commonplace robotics GmbH

Robot Interface CRI

V17 - February 23th, 2021
CPRog Version: V902-11-024
TinyCtrl Version: V980-11-104

Change Log

- V17 / February 23th, 2021: Added Get/SetActive commands for multi client support, changed mode "joint" in STATUS message, added PROG RELATIVETOOL
- V17 / February 3rd, 2021: Added licensing commands, zero torque mode and GSIG
- V17 / January 12th, 2021: Added additional external axis config parameters, added PLC interface input inPause, PROG commands and responses implemented
- V17 / November 10th, 2020: Added virtual box error codes to KINSTATE
- V17 / July 1st, 2020: SetVariableSingle, SetVariablePosCart and SetVariablePosJoint implemented
- V17 / April 29th, 2020: GetNrVariable and GetPosVariable implemented in TinyCtrl
- V16 / January 7th, 2020: Additional RunState status message, GetReferencingInfo in CPRog, ProgramReplayMode in CPRog, GetVersion command in CPRog and TinyCtrl, GetProgramName returns Folders as well, LoadProgram open folders aswell
- V15 / April 24th, 2019: Request SystemVariables and Temperature, Changes in the Variables area. LogMessage transfer added.
- V14 / Feb. 22nd, 2019: Request/Set Variables, Start Program with Offset
- V13: UploadProgram renamed to UploadFile, changed Functionality. Referencing added. Errorcodes adapted. Referencing information added. EXECACK adapted.

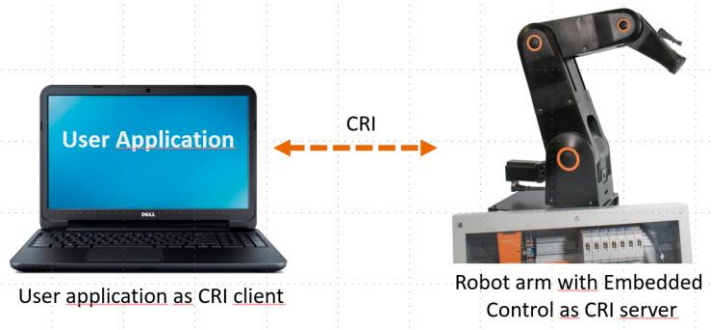
1. Summary

The CRI ethernet interface allows to interact with the iRC igus Robot Control and CPRog robot controls. It is available with the same interface both on the Windows software iRC or CPRog, and the Linux software of the embedded robot control.

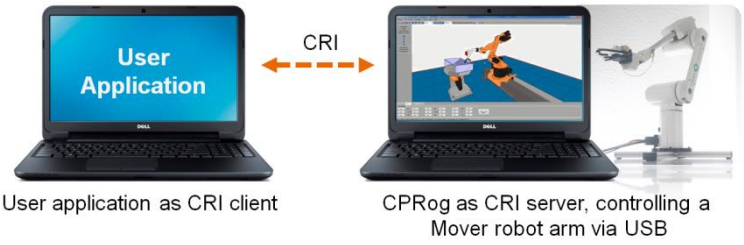
The robot control establishes a socket server which allows a user application to connect via LAN or WLAN. The connected user application is able to remotely control the robot, including jogging, changing variable values and defining / starting robot programs.

For development and testing the iRC / CPRog simulation environment can be used. For real operation only the IP address has to be changed to control the physical robot arm with an embedded control.

For the development of applications, the user application can interact with iRC igus Robot Control / CPRog. The commands are then executed by the simulated robot arm.



With the same interface a robot arm or gantry with embedded control can be addressed.



Robots without embedded control, e.g. the Mover robots, can also be commanded using the CRI interface.

In the further explanations only CPRog is mentioned as robot control software. All functionalities are the same with the iRC igus Robot Control.

2. Example Software

An example client software is available via Github:

https://github.com/CommonplaceRobotics/CRI-DemoClient

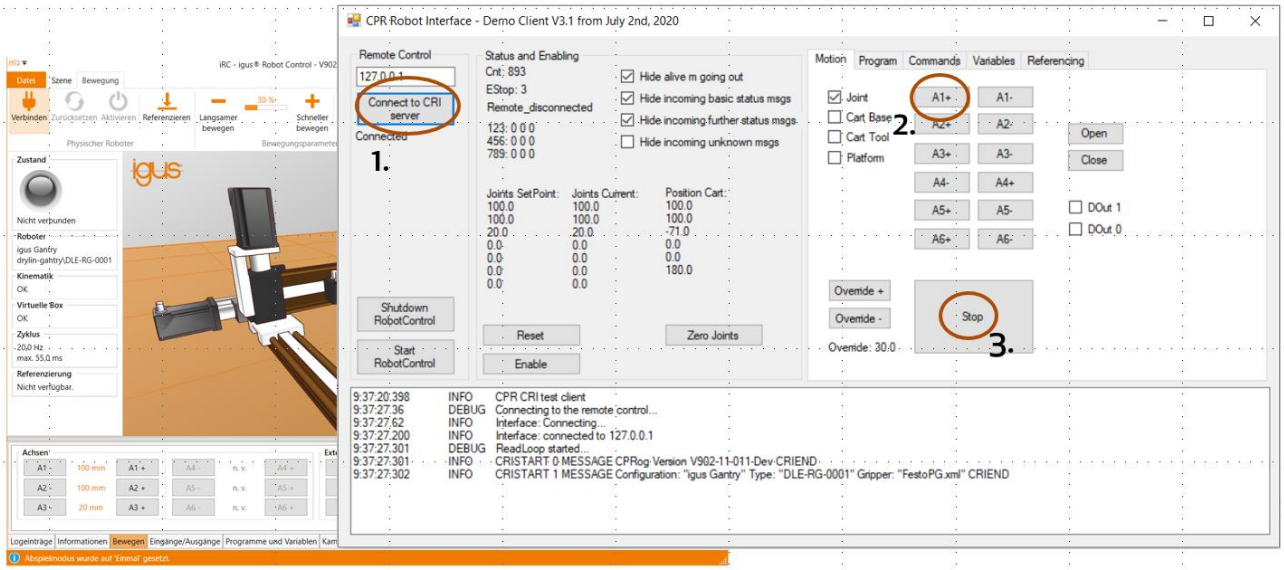
It is provided in C# source code for Microsoft Visual Studio. The software shows how to connect, how to send the control commands and how to parse the robots answers. For a stable operation of course more means regarding e.g. fault detection and error recovery have to be taken.

