
HansRobot

Communication Protocol Interface

Shenzhen Han`s robor Co.,Ltd

1 Edit History:

Version NO.	Date	Modified Content
V2. 5. 1	2017-11-30	1.Match the 2.5.101.80 version DCS
V3. 5. 9	2019-02-21	1.Add interface

2 Purpose of this document is written:

- ✧ Provide reference for interface preparation and maintenance of R&D personnel
- ✧ Provide interface usage instructions for other users who use the Hansrobot protocol interface

Interface Preview

Interface	Description
Electrify	Power the robot
BlackOut	Robot blackout
StartMaster	Start Master
CloseMaster	Close Master
GrpPowerOn	Robot servo on
GrpPowerOff	Robot servo off
GrpStop	Stop robot
GrpReset	Reset, Used to clear errors
StartAssistiveMode	Opening zero force teaching mode
CloseAssistiveMode	Close the zero force teaching mode
SetSpeedUp	Speed up
SetSpeedDown	Speed down
SetKinematicCoordinate	Setting tool coordinates
SetOverride	Set speed ratio
SetOutIOState	Set output IO state
SetSerialDO	Set the output IO state of the common serial port
SetConfigIO	Set the output IO state of the functional serial port
SetSerialAnalogOutput	Set up serial port analog output
SetSerialAnalogMode	Set up serial port analog mode
SetTrackingSwitch	Set control tracking switch
SetUserCoordinate	Setting user coordinates
SetPayload	Setting load
SetMoveJointMotionLimits	Setting the speed limit of joint motion
SetMovePcsMotionLimits	Setting the speed limit of space motion
SetStopJointMotionLimits	Setting the speed limit for stop motion of joint

SetStopPcsMotionLimits	Setting the speed limit for stop motion of space
SetRunningMode	Switch debug mode
SetSimulation	Setting simulative mode
SetBaseMountingAngle	Setting up the installation Angle
SetStartTimer	Open the timer
SetCloseTimer	Off timer
SetToolCoordinateMotion	Set tool coordinate motion
SetConveyorScale	Set percentage of conveyor belts
ReadAcsActualPos	Get the actual position of the angle coordinate
ReadPcsActualPos	Get the actual position of the space coordinate
ReadActualPos	Get the actual position
ReadRobotPosInfo	Get the robot position information
ReadOverride	Get speed
ReadRobotState	Get the state of robot
ReadMoveState	Get the motion state of robot
ReadInIOState	Get the state of input IO
ReadOutIOState	Get the state of output IO
ReadAnalogInput	Get the analog output
ReadSerialDI	Get the digital input bit state specified by the common serial port
ReadConfigDI	Get the digital input bit state specified by the function serial port
ReadSerialDO	Get the digital output bit state specified by the common serial port
ReadConfigDO	Get the digital output bit state specified by the function serial port
ReadSerialAnalog	Get the analog amount of the serial port
ReadMachineOrigin	Get the mechanical origin

ReadConveyorValue	Get the value of the conveyor encoder
ReadMultiEndatPos	Get the multi circle absolute position of the current direction of the robot
ReadDCSStatus	Get the current status of DCS
ReadForceSensorData	Read force sensor data
SetForceSensorMove	Set the state of force sensor motion
ShortJogJ	Angular motion, Fixed distance
ShortJogL	Spatial motion, Fixed distance
LongJogJ	Angular motion, Unfixed distance
LongJogL	Spatial motion, Unfixed distance
MoveJ	Robot moves to the specified angular coordinate position
MoveL	Robot moves straight to the specified space coordinate position
MoveV	Velocity mode motion
MoveZ	zigzagging
MoveB	Immediately change the end point of the robot's current movement to the specified space coordinate position
MoveBJ	Immediately change the end point of the robot's current movement to the specified angle coordinate position
MoveBP	Move in a linear motion to the specified angular coordinate position
MoveC	Arc motion
MoveCL	Circular motion
MoveRelL	The robot moves a certain distance from the specified spatial coordinate direction

MoveRelJ	The robot moves a certain distance from the specified angular coordinate direction
MoveHoming	The robot returns to the origin
StartServo	Set the fixed update cycle and look-ahead time when starting the robot online control (servoJ or servoP)
PushServoJ	Online joint position command control, the joint position is sent with the fixed update time set by StartServo, and the robot will track the joint position command in real time.
PushServoP	The online terminal TCP position command control transmits the TCP position with the fixed update time set by the StartServo, and the robot reversely converts the target position of the target TCP position in real time.
StartPushBlending	Start PushBlending (continuous motion), initialize
PushBlendingL	PushMoveL point
PushBlendingC	PushMoveC point
EndPushBlending	End PushBlending, send the point to the controller and start moving
MoveBEX	Nonuniform motion
HoldScriptFunc	Pause running script
ContinusScriptFunc	Continue running the script
Running function	Running function
ShutDown	Remote shutdown

1	Introduction.....	1
1.1	Overview	1
1.2	Message Format	2
2	System Architecture	4
2.1	Connection Mode.....	4
2.1.1	As Client	4
2.1.2	As Server.....	4
2.1.3	Configuration File	5
3	Communication Protocol	6
3.1	Electrify.....	6
3.2	BlackOut	6
3.3	StartMaster	7
3.4	CloseMaster	7
3.5	GrpPowerOn	7
3.6	GrpPowerOff.....	8
3.7	GrpStop	8
3.8	GrpReset	9
3.9	StartAssistiveMode	9
3.10	CloseAssistiveMode	10
3.11	SetSpeedUp.....	11
3.12	SetSpeedDown	11
3.13	SetKinematicCoordinate	12
3.14	SetOverride	13
3.15	SetOutIOState	13
3.16	SetSerialDO	14
3.17	SetConfigDO.....	14
3.18	SetSerialAnalogOutput	15
3.19	SetSerialAnalogMode	16
3.20	SetTrackingSwitch	16
3.21	SetUserCoordinate	17
3.22	SetPayload.....	18
3.23	SetMoveJointMotionLimits	19
3.24	SetMovePcsMotionLimits	19
3.25	SetStopJointMotionLimits	20

3. 26	SetStopPcsMotionLimits	21
3. 27	SetRunningMode	22
3. 28	SetSimulation.....	23
3. 29	SetBaseMountingAngle	23
3. 30	SetStartTimer	24
3. 31	SetCloseTimer.....	24
3. 32	SetToolCoordinateMotion.....	24
3. 33	SetConveyorScale	25
3. 34	ReadAcsActualPos.....	26
3. 35	ReadPcsActualPos	27
3. 36	ReadActualPos	28
3. 37	ReadRobotPosInfo	29
3. 38	ReadOverride	30
3. 39	ReadRobotState.....	30
3. 40	ReadMoveState	31
3. 41	ReadInIOState.....	32
3. 42	ReadOutIOState	33
3. 43	ReadAnalogInput	34
3. 44	ReadSerialDI.....	34
3. 45	ReadConfiglDI.....	35
3. 46	ReadConfigDo	36
3. 47	ReadSerialDO	36
3. 48	ReadSerialAnalog	37
3. 49	ReadMachineOrigin	37
3. 50	ReadConveyorValue	38
3. 51	ReadMultiEndatPos	39
3. 52	ReadDCSStatus.....	39
3. 53	ReadForceSensorData.....	40
3. 54	SetForceSensorMove	41
3. 55	ShortJogJ.....	42
3. 56	ShortJogL.....	43
3. 57	LongJogJ	43
3. 58	LongJogL	44
3. 59	MoveJ.....	45

3. 60	MoveL.....	45
3. 61	MoveV	46
3. 62	MoveB.....	48
3. 63	MoveBJ	49
3. 64	MoveBP	50
3. 65	MoveC.....	51
3. 66	MoveCL	52
3. 67	MoveRelL	53
3. 68	MoveRelJ	54
3. 69	MoveHoming	55
3. 70	MoveZ.....	55
3. 71	MoveBEX	57
3. 72	StartServo.....	58
3. 73	PushServoj	58
3. 74	PushServoP	59
3. 75	StartPushBlending.....	60
3. 76	PushBlendingL.....	60
3. 77	PushBlendingC	61
3. 78	EndPushBlending.....	62
3. 79	HoldScriptFunc	62
3. 80	ContinusScriptFunc.....	63
3. 81	Running function	63
3. 82	ShutDown	63
4	ErrorCode.....	65
5	Terminology	69
6	Control robot motion process—Sample.....	70
6. 1	Flow chart	70
6. 2	Sample code	71

1 Introduction

1.1 Overview

HansRobot communication protocol interface mainly provides HansRobot communication protocol standard. In accordance with content of this agreement, the user sends the designated message to the robot control system through tcp/ip (Hereinafter referred to as the control system). The control system is processed separately according to different message contents, and the processing result is returned to the sender server.

