate z*)

arrToolCoordAxisID[3] := 13;(*Axis number corresponding to the tool coordinate φ_B*)

(2) Define the zero point of each joint axis to be at the position shown in the figure DOF Serial Robot Arm Joint Angle Zero-point Pose(the actual position of the joint axis does not have to be at the zero point)

(3) Send the HMC_Serial4Control instruction to the joint axis

HMC_Serial4Control(arrToolCoordAxisID , arrJointCoordAxisID , arrLinkLength);

(4) Program with the tool coordinate axis in the cylindrical coordinate system

Move to the pose shown in Figure (a) 4-DOF Manipulator_Tool Coordinates, (ρ, Θ, z, φ_B) = (100, 30°, 250, 45°)

HMC_MoveAbs(arrToolCoordAxisID[0], 100); (*Move to ρ = 100*)

HMC_MoveAbs(arrToolCoordAxisID[1], 30); (*Move to θ = 30°*)

HMC_MoveAbs(arrToolCoordAxisID[2], 250); (*Move to z = 250*)

HMC_MoveAbs(arrToolCoordAxisID[3], 45); (*The end tool rotates around the Y' axis, φ_B = 45°*)

Move to the pose shown in Figure (b) 4-DOF Manipulator Tool Coordinates, (ρ, Θ, z, φ_B) = (-100, 30°, 250, 45°)

HMC_MoveAbs(arrToolCoordAxisID[0], -100); (*Move to ρ = -100*)

HMC_MoveAbs(arrToolCoordAxisID[1], 30); (*Move to θ = 30°*)

HMC_MoveAbs(arrToolCoordAxisID[2], 250); (*Move to z = 250*)

HMC_MoveAbs(arrToolCoordAxisID[3], 45); (*The end tool rotates around the Y' axis, φ_B = 45°*)

If a series of intermediate points will be passed through, the spline interpolation instruction can be used.

5.6.3.2 HMC_Serial4Forward Kinematics (Forward Kinematics of 4-DOF Serial Robot Arm)

5.6.3.2.1 Function prototype

UDINT HMC_ Serial4Forward Kinematics(

POINTER TO LREAL arrLinkLength,

POINTER TO LREAL arrToolCoordinate,

POINTER TO LREAL arrJointAngle)

5.6.3.2.2 Description of Functions

This instruction is used to complete the forward kinematics of the 4-DOF serial robot arm (from joint coordinates to tool coordinates).

5.6.3.2.3 Parameter

Parameters	Statement	Data Type	Description
arrLinkLength	INPUT	POINTER TO LREAL	Connecting rod length array, with a total of three elements, corresponding to L1, L2, and L3 in the zero point pose of the joint angle of the DOF SCARA serial robot arm in the figure
arrToolCoordinate	INPUT	POINTER TO LREAL	Tool coordinate array, consisting of four elements (ρ, θ, z, φ B). Save the forward solution results.
arrJointAngle	INPUT	POINTER TO LREAL	Joint coordinate array, with a total of four elements

Parameters	Statement	Data Type	Description
HMC_Serial4ForwardKinematics	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.6.3.3.1 Instruction type

It is a non-axis motion cache instruction.

5.6.3.3 HMC_Serial4ReverseKinematics (Inverse Kinematics of 4-DOF Serial Robot Arm)

5.6.3.3.1 Function

This instruction is used to complete the inverse kinematics of the 4-DOF serial robot arm (from tool coordinates to joint coordinates). When the input tool coordinates exceed the accessible working space of the mechanism, an error is returned.

This function can be used in the following ways:

Check whether the tool coordinates of the key points are within the accessible working space;

Implement joint coordinate space programming.

Joint coordinate programming can avoid many problems in tool coordinate programming, such as the problem that the motion trajectory between key points may pass through the inaccessible working space of the mechanism, and the singular configuration problem.

If only the positioning motion from point to point is required but the intermediate motion trajectories are not required, you can first plan the tool coordinates of the key position points, and then use this instruction to check the validity of the key points and convert the tool coordinates of valid key points into joint coordinates, after which you can program using the joint coordinates.

5.6.3.3.3 Parameter

Parameters	Statement	Data Type	Description
arrLinkLength	INPUT	POINTER TO LREAL	Connecting rod length array, with a total of three elements, corresponding to L1, L2, and L3 in the zero point pose of the joint angle of the DOF SCARA serial robot arm in the figure
arrToolCoordinate	INPUT	POINTER TO LREAL	Tool coordinate array, consisting of four elements (ρ,.,z,φ B)
arrReferenceJointAngle	INPUT	POINTER TO LREAL	Joint coordinate array (reference value), with a total of four elements, which are the rotation angles of joints 1 to 4 in sequence Since there is more than one inverse solution result, a group of solutions that are closer to the reference position is selected as the inverse

Parameters	Statement	Data Type	Description
			solution result.
arrJointAngle	INPUT	POINTER TO LREAL	Joint coordinate array, with a total of four elements to store the inverse kinematics result, which are the rotation angles of joints 1 to 4 in sequence
HMC_Serial4ReverseKinematics	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.6.3.3.4 Instruction type

It is a non-axis motion cache instruction.

5.7 Self-defined Motion Control Function

5.7.1 HMC_SetDposSwitch (Dpos planning switch switching)

5.7.1.1 Function

This instruction is used to change the Dpos planning switch of the specified axis. When the switch is set to 0, Dpos is planned by the controller internal motion control algorithm. When it is 1, Dpos is planned by the user through the HMC_SetDposVal instruction.

When the Dpos planning switch is set to 1, the motion instructions in the motion cache of the axis will no longer be executed and cannot be cleared through HMC_Cancel. Therefore, it is best to clear the motion cache of the axis before the Dpos planning switch is changed to 1, and send no motion instructions to the axis when the switch is set to 1.

5.7.1.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
ucDposSwitch	INPUT	USINT	Dpos planning switch 0: Dpos is planned by the internal motion control algorithm 1: Dpos is planned by the user
HMC_SetDposSwitch	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.7.1.3 Instruction type

It is a non-axis motion cache instruction.

5.7.2 HMC_SetDposVal (Dpos planning)

5.7.2.1 Function P

When the Dpos planning switch is set to 1, this instruction sets the Dpos of the axis to the specified value, and when the Dpos planning switch is set to 0, this instruction does not work.

After this instruction is used to set Dpos, if you want to wait until the set Dpos takes effect before performing the next action, you can call the HMC_SynchroToAlm instruction after it. HMC_SynchroToAlm will have a synchronization operation with the hard real-time task inside controller, so that the instruction will not be exited until the set Dpos value is processed by the hard real-time task (the hard real-time task is executed once per servo cycle).

Therefore, to use HMC_SetDposVal to update Dpos once per servo cycle, you can call HMC_SetDposVal repeatedly in a loop, and then call the HMC_SynchroToAlm instruction, while minimizing the load on the system (close all unused axes (with axis type set to -1) and reduce the number of running IEC tasks as much as possible).

5.7.2.2 Parameter

Parameters	Statement	Data Type	Description
ucAxisID	INPUT	USINT	Axis number
dSetDposVal	INPUT	LREAL	Set Dpos Value
HMC_SetDposVal	OUTPUT	UDINT	0: Success 16#FFFFFFFF: Error

5.7.2.3 Instruction Type

It is a non-axis motion cache instruction.

5.7.2.4 Example

Set the Dpos planning switch to 1, plan Dpos reasonably to synchronize the positions of axes 0 and 1, and make the speed curve meet the sinusoidal law. The code is as below:

(1) Initialization, task 1.

Program:

```
(*Close unused axes*)
FOR AxisID:=2 TO 15 BY 1 DO
HMC_SetAxisType(AxisID, - 1);
```

END_FOR

HMC_SetAxisType(0, 0); (*Axis 0 is a virtual axis*)

HMC_SetAxisType(1, 0); (*Axis 1 is a virtual axis*)

HMC_SetDposSwitch(0, 1); (*Set the Dpos planning switch of axis 0 to 1*)

HMC_SetDposSwitch(1, 1); (*Set the Dpos planning switch of axis 1 to 1*)

HMC_DefPos(0,0); (*Axis 0 position initialization*)

HMC_DefPos(1, 0); (*Axis 0 position initialization*)

(2) Dpos planning, task 2.

Program:

Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
Dpos			LREAL	0	FALSE	Not Public	
count			DINT	0	FALSE	Not Public	
MaxNum		Number of servo cycles running	DINT	500	FALSE	Not Public	
PI			LREAL	3.1415926	FALSE	Not Public	
A		Dpos amplitude	LREAL	1000	FALSE	Not Public	

count := 0;

WHILE TRUE DO

count := count + 1;

Dpos := A * (1 - COS(2*PI*count/MaxNum)); (*Target position calculation*)

HMC_SetDposVal(0, Dpos);(*Axis 0 Dpos planning*)

HMC_SetDposVal(1, Dpos);(*Axis 1 Dpos planning*)

HMC_SynchroToAlm (); (*Synchronize to algorithmic hard real-time tasks*)

IF count = MaxNum THEN (*Control the number of runs. If the number is not required to control, it can also loop forever*)

EXIT;

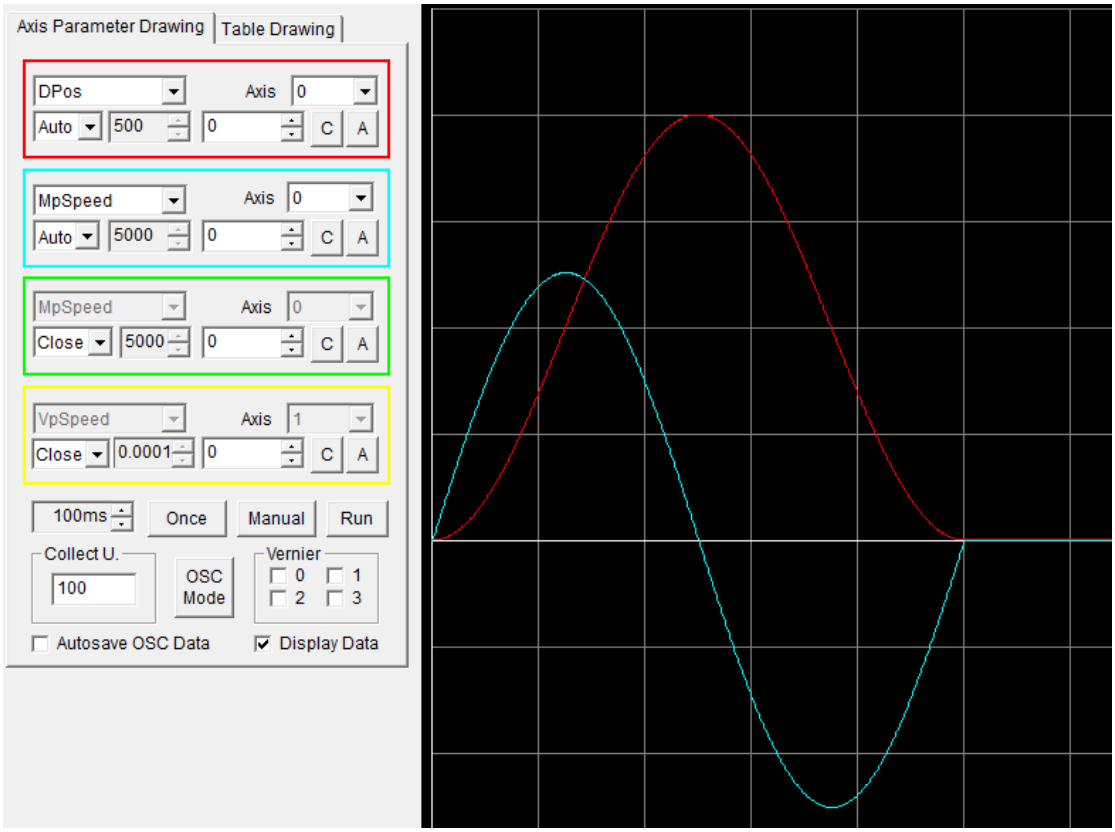
END_IF

END_WHILE

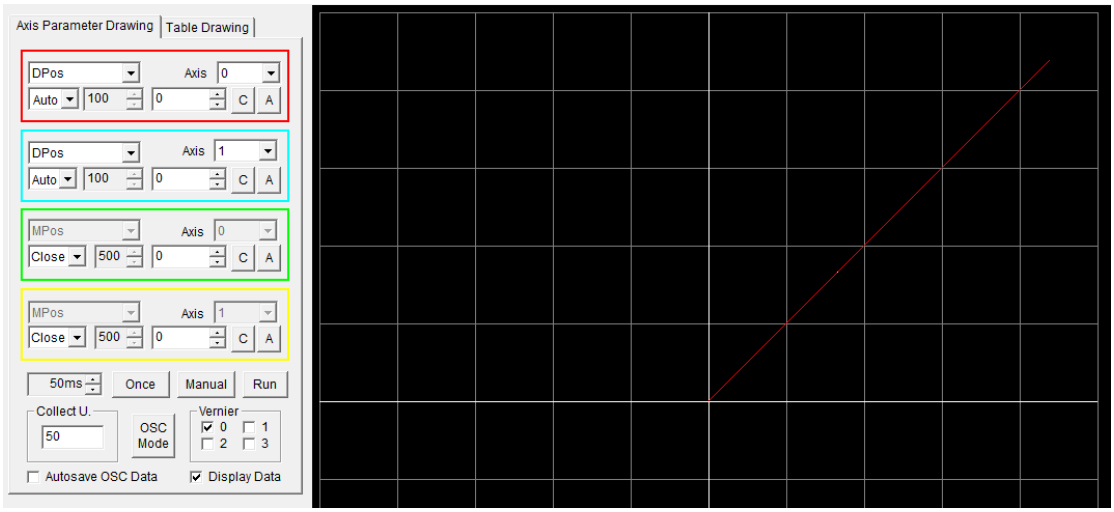
First, run the initialization task 1, and then enter the following instruction in the terminal:

HMC_TriggerScope();TaskStart(2);

The running results are as follows (servo cycle: 1 ms):



Dpos and MpSpeed of Axis 0



Combined Motion Trajectory of Axis 0 and Axis 1

5.7.3 HMC_SynchroToAlm (Synchronizing to Algorithm Tasks)

5.7.3.1.1 Function prototype

UDINT HMC_SynchroToAlm ()

5.7.3.1.2 Description of Functions

This instruction can be used to synchronize hard real-time tasks with algorithms inside the LX. After this

instruction is called, it will wait until the hard real-time task is executed once before exiting (the hard real-time task is executed once per servo cycle).