IP addresses are:

- Local host: 127.0.0.1
- Embedded board: 192.168.3.11, login as root without password

The client computer needs to have an IP in the same subnet, e.g. 192.168.3.1. Step by step instructions how to change the IP address can be found in the net.

A first test operation can be done with 3 steps. As preparation start CPRog and the CRI test client. The loaded CPRog project must have the CRI server activated, see chapter 3.2.



The steps are:

1. Connect the CRI test client with CPRog.
2. Choose the motion type: joint motion or linear motion in base or tool coordinate system.
3. Press e.g. the Z- button one or several times. The simulated robot will move in axis 1. Each time you press the button the velocity will increase by 10%.
4. Press the Stop button to stop all jog motions. Start the loaded robot program with the Start button.
5. For interaction with the physical robot the reset and enable buttons enable the operation.

3. Multi-Client support

Starting with CPRog/iRC V902-V12 and TinyCtrl V980-V12 multiple concurrent CRI connections are supported. Of these connections one can be active, all others are passively listening to the robot state without being able to change it.

If no active connection is present the next connection will be set active. This means that if the active client disconnects no passive client will be promoted to active state but the next new connection will be set active. Use the CMD GetActive command to check if the connection is active and the CMD SetActive command to request a passive connection to become active. Listen to the CMD Active message for state changes.

4. Definition of the CRI Interface

4.1 Interface Set Up

To allow communication the CRI server on the robot or the simulated robot needs to be activated.

On the embedded control the CRI server is active by default. In the iRC / CPRog software the CRI server needs to be activated in the backstage area as in the following picture.

- ➔ File / Interfaces / CRI Interface
- ➔ For a connection on the local computer the IP should be forced to 127.0.0.1
- ➔ Then the server can be started. The project should be saved to store these settings.
- ➔ Now the CRI demo client can connect to the CRI server.

4.2 Description

- The robot controller sets up a server. Clients can connect on Port 3920.
- When the robot controller does not receive at least one alive message every 2 seconds it will close the connection.
- 1 seconds after closing a connection the server is ready to connect to a client again.
- The operating system needs to allow the client-server connection. The according settings e.g. regarding firewalls have to be established.
- Messages to and from the server follow the scheme:
  "CRISTART counter CMD_CATEGORY command_details CRIEND"
  All messages from the server do have an incrementing sCnt as first parameter, the messages from the client an independent cCnt. Both are incremented with each message from 1 to 9999, then reset to 1.
- All values are sent in US style (with points as deliminator) and in mm and degree as units.

4.3 Alive Message and Status Answer

The alive message is necessary in constant time intervals, otherwise the server will declare the connection as dead and disconnect. It also defines the current jog values for the 6 arm joints and 3 gripper joints. The jog values are float numbers in the range of [-100.0 .. 100.0]
The status answer and runstate answers from the server are generated periodically.
Both messages will not trigger an answer.

In case of an error the server will send an error message. Reason can be e.g. joint limit switches, hand singularities or similar reasons possible during jog operation:

```
CRISTART cCnt JOGERROR errordescription CRIEND (to be implemented)
```

Alive Message from Client to server:

```
CRISTART 1234 ALIVEJOG 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0 90.0 CRIEND
CRISTART cCnt ALIVEJOG ja1  ja2  ja3  ja4  ja5  ja6  je1  je2  je3  CRIEND
```

Status Answer from Server to Client (implemented in one line):

```
CRISTART 1234 STATUS MODE joint
POSJOINTSETPOINT 1.00 2.00 3.00 …. 15.00 16.00
POSJOINTCURRENT 1.00 2.00 3.00 …. 15.00 16.00
POSCARTROBOT 10.0 20.0 30.0 0.00 90.00 0.00
POSCARTPLATFORM 10.0 20.0 180.00
OVERRIDE 80.0
DIN 0 DOUT 0
ESTOP 3 SUPPLY 23000 CURRENTALL 2600
CURRENTJOINTS 150 200 …  140 160
ERROR no_error 8 8 8 … 8 8 8
KINSTATE 3
CRIEND
```

Explanations for the status answer:

- The different modes are:
  - "joint" The jog values move the robot arm (1-6) and the up to 3 additional joints in joint space. In CPRog/iRC V902-11 and older the gripper is controlled instead of additional joints.
  - "cartbase" The jog values move the robot arm in Cartesian space (base coordinate system) and the gripper in joint space.
  - "carttool" The jog values move the robot arm in Cartesian space (tool coordinate system) and the gripper in joint space.
  - "platform" A mobile platform is jogged with velocities in X, Y and RZ
  - "fsm" Jog not supported in this mode
- In POSJOINT, CURRENTJOINTS and ERROR always 16 values are provided (values are zero if the joints are not available):
  - 6 values for the robot arm joints
  - 3 values for the gripper
  - 3 values for the external joints