Messages for movement types, the control system returns the result of the command immediately (Typically, the return is 0, indicating that the command is successful and the others are error codes). Then the movement message sender needs to send the "ReadMoveState" message constantly to get the movement Whether done or not. In general, the next movement can be performed only when the robot has completed the current work. Except for "MoveB/MoveBJ" instructions.

For other types of messages, the control system returns the request result. For specific results, please see the

communication protocol.

The communication system only processes one full message at a time. For example, When sending multiple messages at once, the control system processes only the first message that satisfies the protocol format, other discard. If the message sent is incomplete, the control system returns the invalid result of the command.

1. 2 Message Format

(1) Communication protocol

Message name, Param1,Param2,Param3.....Paramn,;

The message format, as shown above, is composed of a message name and parameters. A complete message ends in English comma and semicolon, where each field is separated by an English comma.

Message commands and message replies are all in the ASCII format(String form).

The specific message format is shown in reference 3.

The parameter that is not used is assigned 0.

(2) Return format

1) Successful return:

Message name,OK,Param1,Param2,Param3.....Paramn,;

The message format, as shown above, consists of a message name, a success flag OK and parameters. A complete message ends in English comma and semicolon , where each field is separated by an English comma.

2) Fail to return:

Message name, Fail, ErrorCode,;

The message format, as shown above, consists of a message name, a failure flag Fail and error message code. A complete message ends in English comma and semicolon, where each field is separated by an English comma.

The error code is in the form of 'int'.

The specific error code is shown in reference 4.

2 System Architecture

The protocol adopts the standard c/s architecture. Han's Motor provides the client and server to receive and process the robot control message(can be configured through a configuration file change), and the whole communication process is carried out in tcp/ip mode.

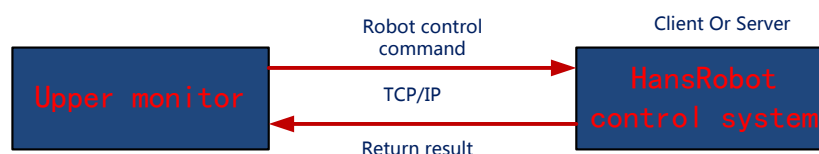


Fig 2-1 system architecture

2.1 Connection Mode

2.1.1 As Client

As a communication client, the HansRobot control system will monitor the connection server signals which is sent by the server of the upper monitor(Triggered in the form of IO). When the IO signal is received, the client will configure the IP address and port number according to the configuration file, and use the tcp/ip protocol to automatically connect the upper monitor server. When the server sends a connection request, if the client already connects to the server, the client closes the current connection first and then connects to the server.

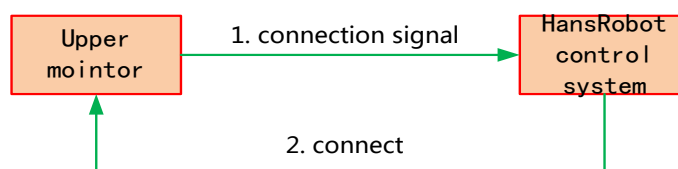


Fig 2-2 connection mode

2.1.2 As Server

As a communication server, the HansRobot control system will

monitor the designed port and waiting for the connection of the client of the upper monitor. In the above process, the port number of the server monitor can be set by configuration file.

2.1.3 Configuration File

By modifying the parameters through the communication protocol,it will not be saved to the configuration file,that is,the single cannon is valid. Restart the controller and restore the default parameters.In HansRobot,the settings of the instruction set will be saved to the configuration file.

3 Communication Protocol

3.1 Electrify

Function: Power the robot

Format: **Electrify,;**

Parameter quantity: None

Successful return:

"Electrify,OK,;"

Notes: successful completion of power up before returning, power up time is about 44s.

Fail to return:

" Electrify,Fail,ErrorCode,;", error types see the error code list

Example:

Electrify,;

3.2 BlackOut

Function: Robot blackout

Format: **BlackOut,;**

Parameter quantity: None

Successful return:

"BlackOut,OK,;"

Notes: successful power outage will only return, power failure time is 3s.

Fail to return:

" BlackOut,Fail,ErrorCode,;", error types see the error code list

Example:

BlackOut,;

3.3 StartMaster

Function: Start master station

Format: `StartMaster,;`

Parameter quantity: None

Successful return:

`"StartMaster,OK,;"`

Notes: the master station will not be returned until successfully started, startup master time is about 4s.

Fail to return:

`" StartMaster,Fail,ErrorCode,;"`, error types see the error code list

Example:

`StartMaster,;`

3.4 CloseMaster

Function: Close master station

Format: `CloseMaster,;`

Parameter quantity: None

Successful return:

`"CloseMaster,OK,;"`

Notes: the master station will not be returned until successfully closed, shut down the master station time is about 2s.

Fail to return:

`" CloseMaster,Fail,ErrorCode,;"`, error types see the error code list

Example:

`CloseMaster,;`

3.5 GrpPowerOn

Function: Robot servo on

Format: `GrpPowerOn,rbtID,;`

Parameter quantity: 1

Successful return:

”GrpPowerOn,OK,;”

Fail to return:

” GrpPowerOn,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:1

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

GrpPowerOn,0,;

3. 6 GrpPowerOff

Function: Robot servo off

Format: GrpPowerOff,rbtID,;

Parameter quantity:1

Successful return:

”GrpPowerOff,OK,;”

Fail to return:

” GrpPowerOff,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

GrpPowerOff,0,;

3. 7 GrpStop

Function: Stop robot

Format: GrpStop,rbtID,;

Parameter quantity: 1

Successful return:

”GrpStop,OK,;”

Fail to return:

” GrpStop,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

GrpStop,0,;

3. 8 GrpReset

Function: Reset, used to clear errors

Format: GrpReset,rbtID,;

Parameter quantity: 1

Successful return:

”GrpReset,OK,;”

Fail to return:

” GrpReset,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

GrpReset,0,;

3. 9StartAssistiveMode

Function:Opening zero force teaching mode。

Format: **StartAssistiveMode,rbtID,;**

Parameter quantity: 1

Successful return:

“StartAssistiveMode,OK,;”

Fail to return:

“ StartAssistiveMode,Fail,ErrorCode,;”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

StartAssistiveMode,0,;

3. 10 CloseAssistiveMode

Function: Close the zero force teaching mode

Format: **CloseAssistiveMode,rbtID,;**

Parameter quantity: 1

Successful return:

“CloseAssistiveMode,OK,;”

Fail to return:

“ CloseAssistiveMode,Fail,ErrorCode,;”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

CloseAssistiveMode,0,;

3.11 SetSpeedUp

function: Speed up

format: `SetSpeedUp,rbtID,nType,nSwitch,;`

parameter quantity: 3

successful return:

`"SetSpeedUp,OK,;"`

fail to return:

`"SetSpeedUp,Fail,ErrorCode,;"` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
nType	int	0>manual accelerate;1=auto accelerate
nSwitch	int	0=stop acceleration;1=sustained acceleration

3.12 SetSpeedDown

function: Speed down

format: `SetSpeedDown,rbtID,nType,nSwitch,;`

parameter quantity:3

successful return:

`"SetSpeedDown,OK,;"`

fail to return:

`"SetSpeedDown,Fail,ErrorCode,;"` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

nType	int	0>manual deceleration; 1=automatic deceleration
nSwitch	int	0=stop deceleration;1=sustained deceleration

3.13 SetKinematicCoordinate

Function: Setting tool coordinates

Format: **SetKinematicCoordinate,rbtID,X,Y,Z,A,B,C,;**

Parameter quantity:7

Successful return:

”SetKinematicCoordinate,OK,;”

Fail to return:

” SetKinematicCoordinate,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	X direction command position, unit mm
Y	double	Y direction command position, unit mm
Z	double	Z direction command position, unit mm
RX	double	RX direction command position, unit degree
RY	double	RY direction command position, unit degree
RZ	double	RZ direction command position, unit degree

Example:

SetKinematicCoordinate,0,0,0,100,0,0,0,;

3.14 SetOverride

function: Set speed ratio

format: **SetOverride,rbtID,override,;**

parameter quantity:2

successful return:

”SetOverride,OK,;”

fail to return:

” SetOverride,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
override	double	set speed ratio, range of 0.01~1 (1%~100%)

Example:

SetOverride,0,0.05,;

3.15 SetOutIOState

function: Set output IO state

format: **SetOutIOState,rbtID,ioIndex,ioState,;**

parameter quantity:3

successful return:

”SetOutIOState,OK,;,;”

fail to return:

” SetOutIOState,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
----------------	------	---------

rbtID	int	Robot index, count from zero
ioIndex	int	The output IO index to set, different robots can be use different index ranges
ioState	int	State required to set (0:low level;1:high level;)

Example:

SetOutIOState,0,1,1,;

3.16 SetSerialDO

function: Set the common serial port output IO state

format: **SetSerialDO,bit,state,;**

parameter quantity:2

successful return:

”SetSerialDO,OK,;”

fail to return:

” SetSerialDO,Fail,ErrorCode,;”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
bit	int	Serial port to set, count from zero
state	int	Set state: 0=low level;1=high level;

Example:

SetSerialDO,1,1,;

3.17 SetConfigDO

function: Set the function serial port output IO state

format: `SetConfigDO,bit,state,;`

parameter quantity:2

successful return:

`"SetConfigDO,OK,;"`

fail to return:

`"SetConfigDO,Fail,ErrorCode,;"`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
Bit	int	Serial port to set, count from zero
State	int	Set state: 0=low level;1=high level;

Example:

`SetConfigDO,1,1,;`

3.18 SetSerialAnalogOutput

function: Set up serial port analog output

format: `SetSerialAnalogOutput, nIndex, dValue , nPattern,;`

parameter quantity:3

successful return:

`"SetSerialAnalogOutput,OK,;"`

fail to return:

`"SetSerialAnalogOutput,Fail,ErrorCode,;"`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nIndex	int	Analog channel, count from zero
dValue	int	Analog value,voltage

		range:0~10V,current range:4~20mA
nPattern	int	Analog type,0=voltage,1=current

Example:

`SetSerialAnalogOutput,0,5,1,;`

3. 19 SetSerialAnalogMode

function: Set up serial port analog mode

format: `SetSerialAnalogMode, nIndex, nPattern,;`

parameter quantity:2

successful return:

`”SetSerialAnalogMode,OK,;”`

fail to return:

`”SetSerialAnalogMode,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nIndex	int	Analog channel, count from zero
nPattern	int	Analog type,0=voltage,1=current

Example:

`SetSerialAnalogMode,0 ,1,;`

3. 20 SetTrackingSwitch

Function: Set control tracking switch

Format: `SetTrackingSwitch,rbtID,TrackSwitch,;`

Parameter quantity:2

Successful return:

`”SetTrackingSwitch,OK,;”`

Fail to return:

`”SetTrackingSwitch,Fail,ErrorCode,;”` , error types see the error

code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
TrackSwitch	Int	Follwing switch:0=close;1=open

Example:

`SetTrackingSwitch,0,1,;`

3. 21 SetUserCoordinate

Function: Setting user coordinates

Format: `SetUserCoordinate,rbtID,X,Y,Z,A,B,C,;`

Parameter quantity:7

Successful return:

`SetUserCoordinate,OK,;`

Fail to return:

`SetUserCoordinate,Fail,ErrorCode,;` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	X direction command position, unit mm
Y	double	Y direction command position, unit mm
Z	double	Z direction command position, unit mm
RX	double	RX direction command position, unit degree

RX	double	RX direction command position, unit degree
RZ	double	RZ direction command position, unit degree

Example:

`SetUserCoordinate,0,0,0,0,180,0,180,;`

3.22 SetPayload

Function:Setting load

Format:

`SetPayload,nRbtID,mass,masscenterX,masscenterY,masscenterZ,;`

Parameter quantity:4

Successful return:

`“SetPayload,OK,;”`

Fail to return:

`“SetPayload,Fail,ErrorCode,;”`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nRbtID	int	Robot index, count from zero
mass	double	Load mass,unit kilogram
masscenterX	double	Load centroid x coordinates,unit millimeter
masscenterY	double	Load centroid y coordinates,unit millimeter
masscenterZ	double	Load centroid z coordinates,unit millimeter

Example:

`SetPayload,0,3,0,0,0,;`

3. 23 SetMoveJointMotionLimits

Function: Setting the speed limit of joint motion

Format:

SetMoveJointMotionLimits,nRbtID,Speed[6],Acc[6],Jerk[6],;

Parameter quantity:19

Successful return:

“SetMoveJointMotionLimits,OK,;”

Fail to return:

“SetMoveJointMotionLimits,Fail,ErrorCode,;” error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nRbtID	int	Robot index, count from zero
Speed[6]	double	Joint motion speed, unit:degree /s
Acc[6]	double	Joint motion acceleration, unit :degree /s ²
Jerk[6]	double	Joint motion jerk, unit: degree /s ³

3. 24 SetMovePcsMotionLimits

Function: Setting the speed limit of space motion

Format:

SetMovePcsMotionLimits,nRbtID,Linear_Speed,Linear_Acc,Linear_Jerk, Angualr_Speed, Angualr_Acc, Angualr_Jerk,;

Parameter quantity:7

Successful return:

“SetMovePcsMotionLimits,OK,;”

Fail to return:

“**SetMovePcsMotionLimits,Fail,ErrorCode,;**” error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nRbtID	int	Robot index, count from zero
Linear_Speed	double	Line motion speed, unit:degree /s
Linear_Acc	double	Line motion acceleration, unit :degree /s ²
Linear_Jerk	double	Line motion jerk, unit: degree /s ³
Angular_Speed	double	Angular motion speed, unit:degree /s
Angular_Acc	double	Angular motion acceleration, unit :degree /s ²
Angular_Jerk	double	Angular motion jerk, unit: degree /s ³

3. 25 SetStopJointMotionLimits

Function: Setting the speed limit for stop motion of joint

Format:

SetStopJointMotionLimits,nRbtID,Speed[6],Acc[6],Jerk[6],;

Parameter quantity:19

Successful return:

“**SetStopJointMotionLimits,OK,;**”

Fail to return:

“**SetStopJointMotionLimits,Fail,ErrorCode,;**” error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nRbtID	int	Robot index, count from zero
Speed[6]	double	Joint cessation speed, unit:degree /s
Acc[6]	double	Joint cessation acceleration, unit :degree /s ²
Jerk[6]	double	Joint cessation jerk, unit: degree /s ³