5.7.3.1.3 Parameter

Parameter	Statement	Data Type	Description
0	OUTPUT	UDINT	Success

5.7.3.1.4 Instruction type

It is a non-axis motion cache instruction.

5.8 Axis Parameter Setting Instruction

5.8.1 HMC_SetAxisType (Setting Axis Type)

5.8.1.1 Function

It is used to set the axis type of the specified axis. For details about the meaning of each axis type, see the axis parameter AxisType.

1. Ranges of axis numbers 0 to 63.

2. When there are instructions in the axis instruction cache of the axis, setting the axis type by calling this function will fail.

3. After the axis type is successfully set, the system will automatically modify the axis parameter RepMode according to different axis types.

When the user sets the axis type to a virtual axis or bus encoder axis, the system will set RepMode=2; when the user sets it to other axis types, the system will set RepMode=0.

5.8.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
EmAxisType	Input	EMAXISTYPE	Axis type
HMC_SetAxisType	Output	UDINT	0: Set successfully Other values: Error code

5.8.1.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.1.4 Example

HMC_SetAxisType(0,0); (*Set axis 0 as a virtual axis*)

HMC_SetAxisType(1,50); (*Set axis 1 as an EtherCAT_Position axis*)

HMC_SetAxisType(60,-1); (*Set axis 60 as an unused axis*)

5.8.2 HMC_HwLimitConfig (Setting Hard Limit Switch)

5.8.2.1 Function

It is used to set the hard limit switch of the specified axis. The limit function is only valid for limited-stroke axes.

5.8.2.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucLimitType	Input	USINT	0: Set forward hard limit (normally open) 1: Set reverse hard limit (normally open) 2: Set forward hard limit (normally closed) 3: Set reverse hard limit (normally closed)
sDiChan	Input	UDINT	Associated DI channel (0-2097151) When sDiChan is -1, it means the hard limit switch is canceled, and it is only valid for limited-stroke axes. The specified axis will no longer be associated with the original forward (reverse) limit DI channel, and will be released immediately even if it is currently in the limit status.
HMC_HwLimitConfig	Output	UDINT	0: Set successfully Other values: Error code

5.8.2.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.2.4 Example

Configure the fifth DI as the forward limit switch of axis 0, and the sixth DI as the reverse limit switch of axis 0.

HMC_HwLimitConfig (0 ,0, 5);

HMC_HwLimitConfig (0 , 1, 6);

5.8.3 HMC_SetUnits (Setting User Units)

5.8.3.1 Function

It is Units, an axis parameter to set the unit conversion factor of the specified axis. This parameter can only be positive values in the line/user unit. "Line" refers to the scale unit corresponding to one pulse of the axis.

By setting the axis parameter Units, the count value or pulse output value fed back by the encoder can be converted into a user-friendly unit value, such as mm, angle, etc.

5.8.3.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
d Units	Input	LREAL	User unit, which cannot be 0 or negative
HMC_SetUnits	Output	UDINT	0: Set successfully, Other values: Error code

5.8.3.3 Instruction type

It is a non-axis motion cache instruction.

5.8.3.4 Example

For example, if an axis makes exactly one revolution every 10000 pulses, that is, one revolution corresponds to 10000 lines. Based on the design of the mechanical structure, if the axis drives the terminal to move 10 mm for each revolution (10000 lines), then the following two methods are equivalent, and both of them can make the terminal move 50 mm (that is, the motor rotates 5 times):

Method 1:

HMC_SetUnits(0, 1000); (*10000 lines/10 mm = 1000 lines/mm*)

HMC_Move(0,50); (/*The terminal runs for 50 mm*/)

Method 2:

HMC_SetUnits(0, 1);

HMC_Move(0,50000); (/*The motor runs for 50000 lines and the terminal runs for 50 mm*/)

Which one to choose among the above two methods depends on the user's usage habits.

5.8.4 HMC_SetJerkMode (Setting Trapezoidal/S-Shaped Acceleration)

5.8.4.1 Function

Set JerkMode. When JerkMode is 0, the speed curve of the motion process is a trapezoidal acceleration and deceleration curve, and when JerkMode is 1, it is an S-shaped acceleration and deceleration curve.

5.8.4.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucJerkMode	Input	USINT	JerkMode (default value: 0). 0: The speed curve of the motion process is a trapezoidal acceleration and deceleration curve; 1: The speed curve of the motion process is a S-shaped acceleration and deceleration curve.
HMC_SetJerkMode	Output	UDINT	0: Set successfully Other values: Error code

5.8.4.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.5 HMC_SetKe (Setting Ke)

5.8.5.1 Function

It is used to set the numerator (axis parameter KeNumerator) and denominator (axis parameter KeDenominator) of the axis parameter Ke. Refer to the description of the axis parameters KeNumerator and KeDenominator.

5.8.5.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
iKeNumerator	Input	DINT	Ke numerator
iKeDenominator	Input	DINT	Ke denominator
HMC_SetKe	Output	UDINT	0: Set successfully Other values: Error code

5.8.5.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.5.4 Additional Information

Ke can be set to a negative value by inverting the actual feedback speed and position direction (some earlier

firmware versions do not support this).

Ke can not be 0.

5.8.5.5 Example

The rotary table is driven by a motor to control rotation, and an encoder is connected to the rotary table to measure its rotation angle.

When the rotary table rotates once, the encoder outputs 8192 pulses. When it is expected to display the rotation angle value of the rotary table directly on the software, since 8192/360 is not divisible, to achieve the highest accuracy, this can be achieved by setting Ke.

HMC_SetAxisType(0,50);

HMC_SetUnits(0, 1);

HMC_SetKe(0,360,8192); (*In conjunction with Units, make the axis unit in angle, and Mpos directly displays the angle*)

5.8.5.6 Precautions for Use

This parameter is only effective for bus axes. This parameter has a similar function as Units. (Ke/Units) together constitute the "unit conversion factor" of the bus axis, and its unit is the "lines/user unit". See Units (Unit conversion factor).

5.8.6 HMC_SetKs (Setting Ks)

5.8.6.1 Function

It is used to set the numerator (axis parameter KsNumerator) and denominator (axis parameter KsDenominator) of the axis parameter step output ratio Ks. Refer to the description of the axis parameters KsNumerator and KsDenominator.

5.8.6.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
iKsNumerator	Input	DINT	Ks numerator
iKsDenominator	Input	DINT	Ks denominator
HMC_SetKs	Output	UDINT	0: Set successfully Other values: Error code

5.8.6.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.6.4 Additional information

Ks shall be >0.

5.8.6.5 Example

The rotary table is controlled by a stepper motor. The stepper motor outputs 8192 pulses to drive the rotary table to rotate once. When it is expected to display the rotation angle value of the rotary table directly on the software, since 8192/360 is not divisible, to achieve the highest accuracy, this can be achieved by setting Ks.

HMC_SetAxisType(0, 50);

HMC_SetUnits(0, 1);

HMC_SetKs(0,8192,360);

HMC_Move(0, 180); (*Drive the rotary table to rotate for 180 degrees*)

5.8.7 HMC_SetFeedBackSrcAddr (Setting User-Fefined Feedback Input Address)

5.8.7.1 Function

Configure the user-defined input source for the actual axis position Mpos. When the axis parameter FeedBackSrcMode=1, the actual position Mpos of the axis is incrementally calculated and updated by the user-defined input source. At this time, Mpos has no relationship with the actual physical input of the axis.

When the axis Mpos is calculated from a user-defined input source, Mpos is calculated cumulatively based on the change increment of the self-defined input source, and the axis parameter Units also participate in the Mpos calculation.

5.8.7.2 Parameter

Parameter	Statement	Data Type	Description
ucAxis	Input	USINT	Axis number
pAddr	Input	POINTER TO LREAL	User-defined variable address
DataType	Input	USINT	Data type of user-defined variables: USINT= 1, SINT=2, UINT=3, INT=4, UDINT=5, DINT=6, F32=7, LREAL=8
HMC_SetFeedBackSrcAddr	Output	UDINT	0: Set successfully Other values: Error code

5.8.7.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.7.4 Example

There is a position sensor whose output signal is a voltage signal. The position to be measured is between 0 and 100000, and the corresponding position sensor outputs a voltage signal between 0 and 10 V. Connect the voltage signal output by the position sensor to the first AI. Calculate this position information in real time using axis 0.

HMC_SetAxisType (0,50);

axis0.FeedBackSrcMode := 1;

HMC_SetUnits(0, 10.0/ 100000.0); (*Set the position and voltage signal conversion coefficient as units*)

HMC_SetFeedBackSrcAddr(0, ADR(AI0), 4); (*Set the first AI as the Mpos calculation source of axis 0*)

HMC_Defpos(0, AI0*(100000/ 10)); (*Initialize the position of axis 0 to the currently measured absolute position, and then incrementally calculate the position value on this basis*)

5.8.7.5 Precautions for Use

This function is only valid for axis types with actual physical input, such as bus axes.

5.8.8 HMC_SetOutDstAddr (Setting User-Defined Output Address)

5.8.8.1 Function

It is to set the self-defined output address of the position closed-loop output value DACOut. In position closed-loop mode, the calculated output value DACOut is output to the physical output channel of the axis. For example, in servo 20 mode, it is output to the AO channel corresponding to the axis; in bus speed mode, it is output to the bus output data area corresponding to the axis. But when the axis parameter OutDstMode=1, the DACOut value stops being output to the physical output channel of the axis, but is output to the self-defined output address of the axis.

When OutDstMode=1, for the bus speed mode axis, the algorithm will immediately output speed 0 and stop the corresponding drive.

5.8.8.2 Parameter

Parameter	Statement	Data Type	Description
ucAxis	Input	USINT	Axis number
pAddr	Input	POINTER TO DINT	User-defined variable address
DataType	Input	USINT	Data type of user-defined variables: USINT= 1, SINT=2, UINT=3, INT=4,

Parameter	Statement	Data Type	Description
			UDINT=5, DINT=6, F32=7, LREAL=8
HMC_SetOutDstAddr	Output	UDINT	0: Set successfully Other values: Error code

5.8.8.3 Instruction Type

It is a non-axis motion cache instruction.

5.8.8.4 Example

Set the output target of bus axis 0 to the system variable diOutPut.

HMC_SetOutDstAddr(0, ADR(diOutPut), 6);

5.8.8.5 Precautions for Use

This function is only valid for axis types in position closed-loop mode, such as bus axes and bus speed mode.

5.8.9 HMC_SetCmd BufferLoad Mode (Setting Axis Instruction Loading Mode)

5.8.9.1 Function

Set the axis instruction loading mode. It is mainly used for auxiliary processing in tangential tracking corner retract mode.

5.8.9.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucCmdLoad Mode	Input	USINT	0: Normal loading mode; 1: After the current instruction is completed, the next instruction will not be loaded
HMC_SetCmdBufferLoad Mode	Output	UDINT	0: Set successfully Other values: Error code

5.8.9.3 Instruction type

Non-axis motion cache instruction

5.8.9.4 Precautions for Use

When the tangential tracking instruction is in the corner paused retract status, if you use HMC_Cancel to cancel the instruction, because the instructions in the XY instruction queue are in the paused status at this time, for safety reasons, the instructions in the X-axis and Y-axis will not be automatically restored after the cancellation is completed. To resume the execution of subsequent instructions in the XY instruction queue, the user needs to call HMC_SetCmdBufferLoad Mode to set the instruction loading mode to normal loading mode (0).

5.8.10 HMC_GetCmd BufferLoad Mode (Getting Axis Instruction Loading Mode)

5.8.10.1 Function

Get the axis instruction loading mode.

5.8.10.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_GetCmdBufferLoad Mode	Output	USINT	0: Normal loading mode 1: After the current instruction is completed, the next instruction will not be loaded 255: Error code

5.8.10.3 Instruction type

It is a non-axis motion cache instruction.

5.8.11 HMC_SetCmdStopMaxTime (Setting the Maximum Instruction End Waiting Time)

5.8.11.1 Function

Set the maximum instruction end waiting time.

5.8.11.2 Parameter

Parameter	Statement	Data Type	Description
uiTime	Input	UDINT	Time (ms)
HMC_SetCmdStopMaxTime	Output	UDINT	0: Succeeded Other values: Failed

5.8.11.3 Instruction type

It is a non-axis motion cache instruction.

5.8.12 HMC_GetAxisSlaveAddr (Getting the Station Address Corresponding to the Axis)

5.8.12.1 Function

It is used to obtain the station address corresponding to the specified axis number. The input is the axis number of the axis to obtain the address. The axis number must be consistent with the axis number of the actual configured device when setting. The output is the station number corresponding to the specified axis. If the axis number is set to an axis number not configured or exceeds the maximum axis number range (0~63), the output is 0xFFFF.

5.8.12.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucAxisID	Input	USINT	M, G or constants	Axis number
HMC_GetAxisSlaveAddr	Output	UDINT	M or G	Station number of axis

5.8.12.3 Instruction type

It is a non-axis motion cache instruction.

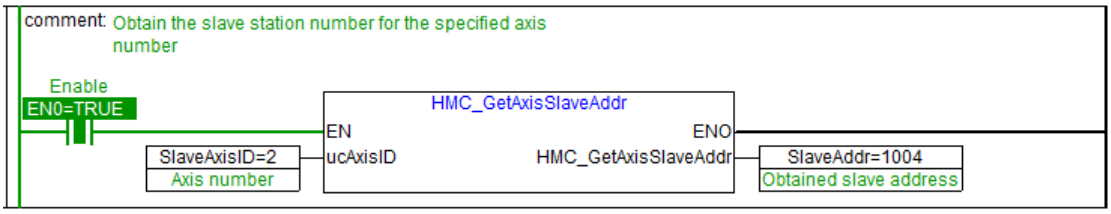
5.8.12.4 Example

The following example uses the HMC_GetAxisSlaveAddr instruction to obtain the address with axis 2. When EN is enabled, this instruction starts the function of obtaining the station address of axis and returns the obtained station address result to the SlaveAddr variable. The obtained station address of axis 2 is 1004, which is consistent with the actual configured address.