  - 4 values for the mobile platform joints
- Joint positions as floating point: POSJOINTSETPOINTS contains the set point joint positions in degree. POSJOINTCURRENT contains the current physical joint positions.
- Cartesian positions: values are provided in mm and degree
  - POSCARTROBOT: the XYZ and ABC values of the robot arms TCP
  - POSCARTPLATFORM: Position XY and rotation RZ of the mobile platform. These values are derived by the wheel odometry and not reliable because of drift and slipping effects.
- Override: The override value from 0 to 100 (floating point)
- DIN and DOUT: Current status of digital Inputs / Outputs, if available. Binary coding.
- ESTOP: Status of Emergency Stop (bit 1) and Main Relay (bit2). 3 means ok.
- SUPPLY: Level of the main supply in V
- CURRENTALL: Motor current for all joints measured by the safety board in mA. The current of the electronics (Linux board and joint controller) is not included.
- CURRENTJOINTS: Motor current of the single joints in mA
- ERROR: One combined error value as string and 16 single byte joint error codes. The errors are:
  - Bit 1: Temp - Overtemperature
  - Bit 2: EStop/LowV - Supply too low
  - Bit 3: MNE Motor not enabled
  - Bit 4: COM - Communication watch dog
  - Bit 5: POS - Position lag
  - Bit 6: ENC - Encoder error
  - Bit 7: OC - Overcurrent
  - Bit 8: DRV – DriverError / SVM
- KINSTATE: Kinematic Status
  - 0: No error
  - 13,14 Joint Limit Min, Max
  - 21,23,24 Cartesian Singularities Center, Reach, Wrist
  - 30-35 Tool reached virtual box limit
  - 99 Motion not allowed, e.g. due to boards not enabled

Runstate Answer from Server to Client (implemented in one line):

```
Example:
CRISTART 1234 RUNSTATE testmotion.xml 12 3 0 2 CRIEND
The robot runs the program “testmotion.xml”, currently command 3 of 12 is
active. The current runstate is 0 (stopped) and the replaymode is 2 (single
step).
```

- The runstates are:
  - 0 Stopped
  - 1 Paused
  - 2 Running
- The replay modes are:

- 0 Single
- 1 Repeat
- 2 Step
- 3 Fast

4.4 Robot Commands

Robot commands are send to operate the robot arm, e.g. to enable the motors, to choose the operation mode or to start and stop robot programs. The general format is:

```
CRISTART cCnt CMD commandname [parameter] CRIEND
```

The server sends an acknowledge or an error after receiving the command, the cnt number is the same as in the command message:

```
CRISTART sCnt CMDACK ref_to_cCnt CRIEND

CRISTART sCnt CMDERROR ref_to_cCnt error_description CRIEND
```

```
CRISTART 1234 CMD Connect CRIEND
CRISTART 1234 CMD Disconnect CRIEND
CRISTART 1234 CMD Reset CRIEND
CRISTART 1234 CMD Enable CRIEND
CRISTART 1234 CMD Disable CRIEND
CRISTART 1234 CMD SetJointsToZero CRIEND
```

```
Only for robolink / drylin robots with reference switches:
CRISTART 1234 CMD ReferenceAllJoints CRIEND
Requests a referencing of all joints
CRISTART 1234 CMD ReferenceSingleJoint j CRIEND
Requests the referencing of a single joint.
j is the joint number starting at 0 for the first joint
CRISTART 1234 CMD ReferenceSingleExternalJoint j CRIEND
Requests the referencing of an external joint, e.g. an additional linear axis.
j is the joint number starting at 0 for the first external joint
CRISTART 1234 CMD GetReferencingInfo CRIEND
Sends information on the current status of the joints, if they are referenced
or not. The server provides the status of the robot joints, and, if available,
external joints. The information contains the general status and 12 values for
the single joint referencing status.
Example answer:
CRISTART 1234 INFO ReferencingInfo 1 Joints 1 1 1 1 1 0 0 0 0 0 0 0 CRIEND
Meaning: The value before "Joints" describes the robot status: 1 means that all
required joints are referenced. 0 means that at least one required joint is not
referenced.
The 12 values behind "Joints" are the referencing status of the single joints.
In this case the first 5 joints are referenced.
```

```
CRISTART 1234 CMD MotionTypeJoint CRIEND
CRISTART 1234 CMD MotionTypeCartBase CRIEND
CRISTART 1234 CMD MotionTypeCartTool CRIEND
CRISTART 1234 CMD MotionTypePlatform CRIEND       - only with mobile base
```

```
Available with CPRog/iRC V902-12 and TinyCtrl V980-12 or newer
CRISTART 1234 CMD ZeroTorque True/False
Requests to enable/disable the zero torque mode. This has no effect if this
mode is not enabled in the robot configuration file. Motor modules must support
torque mode. After disabling the motors are in Disabled state.
CRISTART 1234 CMD ZeroTorque
Requests information on the zero torque mode.
Both commands send the following reply:
CRISTART 1234 CMD ZeroTorque <allowed> <enabled>
Both parameters are True/False. <allowed> shows whether zero torque mode is
```

enabled in the robot configuration, <enabled> shows whether it is enabled.

