



25 MEI 2023

COMMUNICATION WITH THE IGUS ROBOT

FONTYS INDUSTRY PROJECT

AARTS,LUUK L.H.K.

FONTYS FHICT

The Igu Robot arm




Table of contents

Introduction	2
Questions	2
<i>Main question.....</i>	<i>2</i>
<i>Sub questions.....</i>	<i>2</i>
What	2
Why.....	2
How.....	2
How to connect with the Igu robot?	3
What is the message format for the Igu robot?.....	3
<i>What is the Igu robot client send message format?</i>	<i>3</i>
<i>What is the Igu robot server return message format?.....</i>	<i>3</i>
What is the Igu robot communication sequence?	4
<i>Found Client commands</i>	<i>4</i>
How to make the Igu robot arm move?	5
<i>Joint locational movement.....</i>	<i>5</i>
<i>Joint coordinal movement</i>	<i>5</i>
<i>Testing.....</i>	<i>5</i>
Conclusion.....	6
Sources.....	7

Introduction

This document will explain the Igus robot communication protocol and all the research behind this document. After reading this document you will understand how to connect with, control and get data from the Igus robot arm.

Questions

In this document I will answer a few questions that will work together to answer to the question: How does the Igus robot arm Ethernet communication protocol work?

Main question

- How does the Igus robot arm Ethernet communication protocol work?

Sub questions

- How to connect with the Igus robot?
- What is the message format for the Igus robot?
 - o What is the Igus robot sending message format?
 - o What is the Igus robot return message format?
- What is the Igus robot communication sequence?
- How to make the Igus robot arm move?

What

In this document I will show my research procedure, problems I encountered and the solutions/answers to the questions. I will implement the [DOT framework](#) as a guideline on how to display my research and testing in a clear and easy to understand document.

Why

Our challenge for this project is to make the robot arm move a package from a digital conveyor belt to a drop location. One of the most important tasks during the project will be to move the arm from point a to point b. The questions I asked above will pave the pad to creating an application that controls the robot. Doing research and testing the found information will make it easier for us and potential software users, to work with, create and adjust our final product.

How

All questions I will be answering have already been handled by Igus or some communities online. This makes my research mostly based on searching, reading and testing the solutions in existing documentation and discussions online. Since I will be using the DOT framework as a guideline, the Library method will be implemented while reading and searching through documentation and online solutions for information. I will implement the Field and Lab methods when testing the information on the robot whit small demos and by adjusting the final product.

How to connect with the Igus robot?

The Igus robot arm has a contact protocol that it follows while using the ethernet port. In this case the IP address of the robot is 192.168.3.11 and the port that should be used when reaching out is port 3920. When reaching out to the robot arm with a Connect command, it will send a return message, this indicates that a connection is established. In this case the connection should be kept alive by use of alive messages. The Igus robot expects at least one alive message every 2 seconds. This can be more, but not less, if this message is not sent, the connection will be closed and can be reestablished after 1 second.

Source¹

What is the message format for the Igus robot?

The Igus robot receives and sends messages as a string. Every message, send and received, has a Start word and End word so both the client and server can clearly see find the beginning and ending of the message. As stated in the 'CPR_RobotInterfaceCRI.pdf'¹, the Start message is "CRISTART" and the End message is "CRIEND".

What is the Igus robot client send message format?

The base of every message exists out of 5 parts send in the form of a String. Each capital written word won't change each regular word will change depending on the goal of the message and timing.

The message looks like: "CRISTART counter CMD_CATEGORY command_details CRIEND".

- CRISTART will not change, starts the message.
- The counter is set by middle of a value that goes from 1 to 9999 and then set back to 1. This value will grow by 1 with each message send.
- CMD_CATEGORY will be changed into the command it is sending, like "ALIVEJOG".
- Command_details will form the data that is send with this command. In case of the alive message, this will be the jog values of all 6 joints and 3 gripper joints, these jog values will be written as floats with a range of -100.0 – 100.0 and will look like: "10.0 20.0 30.0 40.0 50.0 60.0 -10.0 -20.0 -30.0".
- CRIEND does not change either, and ends the message.

What is the Igus robot server return message format?

The return message of the Robot arm (server) will be sent each time it receives a message from the PLC (client). Just as with the message from the client, the message starts with "CRISTART count" and ends with "CRIEND". Count is the server count, which is not in contact with the client count. The message will contain all data like, error status and joint positions, sent in one long string. The result is a message that will look somewhat like this:

```
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
```

But then printed behind each other in a string.

The exact meaning of every value can be found on page 5 and 6 of the CPR_RobotInterfaceCRI.pdf file that is linked on the Sources page of this document.