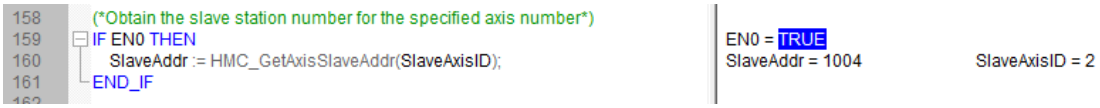
Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN0		Enable	BOOL	TRUE	FALSE	Not Public	☐
SlaveAddr		Obtained slave address	UINT	1004	FALSE	Not Public	☐

LD program implementation:



ST program implementation:



5.8.12.5 Precautions for Use

The input axis number must be set according to the actual configuration to obtain the correct slave station address.

5.8.13 HMC_GetAxisTorque (Getting Axis Torque)

5.8.13.1 Function

It is used to get the actual torque value of the device corresponding to the specified axis number. The input is the axis number of the axis for which the torque value is to be obtained. The axis number must be consistent with the axis number of the actual configured device when setting. The output is the actual torque value of the corresponding device, namely, the value of EtherCAT object dictionary 0x6077 (Torque actual value) in one thousandth of the rated torque. If the set axis number exceeds the maximum value range (0~63) or there is an error in obtaining the torque value of the slave station, the output is 0.

5.8.13.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisTorque	Output	INT	Actual torque value in one thousandth of the rated torque

5.8.13.3 Instruction type

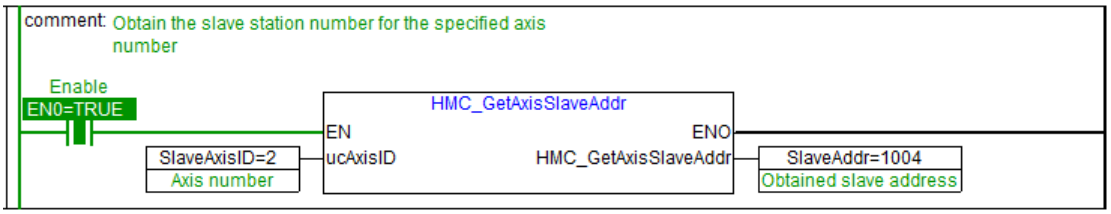
It is a non-axis motion cache instruction

5.8.13.4 Example

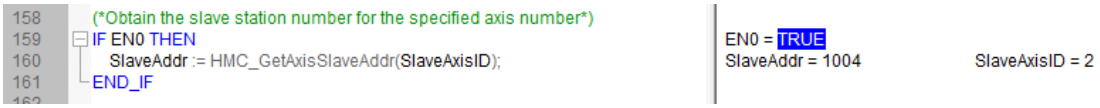
The following example uses the HMC_GetAxisTorque instruction to obtain the actual torque value of the axis 2. When EN is enabled, this instruction starts the function of obtaining the torque and returns the obtained actual

torque value to the ActualTorque variable. The obtained torque value is consistent with the value object dictionary 0x6077 (Torque actual value).

LD program implementation:



ST implementation program:



5.8.13.5 Precautions for Use

The input axis number must be set according to the actual configuration to obtain the corresponding correct actual torque value of the slave station.

5.8.14 HMC_AxisPotInput (Getting Axis Forward Hard Limit Status)

5.8.14.1 Function

It is used to obtain the axis forward hard limit status corresponding to the specified axis number. The input is the axis number of the axis to be obtained. The axis number must be consistent with the actual configured axis number when setting. The output is the current forward hard limit status of the corresponding axis number. When the status is 1, it means that the current position has reached the forward hard limit point. When the status is 0, it means that the current position has not reached the forward hard limit point. If the set axis number exceeds the maximum range (0~63), the output is 0.

The obtained forward hard limit status has the same meaning as bit0 of the axis parameter LimitState (over-limit status).

5.8.14.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucAxisID	Input	BYTE	M, G or constants	Axis number
HMC_AxisPotInput	Output	BOOL	M or G	Axis forward hard limit status. 1: It reaches the forward hard limit. 0: It does not reach

Parameter	Statement	Data Type	Storage Area	Description
				the forward hard limit or the axis number is incorrect

5.8.14.3 Instruction type

It is a non-axis motion cache instruction.

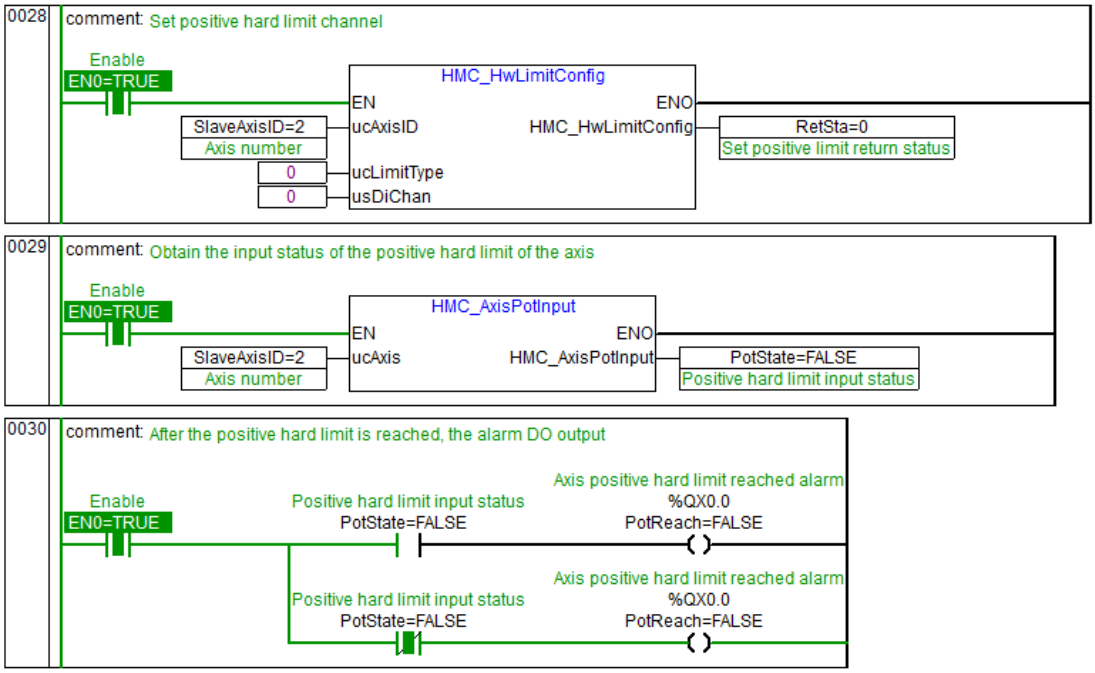
5.8.14.4 Example

The following example uses the HMC_AxisPotInput instruction to obtain the forward hard limit status of the axis 2. Before obtaining it, you need to specify the forward hard limit channel. The example uses the HMC_HwLimitConfig instruction to set the axis forward hard limit channel number to 0. When the axis reaches the forward hard limit point through forward motion, the HMC_AxisPotInput instruction obtains that the forward hard limit status PotState is 1, and an alarm DO is output.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
(*Set positive hard limit channel*)
IF EN0 THEN
    HMC_HwLimitConfig(SlaveAxisID, 0, 0);
END_IF

(*After obtaining the input status of the positive hard limit of the axis, the alarm DO output is triggered w
IF EN0 THEN
    PotState := HMC_AxisPotInput(SlaveAxisID);

    IF PotState THEN
        PotReach := TRUE;
    ELSE
        PotReach := FALSE;
    END_IF

END_IF
```

```
EN0 = TRUE
SlaveAxisID = 2

EN0 = TRUE
PotState = FALSE                SlaveAxisID = 2

PotState = FALSE
PotReach = FALSE

PotReach = FALSE
```

5.8.14.5 Precautions for Use

The forward hard limit is only valid when it is of limited stroke. That is, it is valid when the axis parameter RepMode=0. The forward hard limit channel number also needs to be set.

5.8.15 HMC_AxisNotInput (Getting Axis Reverse Hard Limit Status)

5.8.15.1 Function

It is used to obtain the axis reverse hard limit status corresponding to the specified axis number. The input is the axis number of the axis to be obtained. The axis number must be consistent with the actual configured axis number when setting. The output is the current reverse hard limit status of the corresponding axis number. When the status is 1, it means that the current position has reached the reverse hard limit point. When the status is 0, it means that the current position has not reached the reverse hard limit point. If the set axis number exceeds the maximum range (0~63), the output is 0. The obtained reverse hard limit status has the same meaning as bit1 of the axis parameter LimitState (over-limit status).

5.8.15.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_AxisNotInput	Output	BOOL	Axis reverse hard limit status. 1: It reaches the reverse hard limit. 0: It does not reach the reverse hard limit or the axis number is incorrect

5.8.15.3 Instruction type

It is a non-axis motion cache instruction.

5.8.15.5 Example

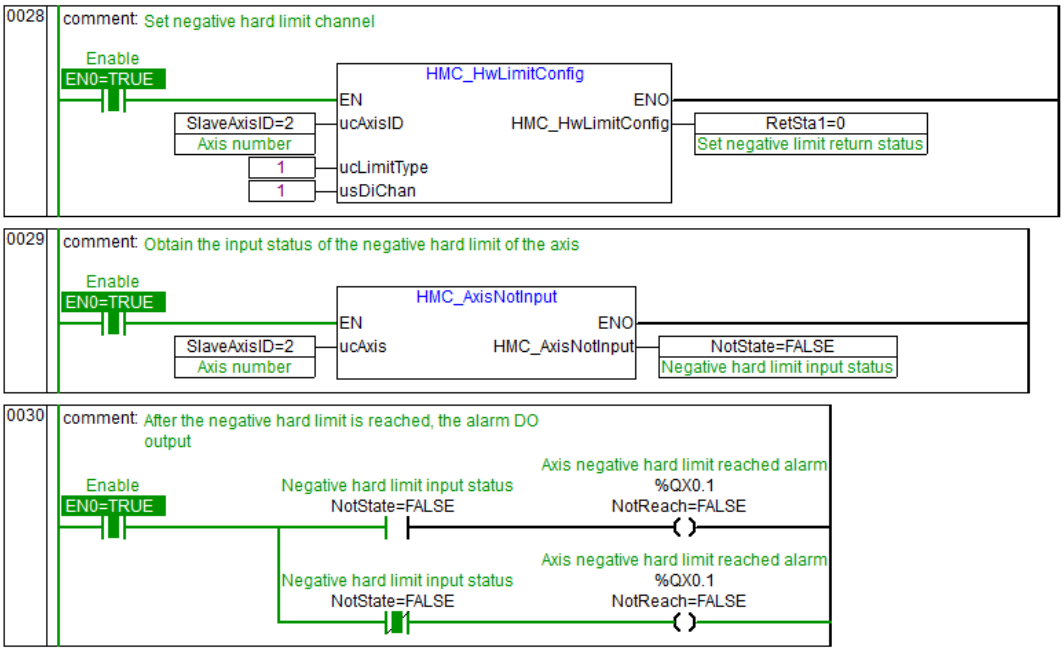
The following example uses the HMC_AxisNotInput instruction to obtain the reverse hard limit status of the axis 2. Before obtaining it, you need to specify the reverse hard limit channel. The example uses the

HMC_HwLimitConfig instruction to set the axis reverse hard limit channel number to 1. When the axis reaches the reverse hard limit point through reverse motion, the HMC_AxisNotInput instruction obtains that the reverse hard limit status NotState is 1, and an alarm DO is output.

Variable definition:

PotState		Positive hard limit input status	BOOL	FALSE	FALSE	Not Public	
NotState		Negative hard limit input status	BOOL	FALSE	FALSE	Not Public	
PotReach	%QX0.0	Axis positive hard limit reached alarm	BOOL	FALSE	FALSE	Not Public	
NotReach	%QX0.1	Axis negative hard limit reached alarm	BOOL	FALSE	FALSE	Not Public	
RetSta1		Set negative limit return status	UDINT	0	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
(*Set negative hard limit channel*)
IF EN0 THEN
    HMC_HwLimitConfig(SlaveAxisID, 1, 1);
END_IF

(*After obtaining the input status of the negative hard limit of the axis, the alarm [
IF EN0 THEN
    NotState := HMC_AxisNotInput(SlaveAxisID);

    IF NotState THEN
        NotReach := TRUE;
    ELSE
        NotReach := FALSE;
    END_IF

END_IF
```

EN0 = TRUE
SlaveAxisID = 2

EN0 = TRUE
NotState = FALSE
SlaveAxisID = 2

NotState = FALSE
NotReach = FALSE

NotReach = FALSE

5.8.15.5 Precautions for Use

The reverse hard limit is only valid when it is of limited stroke. That is, it is valid when the axis parameter RepMode=0. The reverse hard limit channel number also needs to be set.

5.8.16 HMC_AxisStandStillStatus (Getting Axis Stop Status)

5.8.16.1 Function

It is used to check whether the specified axis number is in the stop status. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is whether the corresponding axis is currently in the stop status. When the status is 1, it means that the current axis is in the stop status, that is, there is no motion control instruction for the current axis. When the status is 0, it means that the current axis is in a non-stop status. If the set axis number exceeds the maximum value range (0~63), the output is also 0.

5.8.16.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucAxisID	Input	BYTE	M, G or constants	Axis number
HMC_AxisStandStillStatus	Output	BOOL	M or G	Current stop status of the slave station. 1: in stop status 0: in non-stop status

5.8.16.3 Instruction type

It is a non-axis motion cache instruction.