CRISTART 1234 CMD DOUT 3 true CRIEND
Sets the state of a digital output. The DOut number is 0 .. 63. This has no effect if a program is running.

Available with CPRog/iRC V902-12 and TinyCtrl V980-12 or newer
CRISTART 1234 CMD GSIG 3 true CRIEND
Sets the state of a global signal. The GSig number is 0 .. 99.
A status message in the following format is sent periodically:
CRISTART 1234 GSIG lower upper CRIEND
lower and upper are 64 bit decimal numbers encoding the global signal states.

CRISTART 1234 CMD Override ovr CRIEND
Sets the override for jog motion and replay.
ovr is a floating point value from 0.0 to 100.0

CRISTART 1234 CMD GetVersion CRIEND
Requests the Software name ("TinyCtrl" or "CProg")from the server along with the protocol version implemented by the server. Example answer:

CRISTART 1234 INFO Version CPRog 16 CRIEND
In this case the server is CPRog and implements CRI-Commands up to protocol version 16.

CRISTART 1234 CMD StartProgram CRIEND
CRISTART 1234 CMD StopProgram CRIEND
CRISTART 1234 CMD PauseProgram CRIEND
CRISTART 1234 CMD ProgramReplayMode replayMode CRIEND
replayMode is an integer defining the replay mode for the current and further programs started.
replayMode = 0 Single replay
replayMode = 1 Repeated replay
replayMode = 2 Stepwise replay, the replay is paused after each command and has to be continued with the *StartProgram* command.

Available on TinyCtrl only.
CRISTART 1234 CMD StartProgramWithOffset j0 .. j8 x y z a b c CRIEND
Starts a program with the defined offset values. The offset values are added to the target positions of Joint/JointbyVariable and Linear/LinearbyVariable motions. They do not affect relative motions.
The offsets are valid until another program is loaded, the automatic restart or the manual start do not change the offsets. The use of the StartProgram command as above (e.g. when pressing on Start in CPRog) or loading a program will reset the offset to zero.
Setting and resetting the offsets will generate a logfile entry.
Attention: There is no check for validity of the values, if the values are e.g. too high the program will stop with an out-of-reach error or similar.

j0 .. j8 are the 9 floating point offsets for the 9 possible joints
x .. c are the floating point offsets for the cartesian coordinates

Example:
CRISTART 6789 StartProgramWithOffset 10 -10.0 5 -5.0 0.0 0.0 0.0 0.0 0.0 30.0 20.0 10.0 0.0 0.0 0.0 CRIEND

CRISTART 1234 CMD LoadProgram progName CRIEND
Load a program file from disk into the robot controller. progName is the name in the Directory /Data/Programs/, e.g. "test.xml". Programs loaded before are erased. The program is loaded as program 0. From CRI version 16 onwards, this command can also be used to open a folder.

CRISTART 1234 CMD LoadLogicProgram progName CRIEND
Available in CRI version 16 and newer
Load a logic program file from disk into the robot controller. progName is the name in the Directory /Data/Programs/, e.g. "test.xml". Programs loaded before are erased. The program is loaded as program 0.

CRISTART 1234 CMD DeleteProgram CRIEND
Removes all loaded programs from the robot control (not from the disk). This commands should be called before assembling a new program with Add-commands.

CRISTART 1234 CMD DeleteLogicProgram CRIEND
Removes all logic programs from the robot control (not from the disk).

CRISTART 1234 CMD DeleteProgramFromFile progName CRIEND
Available on TinyCtrl only.
Deletes the file progName in the /Data/Programs/ directory permanently. It is removed from the disk and cannot be restored.
"progName" has to be the filename of the program, e.g. "testmotion.xml". The path /Data/Programs/ is added automatically.

CRISTART 1234 CMD GetProgramInfo CRIEND
Sends an answer with the currently active programs name, the number of commands in the program and the currently active command.
Example:
CRISTART 1234 INFO ProgramInfo testmotion.xml 12 3 CRIEND
The robot runs the program "testmotion.xml", currently command 3 of 12 is active.

CRISTART 1234 CMD GetLogicProgramInfo CRIEND
Available in CRI version 16 and newer
Sends an answer with the currently active logic programs name, the number of commands in the program and the currently active command.
CRISTART 1234 INFO LogicProgramInfo testmotion.xml 12 3 CRIEND

CRISTART 1234 CMD GetProgramName which CRIEND
Available on TinyCtrl only.
Sends an answer with the number of files in directory /Data/Programs/, and the filename of one of the files. From CRI version 16 onwards this command also enumerates folders (distinguished by a trailing slash character).
Example:
CRISTART 1234 INFO ProgramName testmotion.xml 0 7 CRIEND
The first of seven programs has the name "testmotion.xml"

-- Changed Command Name and behavior due to broader focus --
CRISTART 1234 CMD UploadFileInit fileName nrOfLines CRIEND
To upload a file to the /Data/ directory a combination of the commands *UploadFileInit*, *UploadFileLine* and *UploadFileFinish* has to be send. These commands only write the file to the drive. The file has to be loaded afterwards using *LoadProgram*.
UploadFileInit initializes the upload. file*Name* is a string with the file name relative to the /Data/ directory. The path is added by the server.
nrOfLines is an integer with the number of *UploadFileLine* commands following. Between sending the lines there should be small breaks to allow sending the Alive messages.