It became quite useful to understand the return messages while testing. We had some trouble when the motors would not move after sending a move command, the Error MNE, Motors Not Enabled, told us we had to enable the motors before they could move.

What is the Igu robot communication sequence?

By using the example code ² I found out that after the TCP IP connection has been established the robot should still be connected with the client via a message command. Afterwards the robot will acknowledge the connection and the client should start sending Heartbeat messages 'ALIVEJOG'.

I tested these sequences and got the communication working after some debugging. At first, I started sending the alive messages after the TCP connection was made, but since the message counter for the client started counting up before there was a real connection, that did not work well. After starting the alive sequence when connected I could keep the connection alive.

The CPR_RobotInterfaceCRI.pdf documentation gave some commands to send. While working with a demo we made I could test these commands and see how the robot reacted on these commands.

Now when setting up the program we made I will Connect and Enable, which makes the program ready to control the robot. We sometimes lose some messages, so I made buttons for these actions to be able to repeat them when there is no reaction.

Found Client commands

As found earlier, the message sent by the client contains 5 parts combined into 1 String. The CMD_CATEGORY should be changed into CMD when sending commands and the command_details will contain the command and details of the command. The available commands are:

- "Connect" establishes the contact between the client and the server.
- "Disconnect" disconnects the client and the server.
- "Reset" Resets the connection between client and server.
- "Enable" enables the joints to move.
- "Disable" disables the joints to move.
- "SetJointsToZero" returns all joints to their zero (rest) position.

When sending commands, the Robot arm (server) will return an acknowledgement or an error message, like: "CRISTART sCnt CMDACK ref_to_cCnt CRIEND" in this case the sCnt is server count and cCnt is client count. The ref_to_cCnt will be the reference to the cCnt of the command message that is acknowledged. The error message, like: "CRISTART sCnt CMDERROR ref_to_cCnt error_description CRIEND".

How to make the Igus robot arm move?

In the documentation provided by Igus, I found the VAR function that would make it possible to change a variable in the robot arm which would then move the arm according to those changed variables. This would mean that I changed the variables for the joint positions to the positions I wanted them at and the robot should move the joints to those positions.

The variable changing messages were explained in the Igus documentation ¹. Some of the variables I needed to set to move the robot, I could find in the CPR_CommandReference.pdf³. Here are some of the solutions it should potentially deliver.

Joint locational movement

To let the Robot arm move, the variables that represent the target position should be changed. This can be done with the CMD_CATEGORY, "VAR" and the CMD_DETAILS "Joint" combined with the values of the target location of the joints. Ending up looking like:
'CRISTART 1 VAR Joint 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 CRIEND'

Joint coordinial movement

The second option is to move on the arms tool location, which will make the grabbing of an object much easier. To control these variables in the robot we need to set the 'VAR' combined with 'Linear' and the target values which would look like:
'CRISTART 1 VAR Linear 10 10 10 CRIEND'

Testing

While I could find multiple sources that implemented these variables as movement function. And even though the C# demo code ² that I searched extensively to even find the function, used variables to move the robot. I could not get this to work, every time I tried, the robot arm gave me the Error that there was no variable named Joint or Linear.

While debugging Dimitar and I, did get the robot to move, by accident. It turns out, what no documentation I have found stated, the alive message also moves the robot at a speed between -100 to 100. So, when set to 0 it keeps the robot steady but when a joint is set to more or less it will move either left or right (left: >0, right: <0).

Using this information, I have decided to move the robot using the alive message and a timer. I measured the time it takes the robot to move a given angle and created a function that could calculate the time and speed necessary.

For example, it took joint A 41 seconds to move 90 degrees at speed 20. My function will change calculate that it takes the robot then 0,455 seconds to move the robot 1 degree.

By testing it on a higher speed I found out that the speed goes up Linear, so when moving at speed 40, the joint will move 90 degrees in 82 seconds, which means 0,228 seconds per 1 degree.

Conclusion

My conclusion on the main question, how does the Igus robot Ethernet communication protocol work? Is that when combining all acquired information about the communication sequence, commands to send and how to move the robot, I know how to control and move the robot arm.

Although I could not find what the problem with the Joint and Linear variables could be, I did find another solution, which in this time frame was the main goal of this research file.

Sources

¹ [CPR_RobotInterfaceCRI.pdf](#), CRI communication protocol explanation, Commonplace Robotics.

² [Example Code](#) CRI interface C#, Commonplace Robotics.

³ [CPR_CommandReference.pdf](#), Igus robot variables, Commonplace Robotics.