3. 26 SetStopPcsMotionLimits

Function: Setting the speed limit for stop motion of space

Format:

SetStopPcsMotionLimits,nRbtID,Linear_Speed,Linear_Acc,Linear_Jerk, Angualr_Speed, Angualr _Acc, Angualr _Jerk,;

Parameter quantity:7

Successful return:

“SetStopPcsMotionLimits,OK,;”

Fail to return:

“SetStopPcsMotionLimits,Fail,ErrorCode,;” error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nRbtID	int	Robot index, count from zero
Linear_Speed	double	Line cessation speed, unit:degree /s
Linear_Acc	double	Line cessation acceleration, unit :degree /s ²

Linear_Jerk	double	Line cessation jerk, unit: degree /s ³
Angular_Speed	double	Angular cessation speed, unit:degree /s
Angular_Acc	double	Angular cessation acceleration, unit :degree /s ²
Angular_Jerk	double	Angular cessation jerk, unit: degree /s ³

3.27 SetRunningMode

Function: Switch debug mode

Format: `SetRunningMode,nType,;`

Parameter quantity:1

Successful return:

`“SetRunningMode,OK,;”`

Fail to return:

`”SetRunningMode,Fail,ErrorCode,;”` , error types see the error code

list

Parameter detail:

Parameter name	type	Meaning
nType	int	0=normal mode; 1=debug mode

Example:

`SetRunningMode,1,;`

3. 28 SetSimulation

Function:Setting simulative mode

Format:**SetSimulation,nSimulation,;**

Parameter quantity:1

Successful return:

“SetSimulation,OK,;”

Fail to return:

“ SetSimulation,Fail,ErrorCode,;”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nSimulation	int	0=close simulative mode 1=open simulative mode

Example:

SetSimulation,1,;

3. 29 SetBaseMountingAngle

Function:Setting up the installation Angle

Format:**SetBaseMountingAngle,nRbtID,Rotation,Tilt,;**

Parameter quantity:3

Successful return:

“SetBaseMountionAngle,OK,;”

Fail to return:

“ SetBaseMountionAngle,Fail,ErrorCode,;”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nRbtID	int	Robot index, count from zero
Rotation	double	Rotation,unit degree

Tilt	double	Tilt,unit degree
------	--------	------------------

Example:

`SetBaseMountingAngle,0,90,90,;`

3. 30 SetStartTimer

Function:Open the timer

Format:`SetStartTimer,;`

Parameter quantity:0

Successful return:

`“SetStartTimer,OK,;”`

Fail to return:

`“ SetStartTimer,Fail,ErrorCode,;”`, error types see the error code list

Example:

`SetStartTimer,;`

3. 31 SetCloseTimer

Function:Off timer

Format:`SetCloseTimer,;`

Parameter quantity:0

Successful return:

`“SetCloseTimer,OK,;”`

Fail to return:

`“ SetCloseTimer,Fail,ErrorCode,;”` error types see the error code list

Example:

`SetCloseTimer,;`

3. 32 SetToolCoordinateMotion

Function: Set tool coordinate motion

Format: **SetToolCoordinateMotion,rbtID,Switch,;**

Parameter quantity:2

Successful return:

SetToolCoordinateMotion,OK,;

Fail to return:

SetToolCoordinateMotion,Fail,ErrorCode,; , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
Switch	int	Switch: 0=close; 1=open;

Example:

SetToolCoordinateMotion,0,1,;

3.33 SetConveyorScale

Function: Set percentage of conveyor belts

Format: **SetConveyorScale,rbtID,dCount,nDirection,;**

Parameter quantity: 3

Successful return:

SetConveyorScale,OK,;

Fail to return:

SetConveyorScale,Fail,ErrorCode,; , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
scale	double	conveyor belt scale (range of 0~1)

dCount	double	count per meter,greater than 0
nDirection	int	Following direction: 0=negative direction of Y direction 1=positive direction of Y direction

Example:

`SetConveyorScale,0,38956,1,;`

3.34 ReadAcsActualPos

Function: Get the actual position of the angle coordinate

Format: `ReadAcsActualPos,rbtID,;`

Parameter quantity: 1

Successful return:

`” ReadAcsActualPos,OK,J1,J2,J3,J4,J5,J6,;”`

Fail to return:

`” ReadAcsActualPos,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
J1	double	The position of J1 direction, unit: degree
J2	double	The position of J2 direction, unit: degree
J3	double	The position of J3 direction, unit: degree
J4	double	The position of J4 direction, unit: degree
J5	double	The position of J5 direction, unit: degree

J6	double	The position of J6 direction, unit: degree
----	--------	---

Example:

`ReadAcsActualPos,0,;`

3.35 ReadPcsActualPos

Function: Get the actual position of the space coordinate

Format: `ReadPcsActualPos,rbtID,;`

Parameter quantity:1

Successful return:

`” ReadPcsActualPos,OK,X,Y,Z,A,B,C,;”`

Fail to return:

`” ReadPcsActualPos,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	The position of X direction, unit: mm
Y	double	The position of Y direction, unit: mm
Z	double	The position of Z direction, unit: mm
RX	double	The position of RX direction, unit: mm
RY	double	The position of RY direction, unit: mm
RZ	double	The position of RZ direction, unit: mm

Example:

`ReadPcsActualPos,0,;`

3.36 ReadActualPos

Function: Get the actual position

Format: `ReadActualPos,rbtID,;`

Parameter quantity:1

Successful return:

`” ReadActualPos,OK,X,Y,Z,A,B,C,;”`

Fail to return:

`” ReadActualPos,Fail,ErrorCode,;”` , error types see the error code

list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
J1	double	The position of J1 direction, unit: degree
J2	double	The position of J2 direction, unit: degree
J3	double	The position of J3 direction, unit: degree
J4	double	The position of J4 direction, unit: degree
J5	double	The position of J5 direction, unit: degree
J6	double	The position of J6 direction, unit: degree
X	double	The position of X direction, unit: mm
Y	double	The position of Y direction, unit: mm

Z	double	The position of Z direction, unit: mm
RX	double	The position of RX direction, unit: mm
RY	double	The position of RY direction, unit: mm
RZ	double	The position of RZ direction, unit: mm

Example:

`ReadActualPos,0,;`

3. 37 ReadRobotPosInfo

Function: Get the robot position information

Format: `ReadRobotPosInfo,nRbtID,;`

Parameter quantity: 1

Successful return:

`”ReadRobotPosInfo,OK,AcsActualPos[6],PcsActualPos1[6],ActualCurr[6],AcsCmdPos[6],PcsCmdPos[6],CmdCurr[6],;”`

Fail to return:

`” ReadPcsActualPos,Fail,ErrorCode,;”`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
AcsActualPos[6]	double	Actual angle position,unit: degree
PcsActualPos[6]	double	Actual space position,unit: millimetre/degree
ActualCurr[6]	double	Actual current,unit: ampere

AcsCmdPos[6]	double	Command angle position,unit: degree
PcsCmdPos[6]	double	Command space position,unit: millimetre/degree
CmdCurr[6]	double	Command current,unit: ampere

Example:

`ReadRobotPosInfo,0,;`

3.38 ReadOverride

Function: Get speed

Format: `ReadOverride,rbtID,;`

Parameter quantity:1

Successful return:

`"ReadOverride,OK,;"`

Fail to return:

`"ReadOverride,Fail,ErrorCode,;"` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	Int	Robot index, count from zero

3.39 ReadRobotState

Function: Get the state of the robot

Format: `ReadRobotState,rbtID,;`

Parameter quantity:1

Successful return:

`"ReadRobotState,OK,movingState,powerState,errorState,errorCode,errDirectionID,BrakingState,reserval1State,reserval2State"`

,”

Fail to return:

” **ReadRobotState,Fail,ErrorCode,;**”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
movingState	int	0=no movement 1=in motion
powerState	int	0=de enable state 1=enabling state
errorState	int	0=Error free 1=Report errors
errorCode	int	Robot error code
errDirectionID	Int	The wrong direction ID of the manipulator
BrakingState	int	0=no brake operation 1=brake work
Reserval1State	int	Spare
Reserval2State	int	Spare

Example:

ReadRobotState,0,;

3. 40 ReadMoveState

Function: Get the motion state of the robot

Format: **ReadMoveState,rbtID,;**

Parameter quantity:1

Successful return:

” **ReadMoveState,OK,MoveState,;**”

Fail to return:

” **ReadMoveState,Fail,ErrorCode,;**” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
MoveState	int	Current state of motion of robot: 0=motion completion; 1009=in motion; 1013=waiting for execution; 1025=Error reporting state;

Example:

ReadMoveState,0,;

3. 41 ReadInIOState

Function: Get the state of the input IO

Format: **ReadInIOState,rbtID,ioIndex,;**

Parameter quantity:2

Successful return:

” **ReadInIOState,OK,state,;**”

Fail to return:

” **ReadInIOState,Fail,ErrorCode,;**” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	Int	Robot index, count from zero

ioIndex	int	The IO index to read
State	int	The state of input IO: 0=low level; 1=high level;

Example:

`ReadInIOState,0,1,;`

3.42 ReadOutIOState

Function: Get the state of the output IO

Format: `ReadOutIOState,rbtID,ioIndex,;`

Parameter quantity:2

Successful return:

`” ReadOutIOState,OK,state,;”`

Fail to return:

`” ReadOutIOState,Fail,ErrorCode,;”` , error types see the error code

list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
ioIndex	int	The IO index to read, count from one
state	int	The state of input IO: 0=low level; 1=high level;

Example:

`ReadOutIOState,0,1,;`

3. 43 ReadAnalogInput

Function: Get the terminal analog output

Format: `ReadAnalogInput,rbtID,nAnalogIndex,;`

Parameter quantity: 2

Successful return:

`”ReadAnalogInput,OK,nVal,;”`

Fail to return:

`”ReadAnalogInput,Fail,ErrorCode,;”`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
nIndex	Int	Analog index
nVal	Int	Analog input voltage

Example:

`ReadAnalogInput,0,1,;`

3. 44 ReadSerialDI

Function: Get the digital input bit state specified by the common serial port

Format: `ReadSerialDI,bit,;`

Parameter quantity: 1

Successful return:

`”ReadSerialDI,OK,state,;”`

Fail to return:

`”ReadSerialDI,Fail,ErrorCode,;”`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
bit	Int	Input bits needed to obtain
State	Int	The state of being returned, 0=low level;1=high level;

Example:

`ReadSerialDI,1,;`

3.45 ReadConfigDI

Function: Get the digital input bit state specified by the function serial port

Format: `ReadConfigDI,bit,;`

Parameter quantity:1

Successful return:

`” ReadCongfigDI,OK,state,;”`

Fail to return:

`” ReadCongfigDI,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
bit	Int	Input bits needed to obtain
State	Int	The state of being returned, 0=low level;1=high level;

Example:

`ReadConfigDI,1,;`

3. 46 ReadConfigDO

Function: Get the digital output bit state specified by the function serial port

Format: `ReadConfigDO,bit,;`

Parameter quantity: 1

Successful return:

`”ReadConfigDO,OK,state,;”`

Fail to return:

`”ReadConfigDO,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
Bit	Int	Output bits needed to obtain
State	Int	The state of being returned, 0=low level;1=high level;

Example:

`ReadConfigDO,1,;`

3. 47 ReadSerialDO

Function: Get the output bit state specified by the serial port

Format: `ReadSerialDO,bit,;`

Parameter quantity: 1

Successful return:

`”ReadSerialDO,OK,state,;”`

Fail to return:

`”ReadSerialDO,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
Bit	Int	Output bits needed to obtain
State	Int	The state of being returned, 0=low level;1=high level;

Example:

`ReadSerialDO,1,;`

3. 48 ReadSerialAnalog

Function: Get the analog amount of the serial port

Format: `ReadSerialAnalog,nIndex,;`

Parameter quantity: 1

Successful return:

`”ReadSerialAnalog,OK,nVal,;”`

Fail to return:

`” ReadSerialAnalog,Fail,ErrorCode,;”`, error types see the error code

list

Parameter detail:

Parameter name	type	Meaning
nIndex	Int	Analog bits that need to be obtained
nVal	Int	The amount of simulation returned

Example:

`ReadSerialAnalog,1,;`

3. 49 ReadMachineOrigin

Function: Get the multi circle absolute position of the current

direction of the robot (encoder value)

format: `ReadMultiEndatPos,rbtID,;`

Parameter quantity:1

Successful return:

`” ReadMultiEndatPos,OK,Pos[6],;”`

Fail to return:

`” ReadMultiEndatPos,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
Pos[6]	int	The encoder value of direction 1~6

Example:

`ReadMultiEndatPos,0,;`

3. 50 ReadConveyorValue

Function: Get the value of the conveyor encoder

Format: `ReadConveyorValue,rbtID,;`

Parameter quantity:1

Successful return:

`” ReadConveyorValue,OK,conveyorVal,;”`

Fail to return:

`” ReadConveyorValue,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	Int	Robot index, count from zero
conveyorVal	double	The value of the conveyor

		encoder
--	--	---------

Example:

`ReadConveyorValue,0,;`

3. 51 ReadMultiEndatPos

Function: Get the multi circle absolute position of the current direction of the robot (encoder value)

format: `ReadMultiEndatPos,rbtID,;`

Parameter quantity:1

Successful return:

`” ReadMultiEndatPos,OK,Pos[6],;”`

Fail to return:

`” ReadMultiEndatPos,Fail,ErrorCode,;”` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
Pos[6]	int	The encoder value of direction 1~6

Example:

`ReadMultiEndatPos,0,;`

3. 52 ReadDCSStatus

Function: Get DCS status

format: `ReadDCSStatus,;`

Parameter quantity:0

Successful return:

`” ReadDCSStatus,OK,nCurrentStatu,;”`

Fail to return:

” **ReadDCSStatus,Fail,ErrorCode;**”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
nCurrentStatu	int	Current status of DCS: 0=uninitialized 1=Initialization 2=The system board is not connected 3=The system board is not powered. 4=Rtos is not started 5=The controller is not started 6=Config 7=Emergency stop 8=Safe light curtain 9=Error status 10=normal 11=Run the script 12=The primary station is not started 13=zero force teaching state 14=The master station is starting up

Example:

ReadMultiEndatPos,0;

3. 53 ReadForceSensorData

Function: Read force sensor data

Format: **ReadForceSensorData,;**

Parameter quantity:0

Successful return:

” **ReadForceSensorData,OK,Fx,Fy,Fz,Mx, My, Mz,;**”

Fail to return:

” **ReadForceSensorData,Fail,ErrorCode,;**” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
Fx	double	X direction force value,unit: N
Fy	double	Y direction force value,unit: N
Fz	double	Z direction force value,unit: N
Mx	double	X direction torque matrix, unit: Nm
My	double	Y direction torque matrix, unit: Nm
Mz	double	Z direction torque matrix, unit: Nm

Example:

ReadForceSensorData,;

3. 54 SetForceSensorMove

Function: Set the state of force sensor motion

Format: **SetForceSensorMove,rbtID,moveState,;**

Parameter quantity:1

Successful return:

” **SetForceSensorMove,OK ,;**”

Fail to return:

” **SetForceSensorMove,Fail,ErrorCode,;**” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
moveState	int	0: Open 1: Close

Example:

SetForceSensorMove,0,1,;

3.55 ShortJogJ

Function: Angular motion, fixed distance exercise(the movement distance is 2 degrees)