-- Changed Command Name due to broader focus --
CRISTART 1234 CMD UploadFileLine fileLine CRIEND
For every line of the original file one command *UploadFileLine* has to be send. *fileLine* is a string with the line to be written.

-- Changed Command Name due to broader focus -- **CRISTART 1234 CMD UploadFileFinish CRIEND** This command finishes the file upload by closing it. It also compares the received number of lines with the anticipated number.
Supported in CPRog/iRC/TinyCtrl V12 and newer **CRISTART 1234 CMD GetActive CRIEND** **CRISTART 1234 CMD SetActive true/false CRIEND** GetActive requests the active/passive state of the CRI connection. If the connection is passive all commands that change the state of the robot control will be ignored. SetActive requests the CRI connection to set active or passive, in the former case all connections to other clients will be set passive. The following response will be sent by both requests and on state change: **CRISTART 1234 CMD Active true/false CRIEND**

4.5 System Commands

CRISTART 1234 SYSTEM Shutdown 99 CRIEND Only on TinyCtrl. Stops the TinyCtrl software. The number after "Shutdown" has to be 99 to work.
CRISTART 1234 SYSTEM GetBoardTemp CRIEND Provides the temperatures of the motor control PCBs. Example Request: CRISTART 1234 SYSTEM GetBoardTemp CRIEND Example Answer: CRISTART 6789 INFO BoardTemp temp1 temp2 ... temp16 CRIEND temp1 to temp16 are the PCB temperatures of the joint 1 to 16. The values are °C transmitted as floating point numbers.
CRISTART 1234 SYSTEM GetMotorTemp CRIEND Not implemented yet.

4.6 Log Messages

The remote systems can transfer its log messages to the CRI client. This functionality can be switched on or off in the ini files of the remote system.

Implemented in TinyCtrl, currently not in CPRog **CRISTART 1234 LOGMSG msgString CRIEND** Provides the log messages from the remote system. They can be used e.g. for diagnostics.

4.7 Handling of Variables

CRISTART 1234 VAR GetNrVariable variableName CRIEND
Sends an answer with the value of a number variable. The result is one float value. The variable has to be available in the robot control, that means they must have been defined in the running robot program.
Please refer to CommandReference.pdf for details on variables.
Example of an answer:
CRISTART 1234 VARINFO ValueNrVariable currentRow 3.0 CRIEND
The variable currentRow has the value 3.0.
Example of an answer in case of an error, e.g. when the variable is not known:
CRISTART 6789 VARERROR ValueNrVariable currentRow variable_not_known CRIEND

CRISTART 1234 VAR GetPosVariable variableName CRIEND
Sends an answer with the value of a position variable. The result is 15 float values. The variable has to be available in the robot control, that means they must have been defined in the running robot program.
Please refer to CommandReference.pdf for details on variables.
Example of an answer (x-z, A-C, a1-a6 and e1-e3 will be float values):
CRISTART 1234 VARINFO ValuePosVariable currentPos x y z A B C a1 a2 a3 a4 a5 a6 e1 e2 e3 CRIEND
Example of an answer in case of an error, e.g. when the variable is not known:
CRISTART 6789 VARERROR ValuePosVariable currentPos variable_not_known CRIEND

CRISTART 1234 VAR GetSystemVariable variableNumber CRIEND
Sends an answer with the value of a system variable. The result is one int value.

System variables are (number, name and meaning):

0	UpTimeComplete	Time the control is running [Minutes]
1	UpTimeLast	Time the control is running since the last start [Minutes]
2	UpTimeEnabled	Time the control was in "NoError" state [Minutes]
3	UpTimeMotion	Time the robot has been moving [Minutes]
4	ProgramStarts	Number of starts of the main program
10	JointCycles	Number of direction changes in joint 1
..		
18	JointCycles	Number of direction changes in joint 9

Example of an answer:
CRISTART 1234 VARINFO ValueSystemVariable 0 Value 10429 CRIEND
The control has been running for 10429 minutes.
Example of an answer in case of an error, e.g. when the variable is not known:
CRISTART 6789 VARERROR ValueSystemVariable 22 variable_not_known CRIEND
The system variable number 22 is not known.

Implemented in CPRog V902-11-011 and TinyCtrl V980-11-088 and newer
CRISTART 1234 VAR SetVariableSingle variableName newValue CRIEND
Sets the value of a variable. The argument variableName can be the name of a number variable (e.g. nrOfRows). It cannot be an element of a position variable (e.g. targetPos.x).
The variables have to be available in the robot control, that means they must have been already defined in the running robot program.
Please refer to CommandReference.pdf for details on variables.
Standard answer is an acknowledgement:
CRISTART 6789 CMDACK 1234 CRIEND
Example of an answer in case of an error, e.g. when the variable is not known:
CRISTART 6789 CMDERROR 1234 variable_not_known CRIEND

Implemented in CPRog V902-11-011 and TinyCtrl V980-11-088 and newer
CRISTART 1234 VAR SetVariablePosCart variableName x y z a b c e1 e2 e3 CRIEND
Sets the value of a variable. The argument variableName is the name of a position variable (e.g. targetPos). The Cartesian elements of the variable are updated. The joint elements of the variable remain on their old values.
The variables have to be available in the robot control, that means they must have been already defined in the running robot program.
Please refer to CommandReference.pdf for details on variables.
Standard answer is an acknowledgement:
CRISTART 6789 CMDACK 1234 CRIEND
Example of an answer in case of an error, e.g. when the variable is not known:
CRISTART 6789 CMDERROR 1234 variable_not_known CRIEND