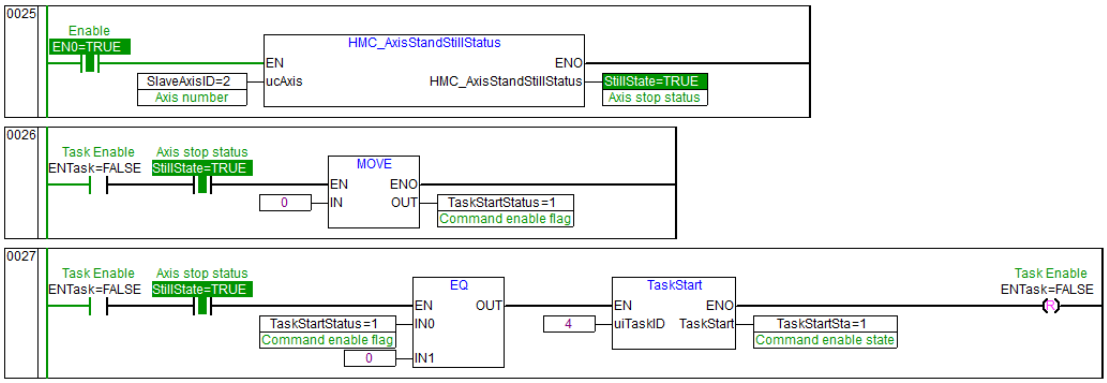
5.8.16.4 Example

The following example uses the HMC_AxisStandStillStatus instruction to obtain the axis stop status of the axis 2. When EN is enabled, this instruction starts the function of obtaining the axis stop status and returns the obtained status result to the StillState variable. This example calls TaskStart to start task 4 when it is determined that the stop status of axis 2 is 1, that is, when there is currently no motion control instruction for the axis. After task 4 runs the motion control instruction, the StillState status obtained through the HMC_AxisStandStillStatus instruction is 0.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN0		Enable	BOOL	TRUE	FALSE	Not Public	☐

LD program implementation:



ST program implementation:

```
IF EN0 THEN
    StillState := HMC_AxisStandStillStatus(SlaveAxisID); (*Obtain axis stop status
    IF ENTask AND StillState THEN
        TaskStartStatus := 0; (*Set task start flag*)
    END_IF
    IF ENTask AND TaskStartStatus = 0 THEN
        TaskStartSta := TaskStart(4); (*Start the task and issue instructions*)
        ENTask := FALSE;
    END_IF
END_IF
```

EN0 = TRUE
StillState = TRUE
SlaveAxisID = 2
ENTask = FALSE
StillState = TRUE
TaskStartStatus = 1
ENTask = FALSE
TaskStartStatus = 1
TaskStartSta = 1
ENTask = FALSE

5.8.17 HMC_AxisSlaveErrorStatus (Getting Axis Slave Error Status)

5.8.17.1 Function

It is used to get the error status of the associated slave station of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the current error status of the corresponding axis. When the status is 1, it means that the current axis is in the error status. When the status is 0, it means that the current axis is normal. If the set axis number exceeds the maximum value range (0~63), the output is also 0.

5.8.17.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_AxisSlaveErrorStatus	Output	BOOL	Current error status of the slave station. 1: Error 0: No error

5.8.17.3 Instruction type

It is a non-axis motion cache instruction.

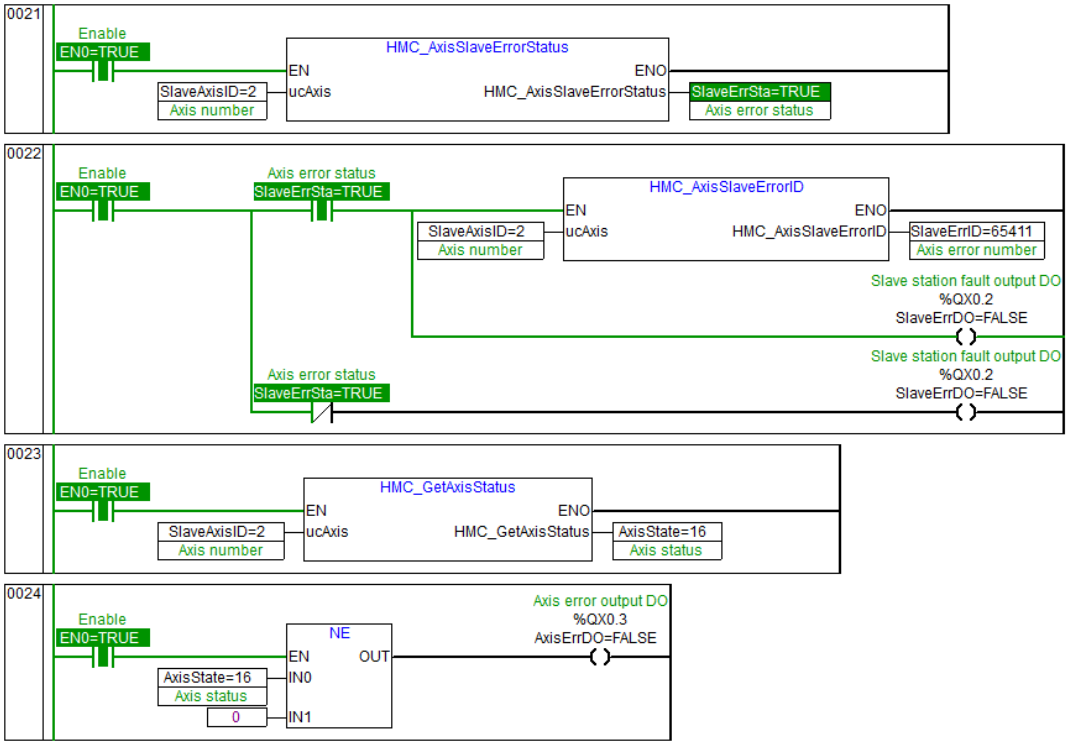
5.8.17.4 Example

The following example uses relevant instructions to obtain the error status, error ID, and axis status of the slave station with axis number 2. This example uses an Omron servo as the slave station, and the slave station axis number is set to 2. When the EtherCAT communication network cable is unplugged and plugged during normal EtherCAT communication, the servo reports fault 0x83. At this time, the slave station error status obtained through the HMC_AxisSlaveErrorStatus instruction is 1. The slave error code obtained through the HMC_AxisSlaveErrorID instruction is 65411 (0xFF83), which is consistent with the fault code reported by the servo. The axis status obtained through HMC_GetAxisStatus is 16, that is, the bus slave station reports an error. After a fault is reported, the corresponding DO output is set.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
SlaveErrSta		Axis error status	BOOL	TRUE	FALSE	Not Public	
SlaveErrID		Axis error number	UDINT	65411	FALSE	Not Public	
SlaveErrDO	%QX0.2	Slave station fault output DO	BOOL	FALSE	FALSE	Not Public	
AxisState		Axis status	UDINT	16	FALSE	Not Public	
AxisErrDO	%QX0.3	Axis error output DO	BOOL	FALSE	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
(*Obtain slave station error status, error code, and axis status*)
IF EN0 THEN
    SlaveErrSta := HMC_AxisSlaveErrorStatus(SlaveAxisID);

    IF SlaveErrSta THEN
        SlaveErrID := HMC_AxisSlaveErrorID(SlaveAxisID);
        SlaveErrDO := TRUE;
    ELSE
        SlaveErrDO := FALSE;
    END_IF

    AxisState := HMC_GetAxisStatus(SlaveAxisID);

    IF AxisErrDO <>0 THEN
        AxisErrDO := TRUE;
    ELSE
        AxisErrDO := FALSE;
    END_IF

END_IF
```

```
EN0 = TRUE
SlaveErrSta = TRUE          SlaveAxisID = 2

SlaveErrSta = TRUE
SlaveErrID = 65411          SlaveAxisID = 2
SlaveErrDO = FALSE

SlaveErrDO = FALSE

AxisState = 16              SlaveAxisID = 2

AxisErrDO = FALSE
AxisErrDO = FALSE

AxisErrDO = FALSE
```

5.8.18 HMC_AxisSlaveErrorID (Getting Axis Slave Error ID)

5.8.18.1 Function

It is used to get the error number of the slave station associated with the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the current error code of the corresponding axis. When the axis is in an error status, the actual error code of the slave station corresponding to the current axis is returned. If the set axis number exceeds the maximum value range (0~63), the output is 0. When an error occurs in obtaining the error number of the axis-associated slave station, 0xFFFFFFFF is returned.

For specific error code information, please refer to the fault description in the corresponding axis manual.

5.8.18.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_AxisSlaveErrorID	Output	UDINT	The current error number of the slave station

5.8.18.3 Instruction type

It is a non-axis motion cache instruction.

5.8.18.4 Example

Refer to the example in HMC_AxisSlaveErrorStatus(Getting axis slave error status).

5.8.19 HMC_GetAxisStatus (Getting Axis Parameter AxisStatus)

5.8.19.1 Function

It is used to get the axis parameter AxisStatus (axis status) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the axis status information of the corresponding axis. If the set axis number exceeds the maximum value range (0~63), the output is 0.

The obtained axis status information is consistent with the axis parameter AxisStatus (axis status). For details, please refer to the description in the axis parameters chapter.

5.8.19.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucAxisID	Input	BYTE	M, G or constants	Axis number
HMC_GetAxisStatus	Output	UDINT	M or G	Axis parameter AxisStatus (axis status) value

5.8.19.3 Instruction type

It is a non-axis motion cache instruction.

5.8.19.4 Example

Refer to the example in HMC_AxisSlaveErrorStatus(Getting axis slave error status).

5.8.20 HMC_GetAxisMcmdType (Getting Axis Parameter McmdType)

5.8.20.1 Function

It is used to get the current instruction type of the specified axis number. The input is the specified axis number to obtain the axis parameters. The axis number must be consistent with the actual configured axis number when setting. The output is the current instruction type of the corresponding axis (axis parameter McmdType). If the set axis number exceeds the maximum value range (0~63) or the current axis instruction is empty, the output is 0.

The obtained axis status information is consistent with the axis parameter McmdType (current instruction type). For details, please refer to the description in the axis parameters chapter.

5.8.20.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisMcmdType	Output	UDINT	Axis parameter McmdType (current instruction type) value

5.8.20.3 Instruction type

It is a non-axis motion cache instruction.

5.8.20.4 Example

The following example uses the HMC_GetAxisMcmdType instruction to obtain the current instruction type of axis 2. When EN is enabled, this instruction starts the function of obtaining the current instruction type and returns the obtained instruction type result to the CCmdType variable. The current instruction type obtained after execution is 10:HMC_MOVE, which is consistent with the actual configured logical configuration.

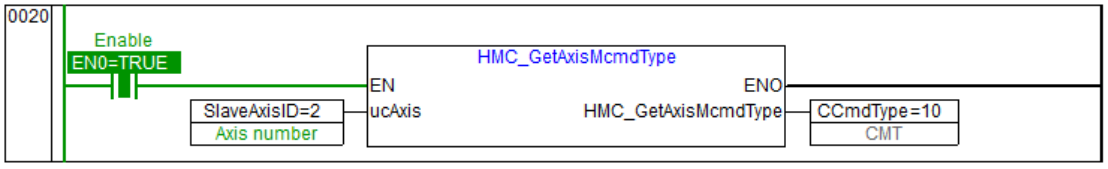
Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
Basic									
AxisID	USINT	0	1	2	3	4	5	6	7
VarName	STRING	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
AxisType	EmAxisType	Unused	Unused	EtherCAT_...	Unused	Unused	Unused	Unused	Unused
Units	LREAL	1	1	1	1	1	1	1	1
Status									
MCmdType	EmCmdType	HMC_McIdle	HMC_McIdle	HMC_Move	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle
NCmdType	EmCmdType	HMC_McIdle	HMC_McIdle	HMC_Forward	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle
AxisStatus	UDINT	0	0	0	0	0	0	0	0
AxisErrMask	UDINT	0	0	0	0	0	0	0	0
LimitState	USINT	0	0	0	0	0	0	0	0

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	

CCmdType			UDINT	10	FALSE	Not Public	
NCmdType			UDINT	18	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
93  (*Get the current instruction type for the specified axis*)
94  IF EN0 THEN
95      CCmdType := HMC_GetAxisMcmdType(SlaveAxisID);
96  END_IF
97
```

EN0 = TRUE
CCmdType = 10 SlaveAxisID = 2

5.8.21 HMC_GetAxisNcmdType (Getting Axis Parameter NcmdType)

5.8.21.1 Function

It is used to get the next instruction type of the specified axis number. The input is the specified axis number to obtain the axis parameters. The axis number must be consistent with the actual configured axis number when

setting. The output is the next instruction type of the corresponding axis (axis parameter NcmdType). If the set axis number exceeds the maximum value range (0~63) or the current axis instruction is empty, the output is 0.

The obtained axis status information is consistent with the axis parameter NcmdType (next instruction type). For details, please refer to the description in the axis parameters chapter.

5.8.21.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisNcmdType	Output	UDINT	Axis parameter NcmdType (next instruction type) value

5.8.21.3 Instruction type

It is a non-axis motion cache instruction.

5.8.21.4 Example

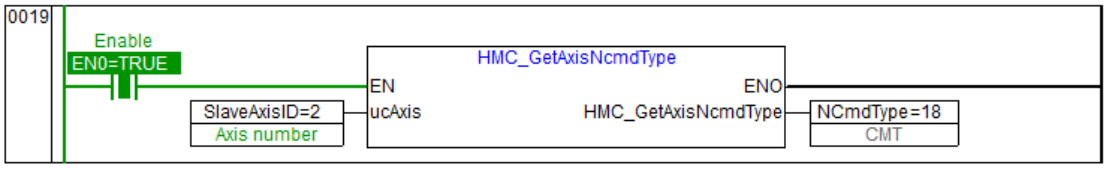
The following example uses the HMC_GetAxisNcmdType instruction to obtain the next instruction type of axis 2. When EN is enabled, this instruction starts the function of obtaining the next instruction type and returns the obtained instruction type result to the NCmdType variable. The next instruction type obtained is 18:HMC_FORWARD, which is consistent with the actual configured logical configuration.

Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
Basic									
AxisID	USINT	0	1	2	3	4	5	6	7
VarName	STRING	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
AxisType	EmAxisType	Unused	Unused	EtherCAT_...	Unused	Unused	Unused	Unused	Unused
Units	LREAL	1	1	1	1	1	1	1	1
Status									
MCmdType	EmCmdType	HMC_McIdle	HMC_McIdle	HMC_Move	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle
NCmdType	EmCmdType	HMC_McIdle	HMC_McIdle	HMC_Forward	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle	HMC_McIdle
AxisStatus	UDINT	0	0	0	0	0	0	0	0
AxisErrMask	UDINT	0	0	0	0	0	0	0	0
LimitState	USINT	0	0	0	0	0	0	0	0

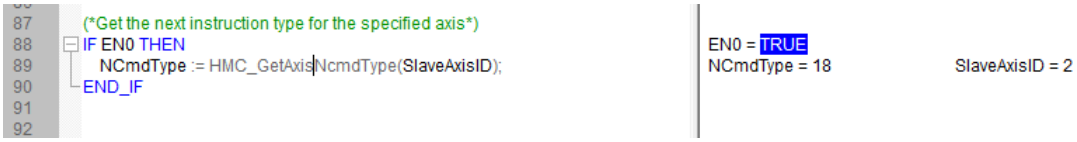
Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
CCmdType			UDINT	10	FALSE	Not Public	
NCmdType			UDINT	18	FALSE	Not Public	

LD program implementation:



ST program implementation:



5.8.22 HMC_GetAxisType (Getting Axis Parameter AxisType)

5.8.22.1 Function

It is used to get the axis parameter AxisType (axis type) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the axis type of the corresponding axis. If the set axis number exceeds the maximum value range (0~63), the output is 0.

The obtained axis status information is consistent with the axis parameter AxisType (axis type). For details, please refer to the description in the axis parameters chapter.

5.8.22.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisType	Output	DINT	Axis parameter AxisType (axis type) value

5.8.22.3 Instruction type

It is a non-axis motion cache instruction.

5.8.22.4 Example

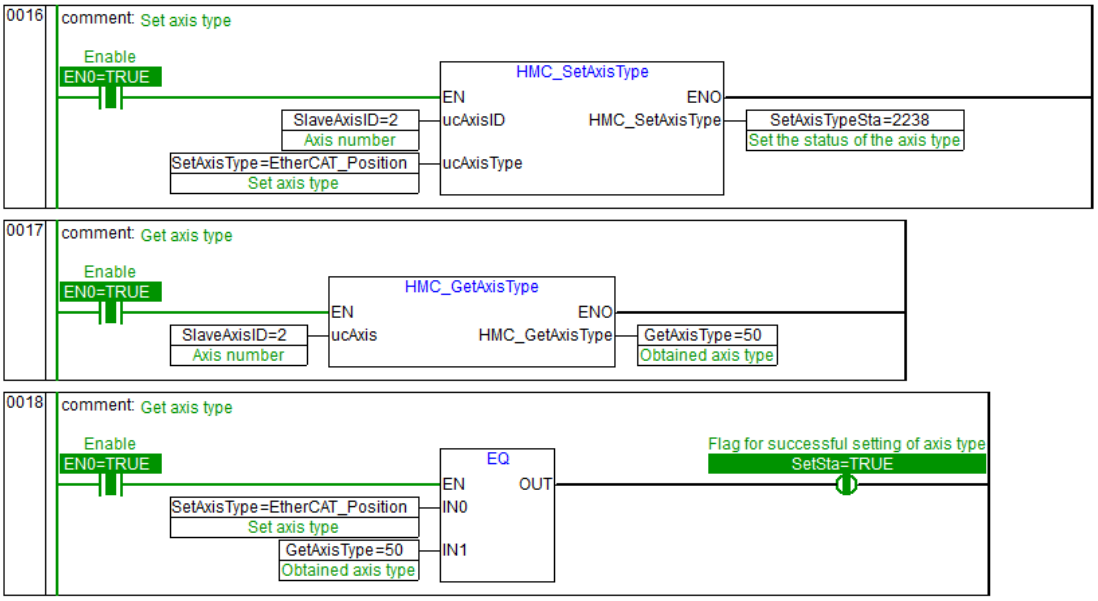
The following example uses the HMC_GetAxisType instruction to obtain the axis type with axis 2. When EN is enabled, this instruction starts the function of obtaining the axis type of axis 2 and returns the obtained axis type result to the AxisType variable. The next instruction type obtained after execution is 50:EtherCAT_Position, which is consistent with the actual configured logical configuration.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	

SetAxisType		Set axis type	EMAXISTYPE	EtherCAT_Position	FALSE	Not Public	
GetAxisType		Obtained axis type	DINT	50	FALSE	Not Public	
SetAxisTypeSta		Set the status of the axis type	UDINT	2238	FALSE	Not Public	
SetSta		Flag for successful setting of axis type	BOOL	TRUE	FALSE	Not Public	

LD program implementation:



ST program implementation:

5.8.23 HMC_GetAxisLimitState (Getting Axis Parameter LimitState)

5.8.23.1 Function

It is used to get the axis parameter LimitState (over-limit status) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the over-limit status of the corresponding axis. If the set axis number exceeds the maximum value range (0~63), the output is 0.

The obtained axis status information is consistent with the axis parameter LimitState (over-limit status). For details, please refer to the description in the axis parameters chapter.

5.8.23.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number

Parameter	Statement	Data Type	Description
HMC_GetAxisLimitState	Output	USINT	Axis parameter LimitState (over-limit status) value

5.8.23.3 Instruction type

It is a non-axis motion cache instruction.

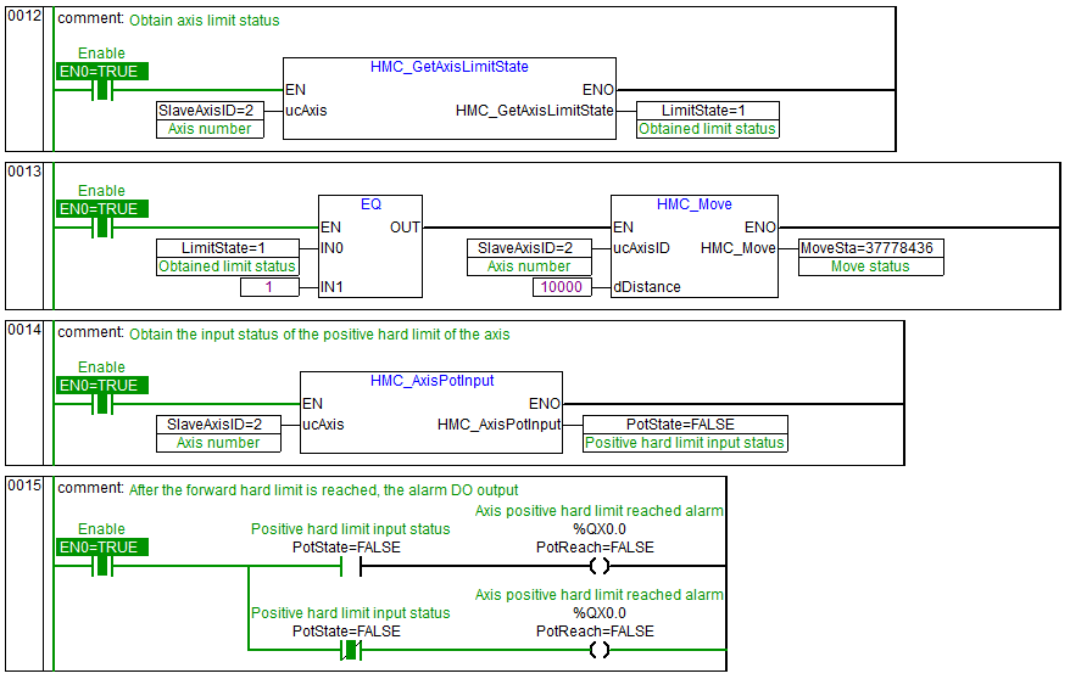
5.8.23.4 Example

The following example uses the HMC_GetAxisLimitState instruction to obtain the limit status of axis 2. Before obtaining it, you need to specify the forward hard limit channel. The example uses the HMC_HwLimitConfig instruction to set the axis forward hard limit channel number to 0. When the axis reaches the forward hard limit point through forward motion, the limit status variable LimitState obtained by the HMC_GetAxisLimitState instruction is 1. At this time, an alarm DO is output and the motor is reversed for 10000 pulses.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
PotState		Positive hard limit input status	BOOL	FALSE	FALSE	Not Public	
NotState		Negative hard limit input status	BOOL	FALSE	FALSE	Not Public	
PotReach	%QX0.0	Axis positive hard limit reached alarm	BOOL	FALSE	FALSE	Not Public	
NotReach	%QX0.1	Axis negative hard limit reached alarm	BOOL	FALSE	FALSE	Not Public	
LimitState		Obtained limit status	USINT	1	FALSE	Not Public	
MoveSta		Move status	UDINT	37778436	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
(*Set positive hard limit channel*)
IF EN0 THEN
    HMC_HwLimitConfig(SlaveAxisID, 0, 0);
END_IF


(*Obtain the input status of the positive hard limit of the axis, and after the status is true, output the alarr
IF EN0 THEN
    LimitState := HMC_GetAxisLimitState(SlaveAxisID); (*Obtain limit status*)
    IF LimitState=1 THEN
        MoveSta := HMC_Move(SlaveAxisID, -10000);
    END_IF

    PotState := HMC_AxisPotInput(SlaveAxisID);
    IF PotState THEN
        PotReach := TRUE;
    ELSE
        PotReach := FALSE;
    END_IF
END_IF
```

```
EN0 = TRUE
SlaveAxisID = 2

EN0 = TRUE
LimitState = 1                SlaveAxisID = 2
LimitState = 1
MoveSta = 37778436            SlaveAxisID = 2

PotState = FALSE              SlaveAxisID = 2
PotState = FALSE
PotReach = FALSE

PotReach = FALSE
```

5.8.24 HMC_GetAxisUnits (Getting Axis Parameter Units)

5.8.24.1 Function

It is used to get the axis parameter Units (unit conversion factor) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the unit conversion factor of the corresponding axis. If the set axis number exceeds the maximum value range (0~63), the output is 0.

The obtained axis status information is consistent with the axis parameter Units (unit conversion factor). For details, please refer to the description in the axis parameters chapter.

5.8.24.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisUnits	Output	LREAL	Axis parameter Units (unit conversion factor) value

5.8.24.3 Instruction type

It is a non-axis motion cache instruction.

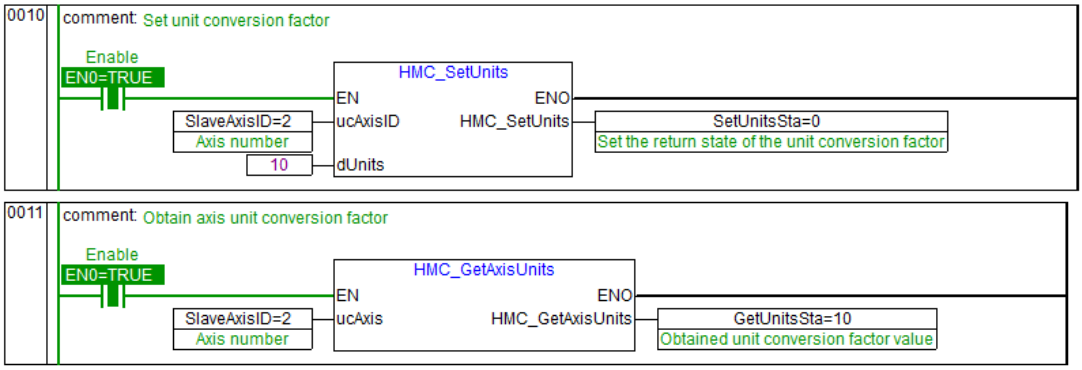
5.8.24.4 Example

The following example uses the HMC_GetAxisUnits instruction to obtain the unit conversion factor of axis 2. When EN is enabled, this instruction starts to obtain the user unit conversion factor that has been set by axis 2, and returns the obtained result to the SlaveUnits Variable. The conversion factor obtained is consistent with that written in the actual logical configuration.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
SetUnitsSta		Set the return state of the unit conversion factor	UDINT	0	FALSE	Not Public	
GetUnitsSta		Obtained unit conversion factor value	LREAL	10	FALSE	Not Public	

LD program implementation:



ST program implementation:

5.8.25 HMC_SetAxisSpeed (Setting Axis Parameter Speed)

5.8.25.1 Function

It is used to set the axis parameter Speed (target speed) of axis number. The input is the specified axis number and the target speed to be set. The axis number must be consistent with the actual configured axis number when setting. When the set target speed is negative, its absolute value will be assigned to the axis parameter Speed. If the setting is successful, 0 is returned. If the set axis number exceeds the maximum value range (0~63), 0xFFFFFFFF is returned.

For details, please refer to the description in the axis parameter Speed (target speed) chapter.

5.8.25.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dSpeed	Input	LREAL	Set target speed
HMC_SetAxisSpeed	Output	UDINT	0: Set successfully 0xFFFFFFFF: Failed

5.8.25.3 Instruction type

It is a non-axis motion cache instruction.

5.8.25.4 Example

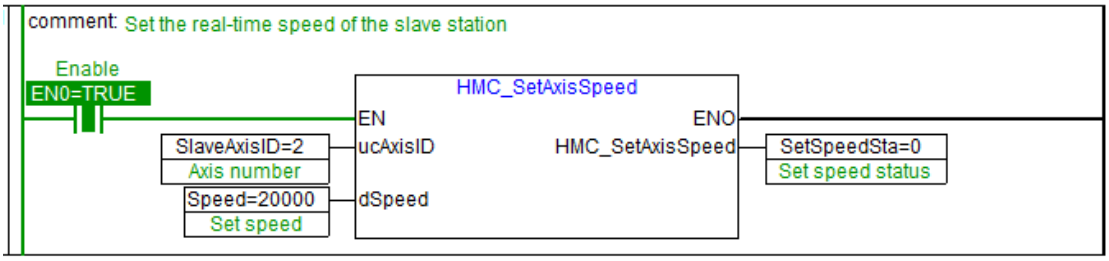
The following example uses the HMC_SetAxisSpeed instruction to set the target speed of axis 2. When EN is enabled, this instruction starts the function of setting the axis target speed. The set speed takes effect in real time. The actual speed of the servo motor is read and is consistent with the set value.

Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	0	0	3638165.0...	0	0	0	0	0
Remain	LREAL	0	0	1180	0	0	0	0	0
Velocity Profile									
Speed	LREAL	10000	2000	20000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	200000	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	200000	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	18000	0	0	0	0	0
VpSpeed	LREAL	0	0	20000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	5000000000	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN0		Enable	BOOL	TRUE	FALSE	Not Public	☐
Speed		Set speed	LREAL	20000	FALSE	Not Public	☐
SetSpeedSta		Set speed status	UDINT	0	FALSE	Not Public	☐

LD program implementation:



ST program implementation:

```
38  (*Set real-time speed*)
39  IF EN0 THEN
40      SetSpeedSta := HMC_SetAxisSpeed(SlaveAxisID, Speed);
41  END_IF
42
```

EN0 = TRUE
SetSpeedSta = 0 SlaveAxisID = 2 Speed = 20000

5.8.26 HMC_SetAxisAccel (Setting Axis Parameter Accel)

5.8.26.1 Function

It is used to set the axis parameter Accel (acceleration) of the specified axis number. The input is the specified

axis number and the acceleration to be set. The axis number must be consistent with the actual configured axis number when setting. When the set acceleration is negative, its absolute value will be assigned to the axis parameter Accel. If the setting is successful, 0 is returned. If the set axis number exceeds the maximum value range (0~63), 0xFFFFFFFF is returned.

For details, please refer to the description in the axis parameter Accel (acceleration) chapter.

5.8.26.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dAccel	Input	LREAL	Set acceleration
HMC_SetAxisAccel	Output	UDINT	0: Set successfully 0xFFFFFFFF: Failed

5.8.26.3 Instruction type

It is a non-axis motion cache instruction.

5.8.26.4 Example

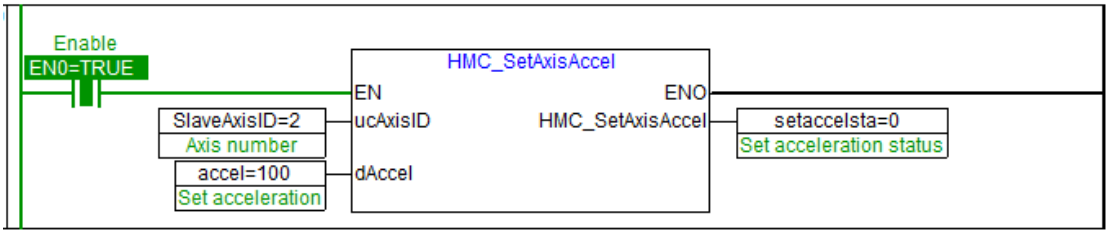
The following example uses the HMC_SetAxisAccel instruction to set the acceleration with axis 2. When EN is enabled, this instruction starts the function of setting the slave station acceleration. After the setting is successful, the monitored axis parameter value is consistent with the value in actual configuration logic settings.

Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	???	???	???	???	???	???	???	???
Remain	LREAL	???	???	???	???	???	???	???	???
Velocity Profile									
Speed	LREAL	10000	2000	3000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	2000	0	0	0	0	0
VpSpeed	LREAL	0	0	3000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	100	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN0		Enable	BOOL	TRUE	FALSE	Not Public	☐
accel		Set acceleration	LREAL	100	FALSE	Not Public	☐
setaccelsta		Set acceleration status	UDINT	0	FALSE	Not Public	☐

LD program implementation:



ST program implementation:

```
33  (*Set acceleration*)
34  IF EN0 THEN
35      setaccelsta := HMC_SetAxisAccel(SlaveAxisID, accel);
36  END_IF
```

EN0 = TRUE
setaccelsta = 0 SlaveAxisID = 2 accel = 100

5.8.27 HMC_SetAxisDecel (Setting Axis Parameter Decel)

5.8.27.1 Function

It is used to set the axis parameter Decel (deceleration) of the specified axis number. The input is the specified axis number and the deceleration to be set. The axis number must be consistent with the actual configured axis number when setting. When the set deceleration is negative, its absolute value will be assigned to the axis parameter Decel. If the setting is successful, 0 is returned. If the set axis number exceeds the maximum value range (0~63), 0xFFFFFFFF is returned.

For details, please refer to the description in the axis parameter Decel (deceleration) chapter.

5.8.27.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dDecel	Input	LREAL	Set deceleration
HMC_SetAxisDecel	Output	UDINT	0: Set successfully 0xFFFFFFFF: Failed

5.8.27.3 Instruction type

It is a non-axis motion cache instruction.

5.8.27.4 Example

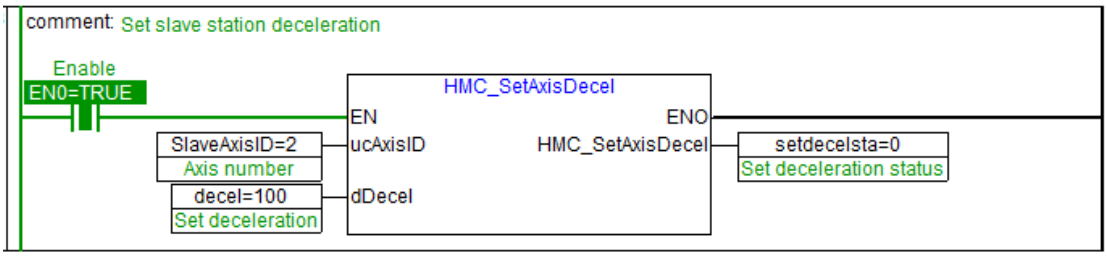
The following example uses the HMC_SetAxisDecel instruction to set the deceleration with axis 2. When EN is enabled, this instruction starts the function of setting the axis deceleration. After the setting is successful, the monitored axis parameter value is consistent with the value in actual configuration logic settings.

Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	???	???	???	???	???	???	???	???
Remain	LREAL	???	???	???	???	???	???	???	???
Velocity Profile									
Speed	LREAL	10000	2000	3000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	2000	0	0	0	0	0
VpSpeed	LREAL	0	0	3000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	100	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
decel		Set deceleration	LREAL	100	FALSE	Not Public	
setdecelsta		Set deceleration status	UDINT	0	FALSE	Not Public	

LD program implementation:



ST program implementation:

5.8.28 HMC_SetAxisJerk (Setting Axis Parameter Jerk)

5.8.28.1 Function

It is used to set the axis parameter Jerk (jerk) of the specified axis number. The input is the specified axis number and the jerk to be set. The axis number must be consistent with the actual configured axis number when setting. When the set jerk is negative, its absolute value will be assigned to the axis parameter Jerk. If the setting is successful, 0 is returned. If the set axis number exceeds the maximum value range (0~63), 0xFFFFFFFF is returned.

For details, please refer to the description in the axis parameter Jerk (Jerk) chapter.

5.8.28.2 Parameter

Parameter	Statement	Data Type	Storage Area	Description
ucAxisID	Input	USINT	M, G or constants	Axis number
dJerk	Input	LREAL	M, G or constants	Set jerk
HMC_SetAxisJerk	Output	UDINT	M or G	0: Set successfully 0xFFFFFFFF: Failed

5.8.28.3 Instruction type

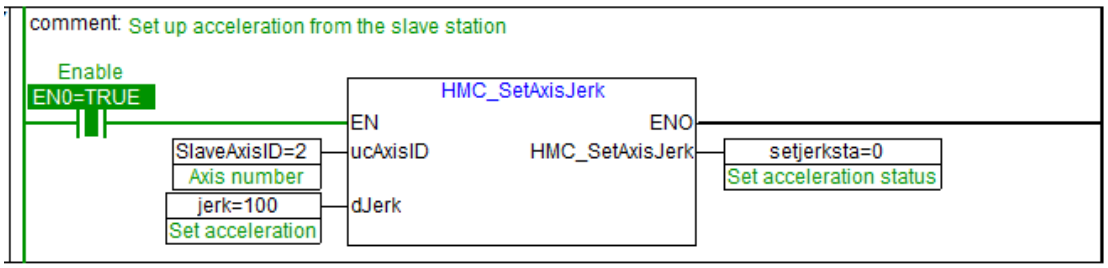
It is a non-axis motion cache instruction.

5.8.28.4 Example

The following example uses the HMC_SetAxisJerk instruction to set the jerk with axis 2. When EN is enabled, this instruction starts the function of setting the axis Jerk. After the setting is successful, the monitored axis parameter value is consistent with the value in actual configuration logic settings.

Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	???	???	???	???	???	???	???	???
Remain	LREAL	???	???	???	???	???	???	???	???
Velocity Profile									
Speed	LREAL	10000	2000	3000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	2000	0	0	0	0	0
VpSpeed	LREAL	0	0	3000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	100	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
jerk		Set acceleration	LREAL	100	FALSE	Not Public	
setjerksta		Set acceleration status	UDINT	0	FALSE	Not Public	




```
22  (*Set acceleration*)
23  IF EN0 THEN
24      setjerksta := HMC_SetAxisJerk(SlaveAxisID, jerk);
25  END_IF
26
```

EN0 = TRUE
setjerksta = 0 SlaveAxisID = 2 jerk = 100

5.8.29 HMC_SetAxisDAC (Setting Axis Parameter DAC)

5.8.29.1 Function

It is used to set the axis parameter DAC (speed mode open-loop output value) of the specified axis number. The input is the specified axis number and the speed mode open-loop output value to be set. The axis number must be consistent with the actual configured axis number when setting. If the setting is successful, 0 is returned. If the set axis number exceeds the maximum value range (0~63), 0xFFFFFFFF is returned.

For details, please refer to the description in the axis parameter DAC (speed mode open-loop output value) chapter.

5.8.29.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
iDAC	Input	DINT	Speed mode open-loop output value
HMC_SetAxisDAC	Output	UDINT	0: Set successfully 0xFFFFFFFF: Failed

5.8.29.3 Instruction type

It is a non-axis motion cache instruction.

5.8.29.4 Example

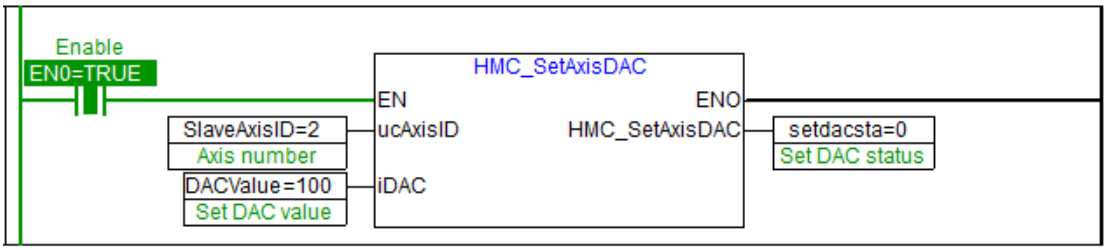
The following example uses the HMC_SetAxisDAC instruction to set the speed open-loop output value of axis 2. When EN is enabled, this instruction starts the speed open-loop output value function of axis. After the setting is successful, the monitored axis parameter value is consistent with the value in actual configuration logic settings.

Axis Parameters	Type	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
HoldIn	UDINT	65535	65535	65535	65535	65535	65535	65535	65535
Input									
KeNumerator	DINT	1	1	1	1	1	1	1	1
KeDenominator	DINT	1	1	1	1	1	1	1	1
EncoderValue	DINT	0	0	6051849	0	0	0	0	0
FeedBackSrcMode	USINT	0	0	0	0	0	0	0	0
FeedBackSrcValue	LREAL	0	0	0	0	0	0	0	0
FeedBackDataType	USINT	0	0	0	0	0	0	0	0
Output									
KsNumerator	DINT	1	1	1	1	1	1	1	1
KsDenominator	DINT	1	1	1	1	1	1	1	1
PulseNum	UDINT	0	0	3	0	0	0	0	0
ServoLoop	USINT	0	0	0	0	0	0	0	0
DAC	DINT	0	0	100	0	0	0	0	0
DACOut	DINT	0	0	0	0	0	0	0	0

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
DACValue		Set DAC value	DINT	100	FALSE	Not Public	
setdacsta		Set DAC status	UDINT	0	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
17  (*Set speed open-loop output value*)
18  IF EN0 THEN
19      setdacsta := HMC_SetAxisDAC(SlaveAxisID, DACValue);
20  END_IF
```

EN0 = TRUE
setdacsta = 0 SlaveAxisID = 2 DACValue = 100

5.8.30 HMC_SetAxisCreepSpeed (Setting Axis Parameter CreepSpeed)

5.8.30.1 Function

It is used to set the axis parameter CreepSpeed (origin regression creep speed) of the specified axis number. The input is the specified axis number and the origin regression creep speed to be set. The axis number must be consistent with the actual configured axis number when setting. When the origin regression creep speed is a negative value, its absolute value will be assigned to the axis parameter CreepSpeed. If the setting is successful, 0 is returned. If the set axis number exceeds the maximum value range (0~63), 0xFFFFFFFF is returned.

For details, please refer to the description in the axis parameter CreepSpeed (origin regression creep speed) chapter.

5.8.30.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
dCreepSpeed	Input	LREAL	Set origin regression creep speed value
HMC_SetAxisDAC	Output	UDINT	0: Set successfully 0xFFFFFFFF: Failed

5.8.30.3 Instruction type

It is a non-axis motion cache instruction.

