**Explicit Messaging Example using EtherNet/IP connection on the YRC1000 to a Rockwell MicroLogix 1400 PLC.**

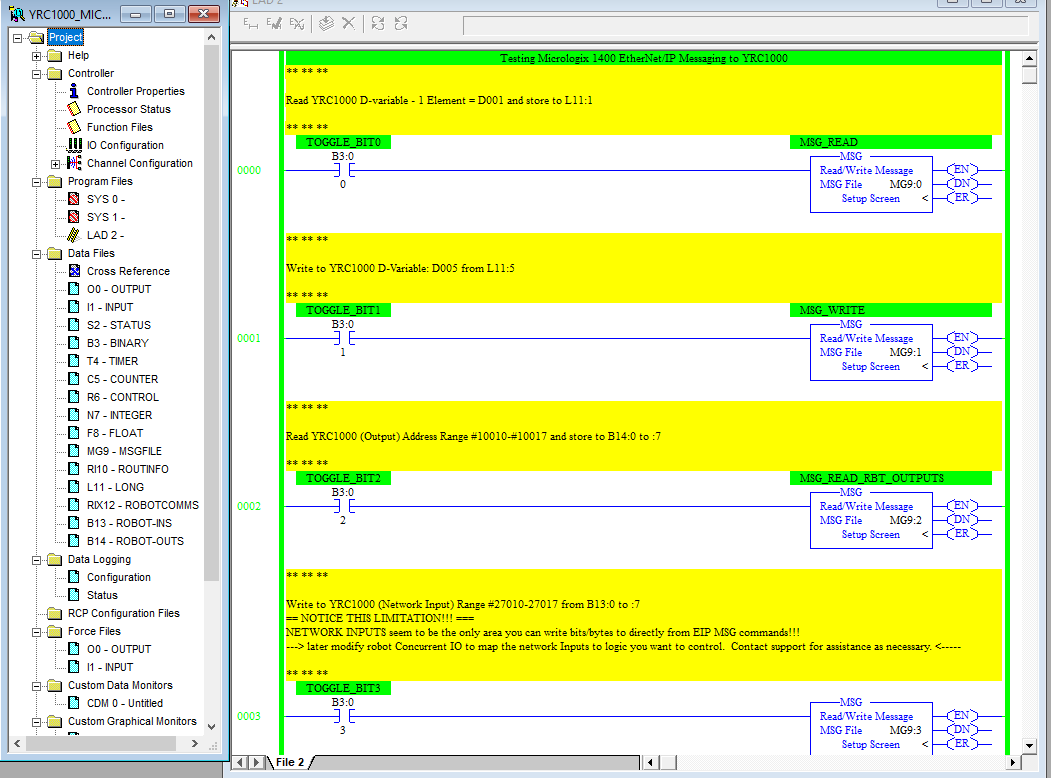
This document builds on the experience in the document which was written for a Compact Logix. If you have not seen that document, please review it first. This document will be quick and dirty, it will include screen shot and hints at how we verified communication between the PLC and robot controller.

**Supporting PLC Project File:**

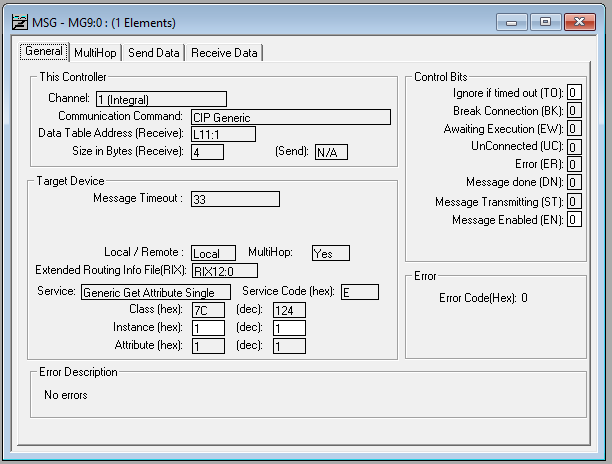
* An example PLC file was also developed. This file contains the exact example code that follows.
  + A version of this project file is embedded in this document:  
    
  + A PDF print of the project file is also embedded:  
    