Implemented in CPRog V902-11-011 and TinyCtrl V980-11-088 and newer
CRISTART 1234 VAR SetVariablePosJoint variableName j0 j1 .. j5 e1 e2 e3 CRIEND
Sets the value of a variable. The argument variableName is the name of a position variable (e.g. targetPos). The joint elements of the variable are updated. The cartesian elements of the variable remain on their old values.
The variables have to be available in the robot control, that means they must have been already defined in the running robot program.
Please refer to CommandReference.pdf for details on variables.
Standard answer is an acknowledgement:
CRISTART 6789 CMDACK 1234 CRIEND
Example of an answer in case of an error, e.g. when the variable is not known:
CRISTART 6789 CMDERROR 1234 variable_not_known CRIEND

4.8 Defining a Robot Program

The following messages add robot commands to the currently loaded program on the robot control. To take effect the robot program must be started using the CMD messages (start, pause, stop). The deletion of all commands can also be done using the CMD message.

With the cmdCnt number the client can provide an id to the program command. This id is used when there are further messages regarding the program command, e.g. error or execution acknowledgments.

The server sends an acknowledgement or an error after receiving the message, the cnt number is the same as in the command message.

```
CRISTART sCnt PROGACK ref_to_cCnt ref_to_cmdCnt CRIEND

CRISTART sCnt PROGERROR ref_to_cCnt ref_to_cmdCnt errordescription CRIEND
```

Currently the following error descriptions are used: `unknown_command`, `incomplete_argument`, `could_not_parse`, system_error

CRISTART cCnt PROG cmdCnt JOINT j0 … j5 EXT j6…j8 VEL velpercent CRIEND Example: CRISTART 1234 PROG 42 JOINT 10.00 20.00 30.00 40.00 50.00 60.00 EXT 70.0 80.0 90.0 VEL 20.0 CRIEND Adds a joint command to the current robot program. The joint values (degree, floating point) for the 6 robot joints and 3 additional joints are defined. The VEL parameter (percent [0..100] floating point) defines the joint velocity in percent. During replay all joints move with a constant velocity to the set point values, the velocities depend on the joint with the longest travel time.
Implemented in CPRog V902-11-024 and TinyCtrl V980-11-102 and newer **CRISTART cCnt PROG cmdCnt RELATIVEJOINT j0 … j5 EXT j6…j8 VEL velpercent CRIEND** Example: CRISTART 1234 PROG 42 RELATIVEJOINT 10.00 20.00 30.00 40.00 50.00 60.00 EXT 70.0 80.0 90.0 VEL 20.0 CRIEND Adds a relative joint command. The parameters are as described for the JOINT command.
CRISTART 1234 PROG cmdCnt LINEAR x y z a b c EXT j6…j8 VELMMS velmms CRIEND Example: CRISTART 1234 PROG cmdCnt LINEAR 10.0 20.0 30.0 40.0 50.0 60.0 EXT 70.0 80.0 90.0 VELMMS 20.0 CRIEND Adds a linear command to the current robot program. The x-z, a-c values define a cartesian base coordinate, j6-j8 define additional joints. The VEL parameter defines the linear velocity in mm/s.
Implemented in CPRog V902-11-024 and TinyCtrl V980-11-102 and newer **CRISTART 1234 PROG cmdCnt RELATIVELINEAR 50.0 50.0 50.0 125.0 CRIEND** Adds a linear command that does a movement relative to the base coordinate system. The values are x, y and z cartesian coordinates, the fourth value is the motion speed in mm/s. All values are float.
Implemented in CPRog V902-11-025 and TinyCtrl V980-11-104 and newer **CRISTART 1234 PROG cmdCnt RELATIVETOOL 50.0 50.0 50.0 125.0 CRIEND** Adds a linear command that does a movement relative to the tool coordinate system. The values are x, y and z cartesian coordinates, the fourth value is the motion speed in mm/s. All values are float.

CRISTART 1234 PROG cmdCnt GRIPPER grpJoint1 jrpJoint2 grpJoint3 CRIEND Example to open the gripper: CRISTART 345 PROG 81 GRIPPER 100.0 0.0 0.0 CRIEND Adds a Gripper statement to the end of the current robot program. *cmdCnt* is an integer as reference to the command *grpJoint1 to 3* are floating point values for the gripper joints ranging from 0.0 to 100.0. A maximum of 3 gripper joints can be commanded (currently only one is supported!). For single joint gripper only the first value is used. **The robot control does not wait for the execution of this command,** the next command is issued in the next cycle. It might be necessary to add a wait command after the gripper command e.g. to ensure that the workpiece gets gripped.
CRISTART 1234 PROG cmdCnt WAIT timeInMS CRIEND Example to wait 5 seconds: CRISTART 827 PROG 23 WAIT 5000 CRIEND Adds a wait statement to the end of the current robot program. *cmdCnt* is an integer as reference to the command *timeInMS* is the wait time in milliseconds
Implemented in CPRog V902-11-024 and TinyCtrl V980-11-102 and newer **CRISTART 1234 PROG cmdCnt DOUT doutNum true CRIEND** Adds a command that enables or disables a digital output. The doutNum is offset by 1 in the CPRog UI, this means that DOut21 in CPRog is number 20 in this command.