5.8.30.4 Example

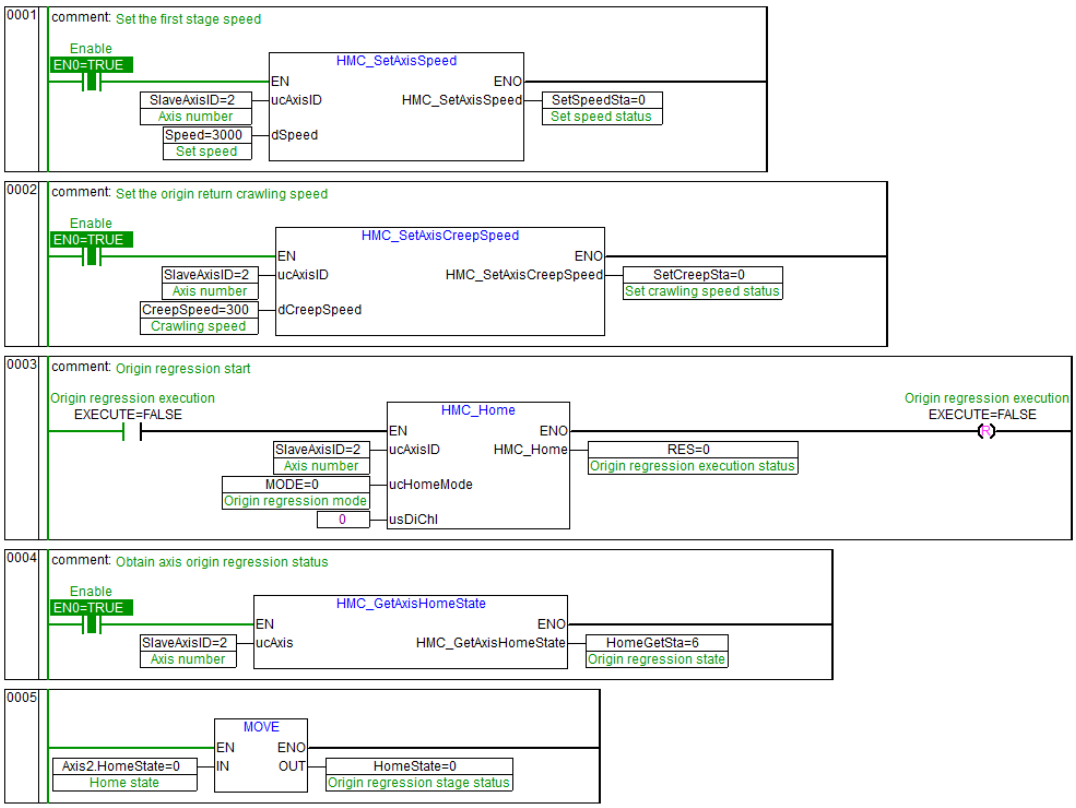
The following example performs the origin regression operation on the servo system with axis 2. The origin regression mode is set to 0, and the DI channel is set to channel 0. The HMC_SetAxisCreepSpeed instruction is used to set the creep speed in the third stage of origin regression. After the setting is completed, the HMC_Home instruction is called to start the origin regression. During the execution of origin regression, the HMC_GetAxisHomeState instruction is used to obtain the origin regression status.

Origin regression process in mode 17: After the origin regression is started, the axis moves forward at the speed set by Speed (stage 1), then decelerates and stops when encountering the rising edge of DI (stage 2), and then moves in the reverse direction at the creep speed set by CreepSpeed (stage 3) when encountering the falling edge of DI, then, with this point set as the origin position, starts to decelerate and stop, then returns to the origin position (stage 4), and finally completes the origin regression process (stage 6).

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
EXECUTE		Origin regression execution	BOOL	FALSE	FALSE	Not Public	
HomeState		Origin regression stage stat...	USINT	0	FALSE	Not Public	
MODE		Origin regression mode	BYTE	0	FALSE	Not Public	
RES		Origin regression execution ...	UDINT	0	FALSE	Not Public	
SetCreepSta		Set crawling speed status	UDINT	0	FALSE	Not Public	
CreepSpeed		Crawling speed	LREAL	300	FALSE	Not Public	
HomeGetSta		Origin regression state	USINT	6	FALSE	Not Public	
TargetSpeed		Target speed	LREAL	0	FALSE	Not Public	
Speed		Set speed	LREAL	3000	FALSE	Not Public	
SetSpeedSta		Set speed status	UDINT	0	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
(*Origin regression*)
IF EN0 THEN
    HMC_SetAxisSpeed(SlaveAxisID, Speed);
    HMC_SetAxisCreepSpeed(SlaveAxisID, CreepSpeed); (*Set crawling speed*)

    IF EXECUTE THEN
        HMC_Home(SlaveAxisID, MODE, 0); (*Origin regression execution*)
        EXECUTE := FALSE;
    END_IF

    HomeGetSta := HMC_GetAxisHomeState(SlaveAxisID); (*Obtain Origin Regression Status*)

END_IF
```

```
EN0 = TRUE
SlaveAxisID = 2          Speed = 3000
SlaveAxisID = 2          CreepSpeed = 300

EXECUTE = FALSE
SlaveAxisID = 2          MODE = 0
EXECUTE = FALSE

HomeGetSta = 0           SlaveAxisID = 2
```

5.8.31 HMC_GetAxisHomeState (Getting Axis Parameter HomeState)

5.8.31.1 Function

It is used to get the axis parameter HomeState (origin regression status) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the origin regression status of the corresponding axis. If the set axis number exceeds the maximum value range (0~63), the output is 0.

The obtained axis status information is consistent with the axis parameter HomeState (origin regression status). For details, please refer to the description in the axis parameters chapter.

5.8.31.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_GetAxisHomeState	Output	USINT	Origin regression status value, please refer to the axis parameter HomeState (origin regression status) for details.

5.8.31.3 Instruction type

It is a non-axis motion cache instruction.

5.8.31.4 Example

Refer to the example in HMC_SetAxisCreepSpeed (Setting axis parameter CreepSpeed).

5.8.32 HMC_GetAxisHard LimitCfg (Getting Axis Limit Configuration)

5.8.32.1 Function

It is used to get the axis hard limit configuration data of the specified aixs. The input is the axis number and the selection flags of the forward and reverse limits. The axis number must be consistent with the actual configured axis number when setting. The selection flags of the forward and reverse limits are to select whether you want to get the configuration information of the forward limit or the reverse limit. The output is the obtained hard limit configuration data, including the limit channel number and the limit logic value. 0 is returned if successful, and -1 is returned if failed.

5.8.32.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
ucSelect	Input	USINT	Set forward and reverse limit flags 0: Forward limit, 1: Reverse limit
pHard LimitChl	Output	POINTER TO UDINT	Limit channel number
pHard LimitLogic	Output	POINTER TO UDINT	Limit logic value 0: Normally open, 1: Normally closed
HMC_GetAxisHardLimitCfg	Output	USINT	0: Set successfully -1: Setting error

5.8.32.3 Instruction type

It is a non-axis motion cache instruction.

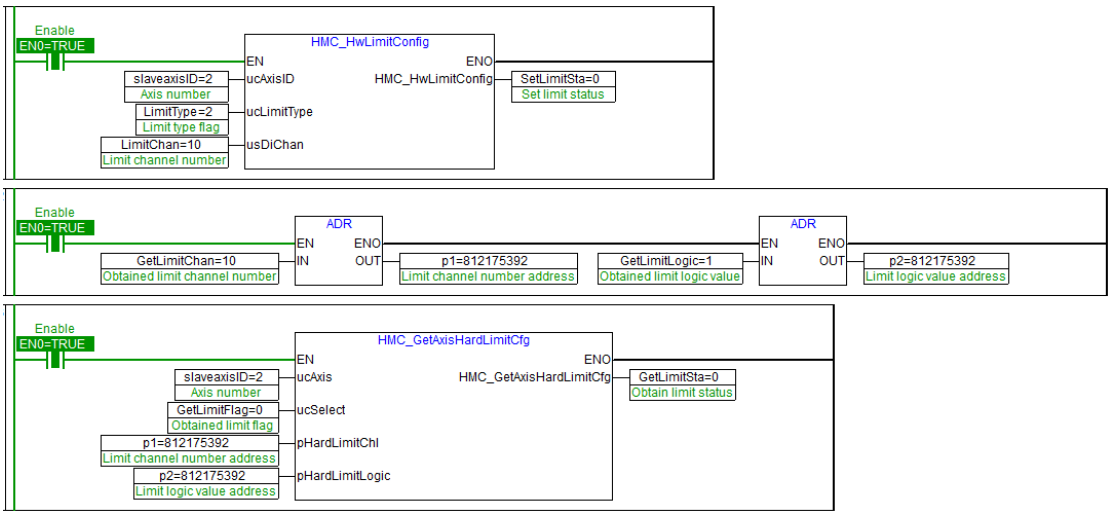
5.8.32.4 Example

The following example uses the HMC_GetAxisHard LimitCfg instruction to obtain the hard limit configuration data of axis 2. After EN0 is enabled, this instruction starts the function of obtaining the axis hard limit configuration data.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
slaveaxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN0		Enable	BOOL	TRUE	FALSE	Not Public	☐
LimitType		Limit type flag	USINT	2	FALSE	Not Public	☐
LimitChan		Limit channel number	UDINT	10	FALSE	Not Public	☐
SetLimitSta		Set limit status	UDINT	0	FALSE	Not Public	☐
GetLimitChan		Obtained limit channel number	UDINT	10	FALSE	Not Public	☐
GetLimitLogic		Obtained limit logic value	UDINT	1	FALSE	Not Public	☐
p1		Limit channel number address	POINTER TO UDINT	812175392	FALSE	Not Public	☐
p2		Limit logic value address	POINTER TO SINT	812175392	FALSE	Not Public	☐
GetLimitSta		Obtain limit status	UDINT	0	FALSE	Not Public	☐
GetLimitFlag		Obtained limit flag	USINT	0	FALSE	Not Public	☐

LD program implementation:



ST program implementation:

```
IF EN0 THEN

    SetLimitSta := HMC_HwLimitConfig(slaveaxisID, LimitType, LimitChan);  (*Configure hard limits*)
    GetLimitSta := HMC_GetAxisHardLimitCfg(slaveaxisID, GetLimitFlag, ADR(GetLimitChan), ADR(GetLimitLogic)); (*Obtain hard limit configuration information*)

END_IF
```

5.8.33 HMC_GetAxisVpSpeed (Getting Axis Parameter VpSpeed)

5.8.33.1 Function

It is used to get the axis parameter VpSpeed (solving speed) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the speed value in user units/second calculated in real time by the motion instruction of the corresponding axis. If the set axis number exceeds the maximum value range (0~63), the output is -1.

The obtained axis status information is consistent with the axis parameter VpSpeed (solving speed). For details, please refer to the description in the axis parameters chapter.

5.8.33.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisVpSpeed	Output	LREAL	Axis parameter VpSpeed (solving speed) value

5.8.33.3 Instruction type

It is a non-axis motion cache instruction.

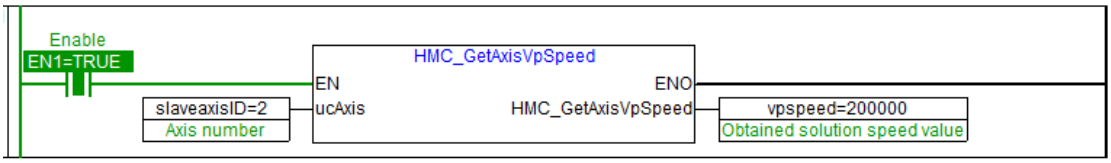
5.8.33.4 Example

The following example uses the HMC_GetAxisVpSpeed instruction to obtain the real-time target solving speed of aixs 2.

Variable type:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
SlaveAxisID		Axis number	USINT	2	FALSE	Not Public	
EN0		Enable	BOOL	TRUE	FALSE	Not Public	
DACValue		Set DAC value	DINT	100	FALSE	Not Public	
setdacsta		Set DAC status	UDINT	0	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
vpspeed := HMC_GetAxisVpSpeed(slaveaxisID);
```

vpspeed = 200000 slaveaxisID = 2

5.8.34 HMC_GetAxisMpSpeed (Getting Axis Parameter MpSpeed)

5.8.34.1 Function

It is used to get the axis parameter MpSpeed (feedback speed) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the real-time measured feedback speed of the corresponding axis in user units/second. If the set axis number exceeds the maximum value range (0~63), the output is -1.

The obtained axis status information is consistent with the axis parameter MpSpeed (feedback speed). For details, please refer to the description in the axis parameters chapter.

5.8.34.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisMpSpeed	Output	LREAL	Axis parameter MpSpeed (feedback speed) value

5.8.34.3 Instruction type

It is a non-axis motion cache instruction.

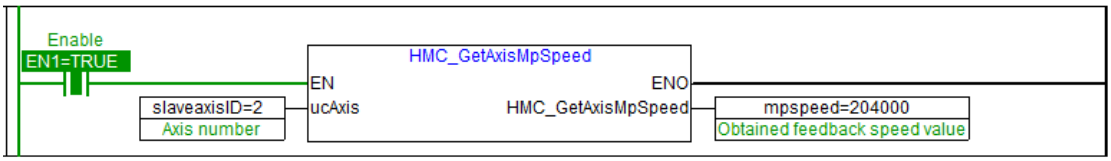
5.8.34.4 Example

The following example uses the HMC_ GetAxisMpSpeed instruction to obtain the real-time measured feedback speed of aixs 2.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
slaveaxisID		Axis number	USINT	2	FALSE	Not Public	
EN1		Enable	BOOL	TRUE	FALSE	Not Public	
mpspeed		Obtained feedback speed value	LREAL	204000	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
mpspeed := HMC_GetAxisMpSpeed(slaveaxisID);
```

mpspeed = 204000 slaveaxisID = 2

5.8.35 HMC_GetAxisDpos (Getting Axis Parameter Dpos)

5.8.35.1 Function

It is used to get the axis parameter Dpos (target position of this cycle) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the calculated target position of this cycle for the corresponding axis in user units. If the set axis number exceeds the maximum value range (0~63), the output is -1.

The obtained axis status information is consistent with the axis parameter Dpos (target position of this cycle). For details, please refer to the description in the axis parameters chapter.

5.8.35.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisDpos	Output	LREAL	Axis parameter Dpos (target position of this cycle) value

5.8.35.3 Instruction type

It is a non-axis motion cache instruction.

5.8.35.4 Example

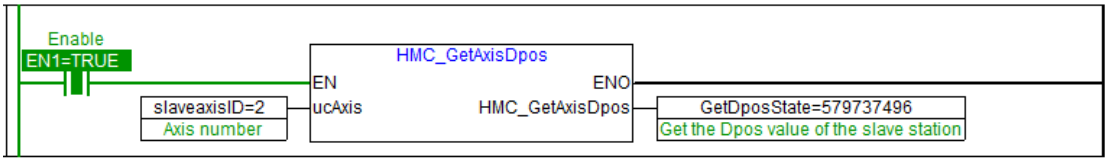
The following example uses the HMC_GetAxisDpos instruction to obtain the current cycle target position of

axis 2.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
slaveaxisID		Axis number	USINT	2	FALSE	Not Public	
EN1		Enable	BOOL	TRUE	FALSE	Not Public	
GetDposState		Get the Dpos value of the slave station	LREAL	579737496	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
14    GetDposState := HMC_GetAxisDpos(slaveaxisID);
```

GetDposState = 579737496 slaveaxisID = 2

5.8.36 HMC_GetAxisMpos (Getting Axis Parameter Mpos)

5.8.36.1 Function

It is used to get the axis parameter Mpos (feedback position) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the real-time feedback position of the corresponding axis in user units. If the set axis number exceeds the maximum value range (0~63), the output is -1.