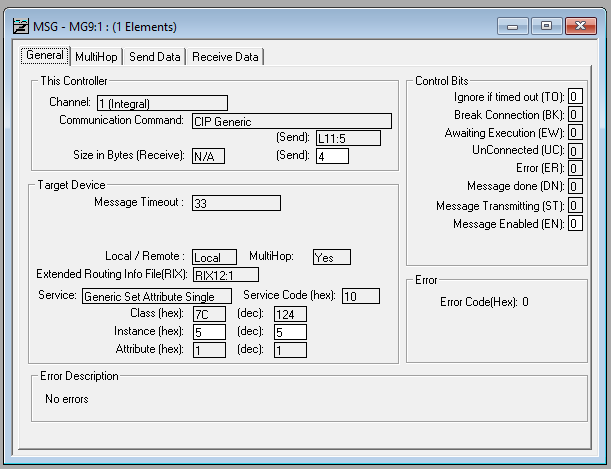
**Tests/Verification:**

1. Set up logic to allow exchange of 32 bit data (D-Variable in robot :: Long variable in PLC)

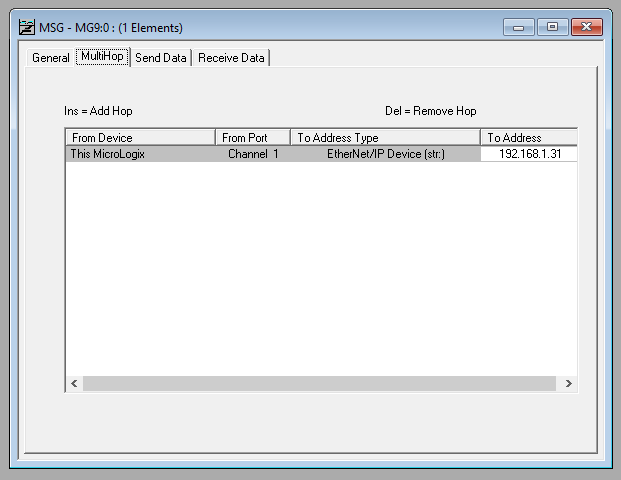


Setup of each Message instruction:



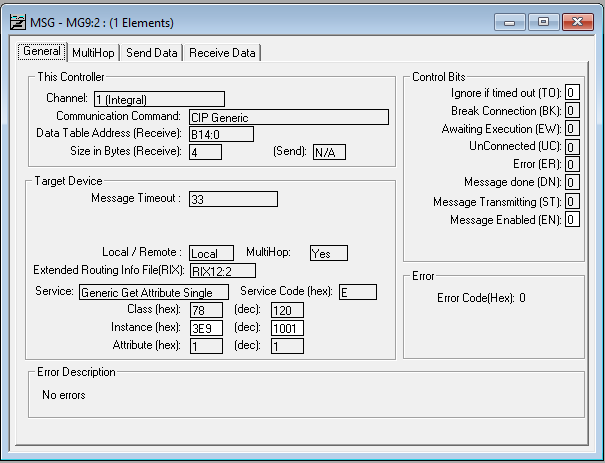


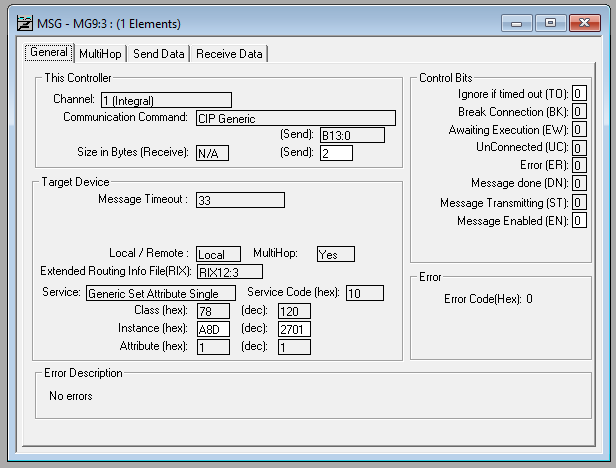
**To see where the Class and Instance numbers came from see Yaskawa manual 178651-1CD, section 5. (EtherNet/IP for YRC1000). Other versions exist for different robot controller types.**

On the MultiHop tab set the robot’s IP address 🡪 192.168.1.31. The same value should be added for every message in the PLC project. Only one screenshot will be shown.

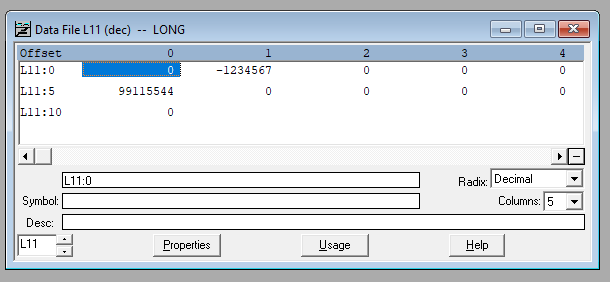
1. Set up logic to allow exchange of I/O data (8 bit groups)
   1. Thoughts – this is inefficient, but a talented programmer can write lots of MSG functions to accomplish what is needed. It may be more cost effective to jump to CompactLogix and implement standard EtherNet/IP IO communication which doesn’t need messaging. I guess the cost different between a low end CompactLogix and MicroLogix 1400 is not enough to warrant the extra control engineering hoops required to jump through this.