4.9 Execution of Robot Programs

Available in CPRog: n/a
Available in TinyCtrl: from Version V980-04-041

After the start of a robot program the server sends messages regarding the current execution status to the client, e.g. "just started execution of command nr 462" or "just reached end of linear motion nr 90".

```
CRISTART sCnt EXECACK cmdNr progNr CRIEND
```

When the program is paused the following message is generated:

```
CRISTART sCnt EXECPAUSE cmdNr progNr CRIEND
```

When the program execution ends an according message is generated:

```
CRISTART sCnt EXECEND cmdNr progNr reason CRIEND
```

In the case of an error during execution an according message is generated:

```
CRISTART sCnt EXECERROR cmdNr progNr errordescription CRIEND
```

Parameter	Type	Description
cmdNr	Int	The command currently being executed in the program progNr
progNr	Int	The nr of the program currently being executed. The nr refers to the programs loaded in the robot controller, not to the files in the folder /Data/Programs. For single programs this parameter is 0. When there are subprograms this parameter can be 1, 2, …, depending on the nr of subprograms.
reason	String	Reason for stopping the execution: PLAN Program finished correctly as programmed USER The user stopped the program execution PLC The PLC interface stopped the program exec. ERROR An external error stopped the execution, e.g. emergency stop.
errordescription	String	Description of the error, e.g. "JointLimits Min exceeded…" Only program execution errors are listed here, not external errors that cause a fault in the complete robot system (emergency stop, overtemperature, …) These are available interpreting the error code and the EXECEND reason.

Remarks:

- When the robot executes the program in "repeat" mode the EXECEND message is not sent.
- When pausing and restarting a program there will be two EXECACK messages for the current command: one when the command is getting active in the program flow before pausing; the second when the command is reactivated after resuming.

- The EXECERROR messages is send e.g. when joint min/max values are exceeded, or in case of singularities.
- If the robot does not start a program, e.g. due to an error or due to missing referencing, there will be no EXECACK message.

4.10 Configuration Commands

Available in CPRog: from version V902-11
Available in TinyCtrl: from version V980-11

The configuration commands are sent to the robot to query or change configuration parameters.

CRISTART 1234 CONFIG SetDIOModules canID1 canID2 canID3 CRIEND Enables or disables DIO modules and sets their CAN IDs. The parameters canId1 to canID3 can be 0 to disable a module or a CAN ID value up to 255. If one of the canIDs is 0 all following IDs must be 0. TinyCtrl will write this change to its project configuration file.
CRISTART 1234 CONFIG GetDIOModules CRIEND Requests the CAN IDs of the DIO modules. The response contains the CAN IDs as described in SetDIOModules: CRISTART 1234 CONFIG DIOModules canID1 canID2 canID3 CRIEND
CRISTART 1234 CONFIG SetDOutDefaults ResetStates0-31 ResetStates32-63 ErrorStates0-31 ErrorStates32-63 CRIEND Sets the DOut states that are set on reset and error. The parameters are 64 bit unsigned integers. Each two bit represent one DOut port: 00 false 01 true 10 no change 11 reserved
CRISTART 1234 CONFIG GetDOutDefaults CRIEND Requests the DOut default values as described in SetDOutDefaults. **CRISTART 1234 CONFIG DOutDefaults ResetStates0-31 ResetStates32-63 ErrorStates0-31 ErrorStates32-63 CRIEND**
Implemented in CPRog only, available for gantry robots only **CRISTART 1234 CONFIG SetGantryLength x y z CRIEND** Sets the axis length of a portal robot for visualization purposes. Use SetKinematicLimits to change the kinematic axis lengths. The parameters are floating point values. TinyCtrl will write this change to its robot configuration file.
Implemented in CPRog only, available for gantry robots only **CRISTART 1234 CONFIG GetGantryLength CRIEND** Requests the axis length of a gantry robot as described in SetGantryLength. Response: CRISTART 1234 CONFIG GantryLength x y z CRIEND
Implemented for gantry robots only **CRISTART 1234 CONFIG SetKinematicLimits minValue maxValue CRIEND** Sets the kinematic axis lengths. The number of parameters depends on the robot type, the upper limit is 9 pairs of minimum and maximum values. The parameters are floating point values. TinyCtrl will write this change to its robot configuration file.
Implemented for gantry robots only **CRISTART 1234 CONFIG GetKinematicLimits CRIEND** Requests the kinematic axis lengths. The response contains up to 9 value pairs as described in SetKinematicLimits: CRISTART 1234 CONFIG KinematicLimits minValue maxValue CRIEND
CRISTART 1234 CONFIG SetPLCInterface inEnable inRequestReference inPlay outNoFault outProgramRunning outRobotIsReferenced inPause CRIEND Sets the PLC interface numbers. inEnable, inRequestReference, inPlay, outNoFault, outProgramRunning, outRobotIsReferenced, inPause are integer values. inPause is optional. TinyCtrl will write this change to its project configuration file.
CRISTART 1234 CONFIG GetPLCInterface CRIEND Requests the PLC interface numbers. The response contains these values as