The obtained axis status information is consistent with the axis parameter Mpos (feedback position). For details, please refer to the description in the axis parameters chapter.

5.8.36.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisMpos	Output	LREAL	Axis parameter Mpos (feedback position) value

5.8.36.3 Instruction type

It is a non-axis motion cache instruction.

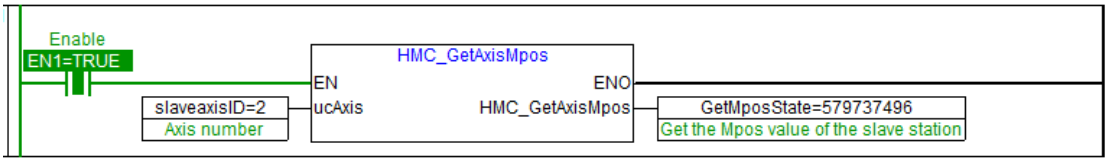
5.8.36.4 Example

The following example uses the HMC_GetAxisMpos instruction to obtain the feedback position of axis 2.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
slaveaxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN1		Enable	BOOL	TRUE	FALSE	Not Public	☐
GetMposState		Get the Mpos value of the slave station	LREAL	579737496	FALSE	Not Public	☐

LD program implementation:



ST program implementation:

```
15    GetMposState := HMC_GetAxisMpos(slaveaxisID);
```

GetMposState = 579737496 slaveaxisID = 2

5.8.37 HMC_GetAxisEnd Pos (Getting Axis Parameter End Pos)

5.8.37.1 Function

It is used to get the axis parameter End Pos (instruction target position) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the target position of the current motion instruction in user units. If the set axis number exceeds the maximum value range (0~63), the output is -1.

The obtained axis status information is consistent with the axis parameter End Pos (instruction target position). For details, please refer to the description in the axis parameters chapter.

5.8.37.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisEnd Pos	Output	LREAL	Axis parameter End Pos (instruction target position) value

5.8.37.3 Instruction type

It is a non-axis motion cache instruction.

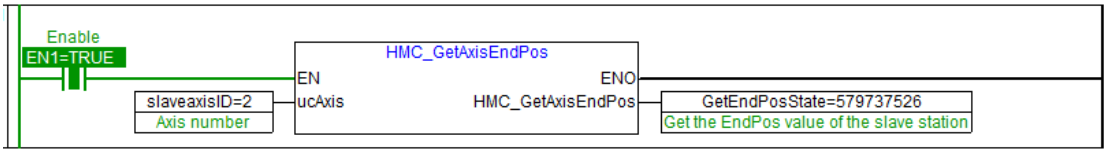
5.8.37.4 Example

The following example uses the HMC_GetAxisEndPos instruction to obtain the current motion instruction target position of axis 2.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
slaveaxisID		Axis number	USINT	2	FALSE	Not Public	☐
EN1		Enable	BOOL	TRUE	FALSE	Not Public	☐
GetEndPosState		Get the EndPos value of the slave station	LREAL	579737526	FALSE	Not Public	☐

LD program implementation:



ST program implementation:

5.8.38 HMC_GetAxisRemain (Getting Axis Parameter Remain)

5.8.38.1 Function

It is used to get the axis parameter Remain (remaining positions) of the specified axis number. The input is the specified axis number. The axis number must be consistent with the actual configured axis number when setting. The output is the remaining displacement of the corresponding axis to the instruction target position in user units. If the set axis number exceeds the maximum value range (0~63), the output is -1.

The obtained axis status information is consistent with the axis parameter Remain (remaining positions). For details, please refer to the description in the axis parameters chapter.

5.8.38.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	BYTE	Axis number
HMC_GetAxisRemain	Output	LREAL	Axis parameter Remain (remaining position) value

5.8.38.3 Instruction type

It is a non-axis motion cache instruction.

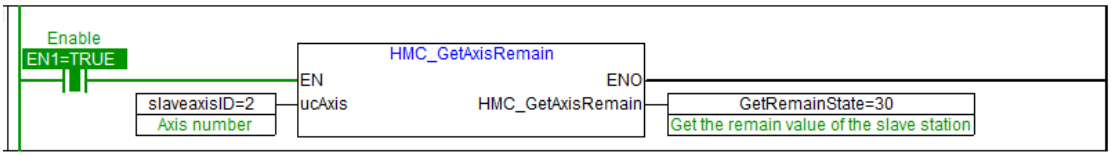
5.8.38.4 Examples

The following example uses the HMC_GetAxisRemain instruction to obtain the remaining positions of axis 2.

Variable definition:

Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
slaveaxisID		Axis number	USINT	2	FALSE	Not Public	
EN1		Enable	BOOL	TRUE	FALSE	Not Public	
GetRemainState		Get the remain value of the slave station	LREAL	30	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
17    GetRemainState := HMC_GetAxisRemain(slaveaxisID);
```

GetRemainState = 30 slaveaxisID = 2

5.9 Status Read Instruction

5.9.1 HMC__GetSysErr (Getting System Error)

5.9.1.1 Function

It is used to get the latest system error. If there is no system error, 0 is returned.

When the algorithm is running in real time, errors will be reported for detected anomalies. Error types include system errors and user errors.

System errors will not occur during normal operation unless the data used by the algorithm is tampered with by external program modules, leading to the introduction of some code branches that are impossible to enter, thus causing system errors.

System errors are a serious error type, so when a system error occurs, the watchdog will be automatically closed and the algorithm will stop solving (except for "timeout error"). After troubleshooting the cause of the error, you need to restart the controller to recover.

The "timeout error" in the system errors, with error ID 1025, is quite special. This error means that the time spent in a single operation of the algorithm module exceeds the current servo cycle. When this system error occurs, only an alarm is generated, with the watchdog not closed, and the algorithm still solves normally.

5.9.1.2 Parameter

Parameter	Statement	Data Type	Description
HMC__GetSysErr	Output	UDINT	0: No error Other values: A system error currently occurs

5.9.1.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.2 HMC_GetUserErrNum (Getting the Number of User Errors)

5.9.2.1 Function

Returns the number of user errors that have occurred.

When there have been no user errors in the history, the return value is 0. If a user error has occurred, currently only 1 is supported and the cumulative number of errors is not supported.

5.9.2.2 Parameter

Parameter	Statement	Data Type	Description
HMC_GetUserErrNum	Output	UDINT	The number of user errors that have occurred.

5.9.2.3 Instruction type

It is a non-axis motion cache instruction.

5.9.2.4 Precautions for Use

When no user error has occurred in the history, the return value is 0. If a user error has occurred, it only supports returning 1 currently, but does not support returning the cumulative number of errors.

5.9.3 HMC_GetUserErrID (Getting User Error Code)

5.9.3.1 Function

Return the user error code.

User errors are caused by illegal parameter settings, such as RepDist being set to less than 0 in loop mode, or the sampling function block setting the sampling channel number beyond the system's supported sampling channel range. Therefore, it can be seen that user errors are caused by human error in parameter settings.

When a user error occurs, the system will not take any other protective actions, such as stopping the calculation or disabling the servo enable.

At present, only the current user error code can be returned, and it is not supported to return error codes by index.

5.9.3.2 Parameter

Parameter	Statement	Data Type	Description
uiErrIndex	Input	UDINT	Error index value range: 0~ (HMC_GetUserErrNum()-1)
HMC_GetUserErrID	Output	UDINT	User error code

5.9.3.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.4 HMC_GetUserErrMes (Getting Additional Information about User Error Code)

5.9.4.1 Function

Return additional information about user error codes. By combining this additional information with other information about the firmware version, the function that generates the user error can be determined.

At present, only the current user's error code additional information can be returned, and it is not supported to return error code additional information by index at the moment.

5.9.4.2 Parameters

Parameter	Statement	Data Type	Description
uiErrIndex	Input	UDINT	Error index Value range: 0~ (HMC_GetUserErrNum()-1)
HMC_GetUserErrMes	Output	UDINT	User's error code additional information

5.9.4.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.5 HMC_GetAllAxisErr (Getting Overall Abnormal Status of All Axes)

5.9.5.1 Function

It is used to get the overall abnormal status of all axes. When the logical AND operation result of the axis status AxisStatus and its mask AxisErrMask of more than 1 axis among the 64 axes is not 0, it is considered that there is an axis abnormality. That is, when the following expression is not 0, all axes are considered to be abnormal overall, and the function returns - 1 (0xFFFFFFFF): ((Axis0.AxisStatus and Axis0.AxisErrMask) or (Axis1.AxisStatus and Axis1.AxisErrMask) or … … or (Axis63. AxisStatus and Axis63.AxisErrMask)).

5.9.5.2 Parameter

Parameter	Statement	Data Type	Description
HMC_GetAllAxisErr	Output	UDINT	0: All axes are normal Other values: Axes are abnormal

5.9.5.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.6 HMC_GetCmdErr (Getting Axis Instruction Error)

5.9.6.1 Function

According to the axis instruction ID, return the axis instruction error. It returns 0 when there is no error, returns 0xffffffff when the axis instruction ID does not exist, and returns the error type when there is an error in the instruction ID.

The axis instruction ID is the return value of this function when the motion control instruction is sent.

5.9.6.2 Parameter

Parameter	Statement	Data Type	Description
uiAxisCmdID	Input	UDINT	Axis instruction ID
HMC_GetCmdErr	Output	UDINT	0: No error Other values: The axis instruction ID is not in the axis cache or the instruction ID is invalid

5.9.6.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.7 HMC_GetCmdState (Getting Axis Instruction Status)

5.9.7.1 Function

It is used to return the axis instruction status according to the axis instruction ID. It includes the following statuses:

0: Waiting to run

1: Running

2: Completed or no instructions

3: Sending

Other values: Parameter error

5.9.7.2 Parameter

Parameter	Statement	Data Type	Description
uiAxisCmdID	Input	UDINT	Axis instruction ID

Parameter	Statement	Data Type	Description
HMC_GetCmdState	Output	USINT	0: Waiting to run: it has been sent to the motion cache but has not started execution 1: Running 2: Completed or no instructions 3: Sending: It is being sent to the motion cache Other values: Error code (0xFF means a user parameter error)

5.9.7.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.8 HMC_WaitCmd Finish (Waiting for Instruction to Complete)

5.9.8.1 Function

Wait until "A motion instruction has completed running" before returning. Before the execution of this instruction is completed, HMC_WaitCmd Finish will remain blocked and will not return.

5.9.8.2 Parameter

Parameter	Statement	Data Type	Description
uiAxisCmdID	Input	UDINT	Axis instruction ID
HMC_WaitCmd Finish	Output	USINT	0: The execution of the instruction is completed Other values: Error code (0xFF means that the axis instruction ID is invalid)

5.9.8.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.9 HMC_WaitCmd Loaded (Waiting Instruction Loading)

5.9.9.1 Function

Wait until "A motion instruction (uiAxisCmdID) is loaded", that is, the instruction just starts executing, before returning. Before the instruction is loaded, HMC_WaitCmd Loaded will remain blocked and will not return. Instruction loading means that the instruction is successfully sent to the axis motion cache.

5.9.9.2 Parameter

Parameter	Statement	Data Type	Description
uiAxisCmdID	Input	UDINT	Axis instruction ID
HMC_WaitCmdLoaded	Output	USINT	0: The execution of the instruction is completed Other values: Error code (0xFF means that the axis instruction ID is invalid)

5.9.9.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.10 HMC_WaitAxisIdle (Waiting for Axis to be Free)

5.9.10.1 Function

Wait until "The motion instruction queue of an axis (ucAxisID) is empty" before returning, otherwise it will remain blocked and will not return.

5.9.10.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_WaitAxisIdle	Output	USINT	0: The axis is idle Other values: Error code (0x17 means the axis number is out of range)

5.9.10.3 Instruction Type

It is a non-axis motion cache instruction.

5.9.11 HMC_GetCmd BufferState (Getting Axis Instruction Cache Status)

5.9.11.1 Function

Get the instruction cache status of an axis (ucAxis). It is used to determine whether the current axis cache is not full, and return the axis cache status. It is mainly intended to prevent the phenomenon of "the program is blocked and does not continue to execute".

5.9.11.2 Parameter

Parameter	Statement	Data Type	Description
ucAxis	Input	USINT	Axis number

Parameter	Statement	Data Type	Description
HMC_GetCmdBufferState	Output	UDINT	0: The instruction cache is not full and instructions can be sent normally 1: The instruction cache is full, and blocking will occur if instructions are sent 2: The instruction cache is locked (currently instructions are being sent into the cache) 0xFFFFFFFF: Function parameter error

5.9.11.3 Instruction Type

It is a non-axis motion cache instruction.

5.10 Fault Clearing Instructions

5.10.1 HMC_ClearAxisErr (Clearing Axis Error)

5.10.1.1 Function

It is used to clear the error of an axis and set AxisStatus=0 for an axis.

Before clearing an error, it is necessary to first resolve the error in order to successfully clear it, otherwise the error cannot be cleared.

5.10.1.2 Parameter

Parameter	Statement	Data Type	Description
ucAxisID	Input	USINT	Axis number
HMC_ClearAxisErr	Output	UDINT	0: Succeeded; Other values: Error code (0xFFFFFFFF means the axis number is out of range)

5.10.1.3 Instruction Type

It is a non-axis motion cache instruction.

5.10.2 HMC_ClearAllErr (Clearing All Axis Errors)

5.10.2.1 Function

It is used to clear the errors of all axes and set AxisStatus=0 for all axes.

Before clearing an error, it is necessary to first resolve the error in order to successfully clear it, otherwise the error cannot be cleared.

5.10.2.2 Parameter

Parameter	Statement	Data Type	Description
HMC_ClearAllErr	Output	UDINT	0: Succeeded

5.10.2.3 Instruction type

It is a non-axis motion cache instruction.

Beijing HollySys Intelligent Technologies Co., Ltd..

Address: Di Sheng Middle Road, No.2,

Economic-Technological Development Area, 100176, Beijing, China

Tel: +86 010-5898 1588

Consulting Hotline: 400-811-1999

Technology Request: +86 010-58981514

Fax: +86 010-5898 1558

Hollysys Group Website: https://www.hollysys.com

Factory Automation Business Website: https://fa.hollysys.com/