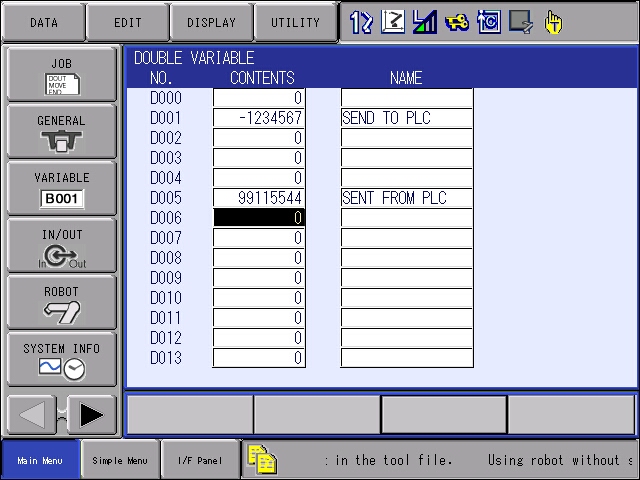






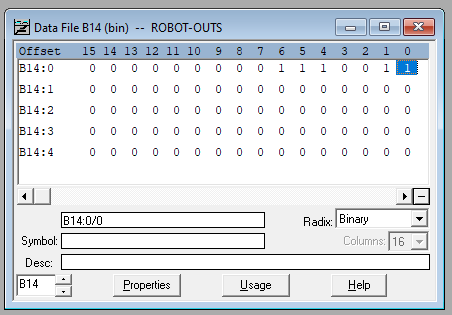
1. Compile, download and test the integration.
   1. First remember only 1 message can be run concurrently. It is up to the programmer to write logic that prevents the messages from running simultaneously. Erroneous data may be returned if 2 messages read or write to the robot controller simultaneously. Timers or blocking bits can be used to accomplish this. The logic in this guide is a starting point, not full implementation.
      1. For the demo program to operate properly only 1 TOGGLE\_BIT should be active at a time.
   2. Toggle bit0 off and on.
   3. Toggle bit1 off and on.
   4. 32 bit data is now exchanged.

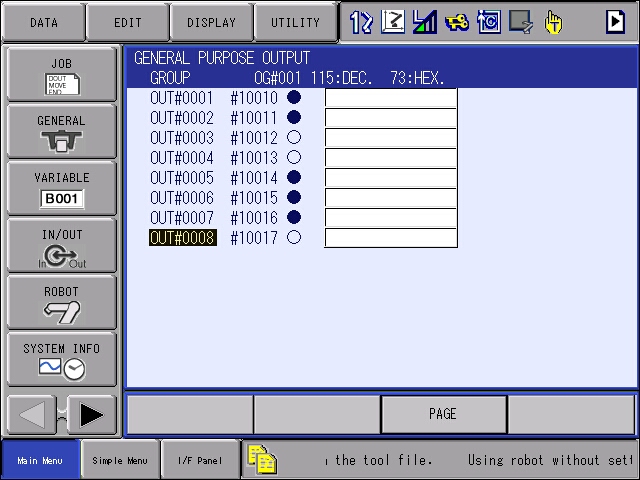




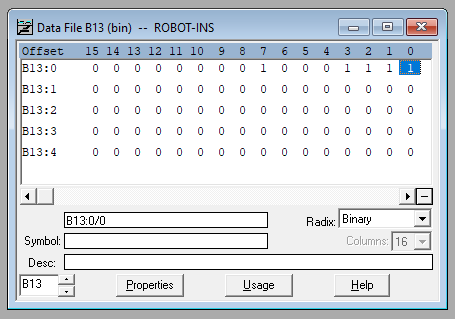
* 1. Toggle bit2 off and on.
  2. Toggle bit3 off and on.
  3. 8 bits of I/O data is now exchanged.

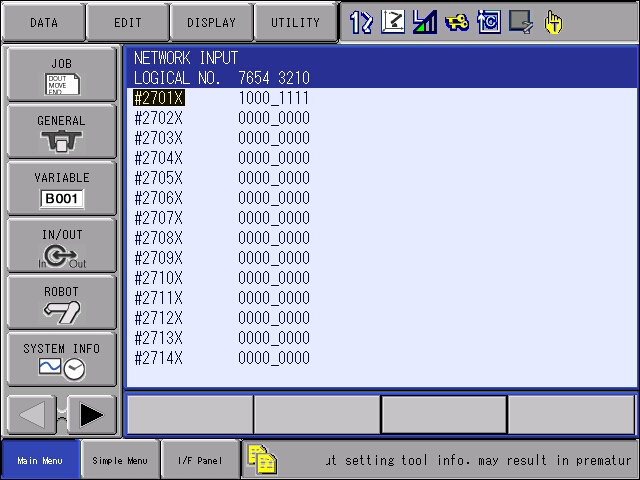
Robot Outputs (#10010 to #10017) to PLC Inputs (B14:0 to B14:7)





PLC Outputs (B13:0 to B13:7) to Robot **Network** Inputs (#27010 to #27017)





Other data format exchanges can be accomplished by changing class and instance numbers per the manual mentioned earlier.

**Appendix:**

I am not sure if there is much value in these files, but I took screen caps, so I’m putting them here just in case.  