described in SetPLCInterface. inPause will only be sent if it is supported. **CRISTART 1234 CONFIG PLCInterface inEnable inRequestReference inPlay outNoFault outProgramRunning outRobotIsReferenced inPause CRIEND**
CRISTART 1234 CONFIG SetPLCInterfaceEnabled active autoConnect CRIEND Enables or disables the PLC interface and the auto connect function. active and autoConnect must be 'True' or 'False'. TinyCtrl will write this change to its project configuration file.
CRISTART 1234 CONFIG GetPLCInterfaceEnabled CRIEND Requests the PLC enabled and autoConnect states. The response contains these values as described in SetPLCInterfaceEnabled: **CRISTART 1234 CONFIG PLCInterfaceEnabled active autoConnect CRIEND**
CRISTART 1234 CONFIG SetBrake DOut CRIEND Sets the brake DOut number, -1 to disable the brake.
CRISTART 1234 CONFIG GetBrake CRIEND Requests the brake DOut number, -1 is returned if the brake is disabled: **CRISTART 1234 CONFIG Brake Dout CRIEND**
CRISTART 1234 CONFIG SetProgramDefaultState defaultState CRIEND defaultState is an integer defining the default state after an error or reset. defaultState = 0 Paused defaultState = 1 Stopped
CRISTART 1234 CONFIG GetProgramDefaultState CRIEND Requests the program default state as defined in SetProgramDefaultState. **CRISTART 1234 CONFIG ProgramDefaultState defaultState CRIEND**
CRISTART 1234 CONFIG SetCamera type name parameters CRIEND Sets camera parameters. The parameters depend on the type: Type "None": No parameters, removes the camera if it exists. Type "IFMO2D" (IFM O2D camera): active IP port scaleX scaleY originX originY originZ lookX lookY lookZ upY upY upZ zDistance imageEnabled Type "SolutionBinPicking": active IP port originX originY originZ lookX lookY lookZ upY upY upZ zDistance imageEnabled active and imageEnabled must be 'True' or 'False', IP must be IPv4 address string, port must be integer, all further parameters are floating point numbers. More types might be added in future.
CRISTART 1234 CONFIG GetCameras CRIEND Requests all camera configurations. Each camera configuration will be sent as a separate response in the following format: CRISTART 1234 Camera count type name parameters CRIEND Count is the total number of cameras. Type, name and parameters are as described in SetCamera.
CRISTART 1234 CONFIG ClearCameras CRIEND Removes all cameras.
CRISTART 1234 CAMINFO CameraResult type name status parameters CRIEND Camera results sent by the CRI server. Type and name as described in SetCamera. Status values: INACTIVE, NOTCONNECTED, CONNECTED, ERROR Parameters depend on camera type: Type "IFMO2D": posX posY posZ oriA oriB oriC modelClass Type "SolutionBinPicking": posX posY posZ oriA oriB oriC modelClass More types might be added in future.
CRISTART 1234 CAMINFO CameraImage name imagedata CRIEND Camera images sent by the CRI server. Name as described in SetCamera, imagedata is either "NOIMAGE" or Base64 encoded image data.
CRISTART 1234 CONFIG SetExternalAxes number [parameters] [parameters] [parameters] CRIEND Sets the external axes configuration. Number describes the number of external axes, 0 disables all external axes (only 0-1 supported in CPRog/iRC <= V902-11-023 and TinyCtrl <= V980-11-100). For each external axis a set of parameters follows in the following format: **Type kinematic canid gearscale min max velmax acc accinc [directionAngleToY lz0 dir offset]**

Type is a string stating the axis type. It must not contain a whitespace. "na" if no type is given. Kinematic must be "Dependent" or "Independent" Canid is the CAN-module ID The last four parameters may be missing if only one axis is specified and are only relevant if kinematic is set to "Dependent". This configuration will only be applied after saving the project and reloading it (CPRog/iRC) or restarting the robot control (TinyCtrl)
CRISTART 1234 CONFIG GetExternalAxes CRIEND Requests the external axes configuration. The parameters are as described in SetExternalAxes. **CRISTART 1234 CONFIG ExternalAxes number [parameters] [parameters] [parameters]**
CRISTART 1234 CONFIG SetTool filename [parameters for future use] CRIEND Sets the tool configuration filename or "none" to remove the tool. The file must be present at the robot control. Optional parameters might be added in future. TinyCtrl must re restarted for this command to take effect. The following response will be sent if the file does not exist: **CRISTART 1234 CONFIGERROR Tool invalid_file CRIEND**
CRISTART 1234 CONFIG GetTool CRIEND Requests the tool configuration filename. Optional parameters might be added in future. **CRISTART 1234 CONFIG Tool filename [parameters for future use] CRIEND**
CRISTART 1234 CONFIG SetVBox enabled xmin xmax ymin ymax zmin zmax CRIEND Sets the virtual box configuration. 'enabled' must be True or False, the parameters are float values.
CRISTART 1234 CONFIG GetVBox CRIEND Requests the virtual box configuration as described for SetVBox. **CRISTART 1234 CONFIG VBox enabled xmin xmax ymin ymax zmin zmax CRIEND**

4.11 Licensing Commands

Available in CPRog: from version V902-12
Available in TinyCtrl: from version V980-12

License information can be queried via the following commands. Installing new licenses is done via SSH.

CRISTART 1234 LICENSE GetInfo CRIEND Requests license information. The following response will be sent: **CRISTART 1234 LICENSE Info <valid/invalid> <evaluation period> <date> <owner> CRIEND** Evaluation period is the remaining number of seconds that all extended features can be used without license.
CRISTART 1234 LICENSE GetFeatures CRIEND Requests information features that are enabled by the installed license. The following response will be sent: **CRISTART 1234 LICENSE Features <list of features> CRIEND** <list of features> is a list of strings separated by semicolon. Additional parameters may be added to the entries in future.
CRISTART 1234 LICENSE GetDeviceID CRIEND Requests the device ID that is needed for a license request. The following response will be sent: **CRISTART 1234 LICENSE DeviceID <ID> CRIEND** <ID> is a SHA256 hash encoded as a hex string