Format: **ShortJogJ,rbtID,DirectionID,Derection,;**

Parameter quantity:3

Successful return:

”**ShortJogJ,OK,;**”

Fail to return:

” **ShortJogJ,Fail,ErrorCode,;**” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
DirectionID	int	Point moving axle ID(0-5) (0:J1,1:J2,2:J3,3:J4,4:J5,5:J6)
Derection	int	Movement direction: 0=negative;1=positive;

Example:

ShortJogJ,0,3,1,;

3. 56 ShortJogL

Function: Angular, unfixed distance movement

Format: LongJogJ,rbtID,DirectionID,Derection,;

Parameter quantity:3

Notes: When the order is issued, the other stop command must be issued to stop the movement

Successful return:

”LongJogJ,OK,;”

Fail to return:

” LongJogJ,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
DirectionID	int	Point moving axle ID(0-5) (0:J1,1:J2,2:J3,3:J4,4:J5,5:J6)
Derection	int	Movement direction: 0=negative;1=positive;

Example:

LongJogJ,0,3,1,;

3. 57 LongJogJ

Function: Angular, unfixed distance movement

Format: LongJogJ,rbtID,DirectionID,Derection,;

Parameter quantity:3

Notes: When the order is issued, the other stop command must be issued to stop the movement

Successful return:

”LongJogJ,OK,;”

Fail to return:

” LongJogJ,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
DirectionID	int	Point moving axle ID(0-5) (0:J1,1:J2,2:J3,3:J4,4:J5,5:J6)
Derection	int	Movement direction: 0=negative;1=positive;

Example:

LongJogJ,0,3,1,;

3. 58 LongJogL

Function: Spatial motion, unfixed distance motion

Format: LongJogL,rbtID,DirectionID,Derection,;

Parameter quantity:3

Notes: When the order is issued, the other stop command must be issued to stop the movement

Successful return:

”LongJogL,OK,;”

Fail to return:

” LongJogL,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
DirectionID	int	Point moving axle ID(0-5) (0:X,1:Y,2:Z,3:A,4:B,5:C)
Derection	int	Movement direction: 0=negative;1=positive;

Example:

LongJogL,0,3,1,;

3. 59 MoveJ

Function: Robot moves to the specified angular coordinate position

Format: **MoveJ,rbtID,J1,J2,J3,J4,J5,J6,;**

Parameter quantity: 7

Successful return:

”MoveJ,OK,;”

Fail to return:

” MoveJ,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
J1	double	Direction 1 command position, unit degree
J2	double	Direction 2 command position, unit degree
J3	double	Direction 3 command position, unit degree
J4	double	Direction 4 command position, unit degree
J5	double	Direction 5 command position, unit degree
J6	double	Direction 6 command position, unit degree

Example:

MoveJ,0,0,0,90,0,90,0,;

3. 60 MoveL

function: Robot moves straight to the specified space coordinates

format: **MoveL,rbtID,X,Y,Z,RX,RY,RZ,;**

parameter quantity: 7

successful return:

”MoveL,OK,;”

fail to return:

” MoveL,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	X direction command position, unit mm
Y	double	Y direction command position, unit mm
Z	double	Z direction command position, unit mm
RX	double	RX direction command position, unit degree
RY	double	RY direction command position, unit degree
RZ	double	RZ direction command position, unit degree

Example:

MoveL,0,450,0,450,180,0,-180,;

Note: there is a singular point in space motion.

3.61 MoveV

Function: The robot moves in speed. Note: the stop will stop when the shot is stopped.

format: **MoveV,rbtID,X,Y,Z,RX,RY,RZ,;**

parameter quantity: 7

successful return:

”MoveV,OK,;”

fail to return:

” MoveV,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	X-direction movement speed, Unit mm/second
Y	double	Y- direction movement speed, Unit mm/second
Z	double	Z-direction movement speed, Unit mm/second
RX	double	RX-direction movement speed, Unit degree/ second
RY	double	RY-direction movement speed, Unit degree/ second
RZ	double	RZ-direction movement speed, Unit degree/ second

Example:

MoveV,0,300, 300,450, 80,50, 80,;

Note: there is a singular point in space motion.

3. 62 MoveB

Function: Immediately change the end point of the robot's current movement to the specified space coordinate position

Format: **MoveB,rbtID,X,Y,Z,A,B,C,;**

Parameter quantity: 7

Successful return:

”MoveB,OK,;”

Fail to return:

” MoveB,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	X direction command position, unit mm
Y	double	Y direction command position, unit mm
Z	double	Z direction command position, unit mm
RX	double	RX direction command position, unit degree
RY	double	RY direction command position, unit degree
RZ	double	RZ direction command position, unit degree

Example:

MoveB,0,450,0,450,180,0,-180,;

;Sleep

MoveB,0,440,0,450,180,0,-180,;

;Sleep

MoveB,0,430,0,450,180,0,-180,;

Note: there is a singular point in space motion.

3. 63 MoveBJ

Function: Immediately change the end point of the robot's current movement to the specified angular coordinate position

Format: **MoveBJ,rbtID,X,Y,Z,A,B,C,;**

Parameter quantity: 7

Successful return:

”MoveBJ,OK,;”

Fail to return:

” MoveBJ,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
J1	double	Direction 1 command position, unit degree
J2	double	Direction 2 command position, unit degree
J3	double	Direction 3 command position, unit degree
J4	double	Direction 4 command position, unit degree
J5	double	Direction 5 command position, unit degree
J6	double	Direction 6 command position, unit degree

Example:

MoveBJ,0,0,0,90,0,90,0,;

;Sleep

MoveBJ,0,0,0,90,0,93,0,;

;Sleep

MoveBJ,0,0,0,90,0,96,0,;

3. 64 MoveBP

Function: Move in a linear motion to the specified angular coordinate position .

Format: MoveBP,rbtID,J1, J2, J3,J4,J5,J6,;

Parameter quantity: 7

Successful return:

”MoveBP,OK,;”

Fail to return:

” MoveBP,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
J1	double	Direction 1 command position, unit degree
J2	double	Direction 2 command position, unit degree
J3	double	Direction 3 command position, unit degree
J4	double	Direction 4 command position, unit degree
J5	double	Direction 5 command position, unit degree
J6	double	Direction 6 command position, unit degree

Example:

MoveBP,0,0,0,90,0,90,0,;

;Sleep

MoveBP,0,0,0,90,0,93,0,;

;Sleep

MoveBP,0,0,0,90,0,96,0,;

3. 65 MoveC

Function: Arc motion

Format: MoveC,rbtID,ViaCoord[3],GoalCoord[6] ,Type,;

Parameter quantity: 11

Successful return:

”MoveC,OK,;”

Fail to return:

” MoveC,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
ViaCoord[3]	double	Through position, X、Y、Z direction command position, unit mm
GoalCoord[6]	double	Target position, X、Y、Z、RX、RY、RZdirection command position, unit mm/degree
Type	int	Type of circle; 0= Unconstrained. 1= Fixed mode: The orientation of circle movement is defined by the first point. Then the robot will

		keep that orientation in the whole circle movement.
--	--	---

Example:

;Motion to the initial position

MoveJ,0,0,0,90,0,90,0,;

;Using the “ReadMoveState” instruction to judge the movement place

MoveC,0,410,-50,420,350,-100,420,180,0,180,0,;

Note: there is a singular point in space motion.

3. 66 MoveCL

Function: Circular motion

Format:

MoveCL,rbtID,ViaCoord[3],GoalCoord[6],dCircleCnt,Type,;

Parameter quantity: 12

Successful return:

”MoveCL,OK,;”

Fail to return:

” MoveCL,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
ViaCoord[3]	double	Through position, X、Y、Z direction command position, unit mm
GoalCoord[6]	double	Target position, X、Y、Z、RX、RY、RZdirection command

		position, unit mm/degree
dCircleCnt	double	Number of circles, count from one
Type	int	Type of circle; 0= Unconstrained. 1= Fixed mode: The orientation of circle movement is defined by the first point. Then the robot will keep that orientation in the whole circle movement.

Example:

;Motion to the initial position

MoveJ,0,0,0,90,0,90,0,;

;Using the “ReadMoveState” instruction to judge the movement place

MoveCL,0,410,-50,420,350,-100,420,180,0,180,0,;

Note: there is a singular point in space motion.

3. 67 MoveRelL

Function: Robot moves a certain distance from the specified spatial coordinate direction

Format: **MoveRelL,rbtID,directionID,direction,distance,;**

Parameter quantity: 4

Successful return:

”MoveRelL,OK,;”

Fail to return:

” MoveRelL,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
directionID	int	Spatial coordinate direction ID, count from zero
direction	int	Movement direction: 0=negative;1=positive;
distance	double	Relative motion distance, X, Y, Z unit mm, A, B,C unit degree

Example:

The Z direction moves in the direction of 10 millimetres in the direction of positive direction

`MoveRelL,0,2,1,10,;`

Note: there is a singular point in space motion.

3. 68 MoveRelJ

Function: Robot moves a certain distance from the specified angular coordinate direction

Format: `MoveRelJ,rbtID,directionID,direction,distance,;`

Parameter quantity: 4

Successful return:

`"MoveRelJ,OK,;"`

Fail to return:

`" MoveRelJ,Fail,ErrorCode,;"` , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
directionID	int	Angular coordinate direction ID, count from zero
direction	int	Movement direction:

		0=negative;1=positive;
distance	double	Relative motion distance, unit degree

Example:

The J3 direction moves 10 degrees in the direction of positive direction

`MoveRelJ,0,2,1,10,;`

3.69 MoveHoming

Function: Robot returns to the origin

Format: `MoveHoming,rbtID,;`

Parameter quantity: 1

Successful return:

`”MoveHoming,OK,;”`

Fail to return:

`” MoveHoming,Fail,ErrorCode,;”`, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

Example:

`MoveHoming,0,;`

3.70 MoveZ

Function: zigzagging

Format:

`MoveZ,rbtID,coordStart[6],coordEnd[6],coordVia[6],dWidth,nMoveZBaseCoord, dDensity, nUseZDensityint,;`

Parameter quantity: 22

Successful return:

”MoveZ,OK;”

Fail to return:

” MoveZ,Fail,ErrorCode;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
coordStart[6]	double	Starting position, unit:mm
coordEnd[6]	double	End position, unit:mm
coordVia[6]	double	Transition position, unit:mm
dWidth	double	Motion width,unit:mm
nMoveZBaseCoord	int	User coordinates index: 0=Using the current user coordinates 1=Plane coordinates determined using three points
dDensity	double	Motion density, unit:mm
nUseZDensityint	int	Use density: 0=Use the speed ratio to calculate the density 1=Use the dDensity parameter as the density

The speed ratio calculation density formula is as follows:

$$nUseZDensityint = \text{IntervalDist} \times \frac{0.008}{\text{VelocityRatir} * \text{MaxLineVelocity}}$$

IntervalDist: Spacing, default is 0.002

VelocityRatir: Speed ratio, such as 10% is 0.1

MaxLineVelocity: Maximum line speed, unit:mm/s

Example:

MoveZ,0,420,0,415,180,0,180,400,0,415,180,0,180,420,100,415,180,0,180,10,1,10,0,;

3. 71 MoveBEX

Function: Nonuniform motion

Format: MoveBEX,rbtID,X,Y,Z,RX,RY,RZ,dOverride,;

Parameter quantity: 7

Successful return:

” MoveBEX,OK,;”

Fail to return:

” MoveBEX,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
X	double	X direction command position, unit mm
Y	double	Y direction command position, unit mm
Z	double	Z direction command position, unit mm
RX	double	RX direction command position, unit degree
RY	double	RY direction command position, unit degree
dOverride	double	set speed ratio

Example:

MoveL,0,450,0,450,180,0,-180,0.1;

Note: there is a singular point in space motion.

3.72 StartServo

Function: Set the fixed update cycle and look-ahead time when starting the robot online control (servoJ or servoP).

Format: **StartServo,rbtID,servoTime,lookaheadTime,;**

Parameter quantity: 3

Successful return:

” StartServo,OK,;”

Fail to return:

” StartServo,Fail,ErrorCode,;”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
servoTime	double	The period of fixed location update is recommended to be greater than 0.015s
lookaheadTime	double	Prospective time, recommended between 0.05s and 0.2s

3.73 PushServoj

Function: Online joint position command control, the joint position is sent with the fixed update time set by StartServo, and the robot will track the joint position command in real time.

Note: The user needs to input a continuous track position. If there is no update position for more than 2 update cycles, the line control will be turned off.

Format: **PushServoJ,rbtID,jointPosition [6],;**

Parameter quantity: 7

Successful return:

” PushServoJ,OK,;”

Fail to return:

” **PushServoJ,Fail,ErrorCode,;**”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
jointPosition [6]	double	J1~J6 Direction command position, unit degree

3. 74 PushServoP

Function: The online terminal TCP position command control transmits the TCP position with the fixed update time set by the StartServo, and the robot reversely converts the target position of the target TCP position in real time.

Note: The user needs to input a continuous track position. If there is no update position for more than 2 update cycles, the line control will be turned off.

Format: **PushServoP,rbtID, tcpPosition [6],;**

Parameter quantity: 7

Successful return:

” **PushServoP,OK,;**”

Fail to return:

” **PushServoP,Fail,ErrorCode,;**”, error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
tcpPosition [6]	double	X、Y、Z、RX、RY、RZ Direction command position, X, Y, Z unit mm, A, B,C unit degree

3. 75 StartPushBlending

Function: Start PushBlending (continuous motion), initialize, you can start Push point

Format: **StartPushBlending,rbtID ;**

Parameter quantity: 1

Successful return:

” StartPushBlending,OK,;”

Fail to return:

” StartPushBlending,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

3. 76 PushBlendingL

Function: PushMoveL point.

Format: **PushBlendingL,rbtID,tcpPosition [6],Radius,;**

Parameter quantity: 8

Successful return:

” PushBlendingL,OK,;”

Fail to return:

” PushBlendingL,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
tcpPosition [6]	double	X、Y、Z、RX、RY、RZ Direction command position, X, Y, Z unit

		mm, A, B,C unit degree
Radius	double	Transition radius

3.77 PushBlendingC

Function: PushMoveL point.

Format: PushBlendingC,rbtID, ViatcpPosition [6], ViatcpPosition [6], Type,;

Parameter quantity: 8

Successful return:

” PushBlendingC,OK,;”

Fail to return:

” PushBlendingC,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero
ViatcpPosition [6]	double	X、Y、Z、RX、RY、RZ Direction command position of the transition point, X, Y, Z unit mm, A, B,C unit degree
EndtcpPosition [6]	double	X、Y、Z、RX、RY、RZ Direction command position of the target point, X, Y, Z unit mm, A, B,C unit degree
Type	Int	0=fixed attitude 1=unfixed attitude

3. 78 EndPushBlending

Function: End PushBlending, send the point to the controller and start motion, calculate error or parameter error need to re-Push .

Format: EndPushBlending,RbtID,;

Parameter quantity: 1

Successful return:

” EndPushBlending,OK,;”

Fail to return:

” EndPushBlending,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
rbtID	int	Robot index, count from zero

3. 79 HoldScriptFunc

Function: Pause running script

Format: HoldScriptFunc,;

Parameter quantity: 0

Successful return:

” HoldScriptFunc,OK,;”

Fail to return:

” HoldScriptFunc,Fail,ErrorCode,;” , error types see the error code list

Example:

HoldScriptFunc,;

3. 80 ContinusScriptFunc

Function: Continue running the script

Format: `ContinusScriptFunc,;`

Parameter quantity: 0

Successful return:

`"ContinusScriptFunc,OK,;"`

Fail to return:

`" ContinusScriptFunc,Fail,ErrorCode,;"`, error types see the error code list

Example:

`ContinusScriptFunc,;`

3. 81 Running function

Function: Running function

Format: `Function name,parameter1, parameter2,..., parameterN,;`

Parameter quantity: N

Successful return:

`" Function name,return character,;"`

Fail to return:

`" Function name,Fail,Error Message,;"`, error types see the error code list

Example:

`;The function F_test has been compiled`

`F_test,5,;`

3. 82 ShutDown

Function: Remote shutdown .

Format: ShutDown,Type,;

Parameter quantity: 1

Successful return:

” ShutDown,OK,;”

Fail to return:

” ShutDown,Fail,ErrorCode,;” , error types see the error code list

Parameter detail:

Parameter name	type	Meaning
Type	int	0=shutdown, 1=restart

4 ErrorCode

ErrorCode	Meaning
10000	Short circuit error
10001	Over voltage limit error
10002	Under voltage limit error
10003	Over velocity limit error
10004	Execute error
10005	Over current error
10006	Encoder error
10007	Following position error
10008	Following velocity error
10009	Negative limit error
10010	Positive limit error
10011	Server over heating error
10012	Max current error
10013	Emergency stop error
10014	UDM error
10015	Server parameter error
20000	Controller is not started
20001	Master is not started
20002	some slave is dropped
20003	Robot on safe stop state
20004	Robot on physical stop state
20005	Robot out safe space
20006	Robot enable time out
20007	Robot not electrify
20008	Starting master station error
30000	Collision shutdown
30001	Robot Collide with body
30002	Over joint limit error

30003	Singularity error
30004	General stopping criterion
30005	calculate failed
30006	UDM Status Error
30007	Init slave Error
30008	Home Step2 Error
30009	Out Of Direction Limit Error
30010	Out Of Direction Current Error
30011	Wrong load or mounting angle
30012	Motor limit temperature exceeded
1001	The robot has not been initialized
1002	Master station has not been started
1003	Slave station drop off
1004	The robot is safely locked
1005	The physical stop
1006	Robot has not been servo on
1007	Error reporting from slave station
1008	Robot beyond safe space
1009	In robot motion
1010	Invalid command
1011	Parameter error
1012	Function call format error
1013	Waiting for command execution
1014	IO does not exist
1015	Robots do not exist
1016	No connection server
1017	Network timeout
1018	Connection failed
1019	Serial connection failed
1020	No zero position is set
1021	The last same command has not been

	completed
1022	Serial port Di is empty
1023	Serial port DO is empty
1024	Wait timeout
1025	Error status
1026	Stop robot
1027	Robot has been servo off
1028	Robot has been servo on
1029	Function has not been enabled
1030	Start master timeout
1031	The robot has not been powered on
1032	Serial port has not been started
1033	The simulation state command is invalid
1034	RTOS Library not exist
1035	DCS Handle Command thread crash
1039	Script running
1040	Xml Param Error
1041	System Board Not Connect
1042	Controller Not Start
1043	Controller Status Error
1044	Robot in TeachMode
1045	Robot Already Electrify
1046	Connect to Modbus Failed
1047	Master is Started
1048	Parameter over specified payload
1049	DCS Status Error
1050	Target position invalid
1051	Robot Drive Operating
1052	Start Master Error
1053	Initialize slaves Error!HomeStep2 Fail
1054	ModbusRTU disconnected state
1055	ModbusRTU is busy
1056	Blending didn't start

1057	Blending is not over
2000	Failed to load library
2001	The script is empty
2002	Compile error
2003	Reload script error
2004	Function does not exist
2005	Function return type error
2006	MissSignal1
2007	MissSignal2
2008	Parameter type error
2009	There is no header file included
2010	No return value
2012	UDM Stack Err
2013	Script been lock,maybe compiling
2014	Not In RunScript Statu
2015	Serial Close
2016	Serial Close
2017	Controller not started
2018	Socket Not Connected
2020	Function Name have Space.
2021	Socket Error
2022	Function broken stop.
2023	Timer running error
2024	Enable SwitchON key error

5 Terminology

Terminology	Explain
ACS	Joint coordinates, unit degree
PCS	World coordinate, unit mm、 unit degree

6 Control robot motion process—Sample

6.1 Flow chart

When the robot has been servo on, the robot can move to the target position by sending a series of move instructions and a short and long moving instructions. The following points need to be noted in the process of sending motion instructions.

1. Only when the current motor command of the robot is successfully completed can the next movement command be issued(The robot motion state is obtained by sending the robot motion state command " ReadMoveState ").
2. When the return of the command that calls the robot's motion state is in motion, The upper monitor should be called periodically to obtain instructions about the robot's motion state until the robot's motion state is identified. such as, the motion is completed or faulty.
3. When the control system returns the robot motion error, the upper monitor should call GrpReset command to clean the control system. Only when the error is successful can the call to the next movement be continued.
4. If the control system returns to the robot movement is complete, you can continue with the next movement instruction, then repeat 2-4.

The specific process is shown below:

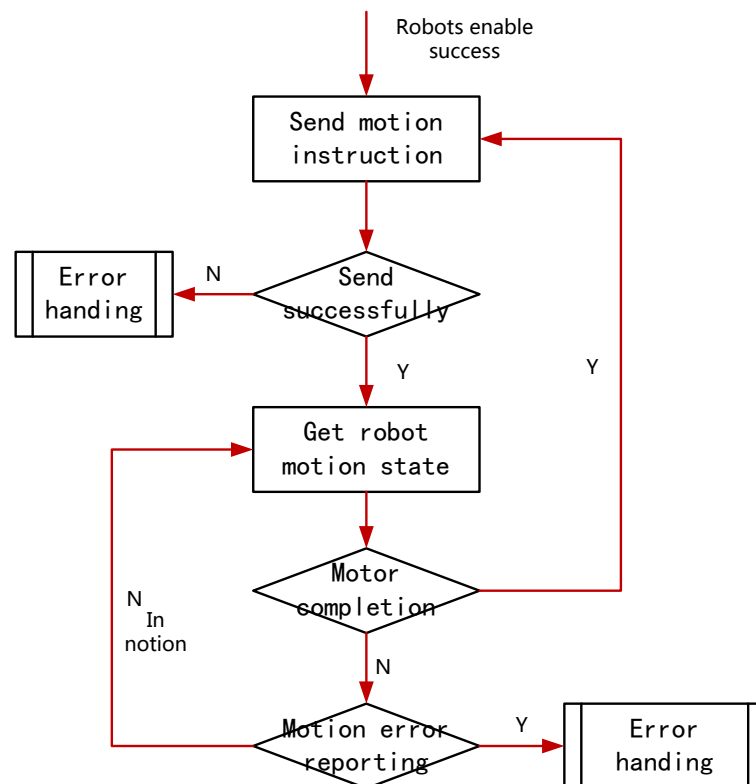


Fig Flow chart of robot motion

6.2 Sample code

Pseudo code is as follows, for reference only:

// Waiting for the movement to complete

int WaitMotionFinish()

{

while(1)

{

// 1. Send get robot motion state message: ReadMoveState,;

// 2. Parsing the returned motion state: MoveState

if(MoveState == 1009)

{

// In motion, wait for 10 milliseconds, query again

Sleep(10);

}

else if(MoveState == 0)


```
        {  
            // The movement is completed and is returned directly to  
the zero.  
            return 0;  
        }  
    else  
    {  
        // Other error conditions are dealt with separately  
        return nErrCode;  
    }  
}  
return 0;  
}
```