**Explicit Messaging Example using EtherNet/IP connection on the DX200 to a Rockwell PLC.**

The original intent of this document was to verify message communication for a DX200. Since its original writing, it has been confirmed that the approach below is valid for on a DX100, DX200, YRC1000, or YRC1000micro.

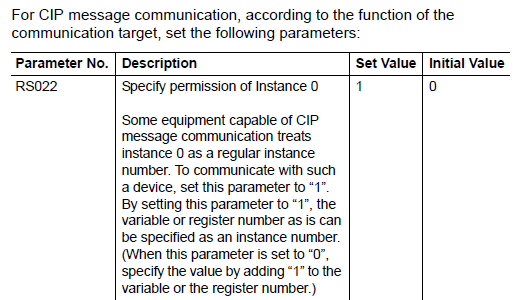
*EtherNet/IP Safe can not be used in combination with the Messaging techniques described in this document. When EtherNet/IP Safe is enabled, Messaging is not available.*

**Reference Part #s for Hardware and Documentation:**

* DX100
  + **156543-1** :: ACCESSORY,ETHERNET IP,PCI,MASTER/SLAVE,DX100
    - PCI Network board
  + **156530-1CD** :: MANUAL,ETHERNET IP,PCI,DX100
* DX200
  + **168183-1** :: ACCESSORY,ETHERNET IP,STANDARD LAN PORT,MASTER/SLAVE,DX200
  + **165838-1CD** :: Manual ~ EtherNet/IP COMMUNICATION FUNCTION (For LAN PORT)
  + **165022-1CD** :: Manual ~ EtherNet/IP Board Robot as Slave to ControlLogix/CompactLogix PLC Configuration Example, DX100/DX200
* YRC1000
  + **179298-1** :: ACCESSORY,ETHERNET/IP,STANDARD LAN PORT,MASTER/SLAVE,YRC1000
  + **178651-1CD** :: Manual ~ ETHERNET/IP COMMUNICATION,STD LAN PORT,YRC1000
  + **181389-1CD** :: Manual ~ ETHERNET/IP CONFIGURATION,ROBOT AS ADAPTER TO CONTROLLOGIX/COMPACTLOGIX PLC,SUPPLEMENT,YRC1000
* YRC1000micro
  + **185230-1** :: ACCESSORY,ETHERNET/IP,STANDARD LAN PORT MASTER/SLAVE,YRC1000micro
  + **181260-1CD** :: Manual ~ ETHERNET/IP COMMUNICATION,STD LAN PORT,YRC1000micro
  + **185733-1CD** :: Manual ~ ETHERNET/IP CONFIG ROBOT AS ADAPTER CONTROLLOGIX/COMPACTLOGIC PLC,SUPPLEMENT,YRC1000MICRO

**NOTE: There is a limitation of one message processing in the robot controller at a time. This limitation is not stated in the manual, but is known by Yaskawa. This means the PLC logic must be structured around this limitation. Do not try to send messages in parallel! This will result in ambiguous data stored in the robot controller or returned from the robot controller.**

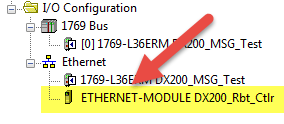
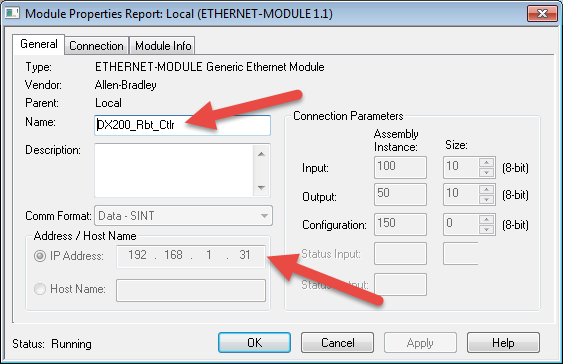
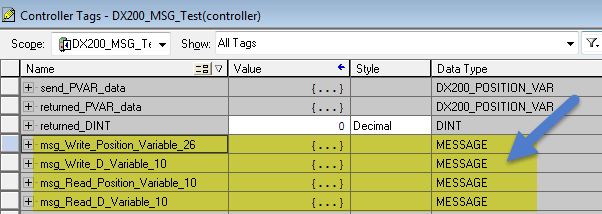
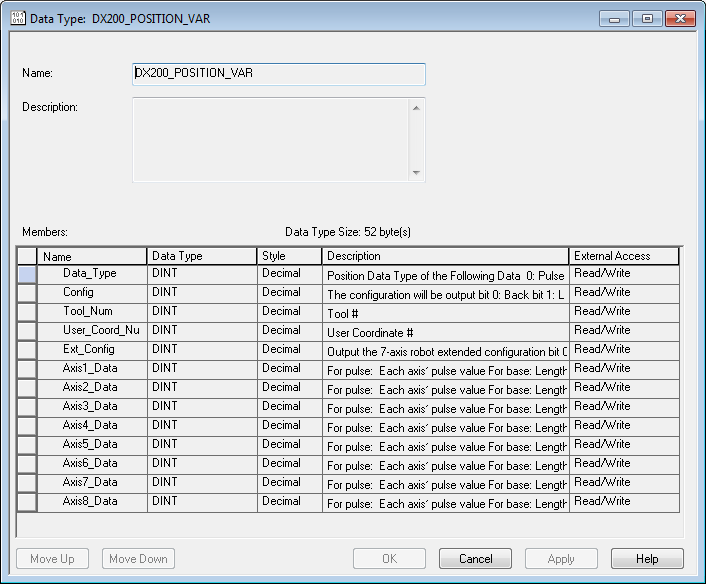
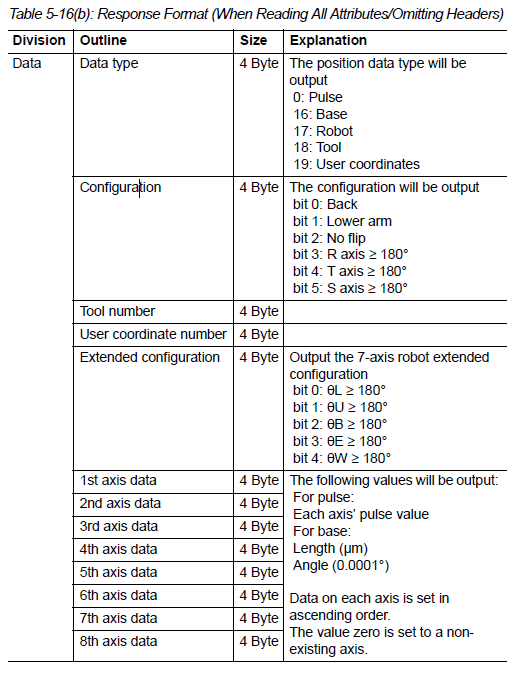
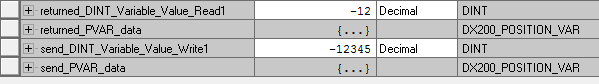
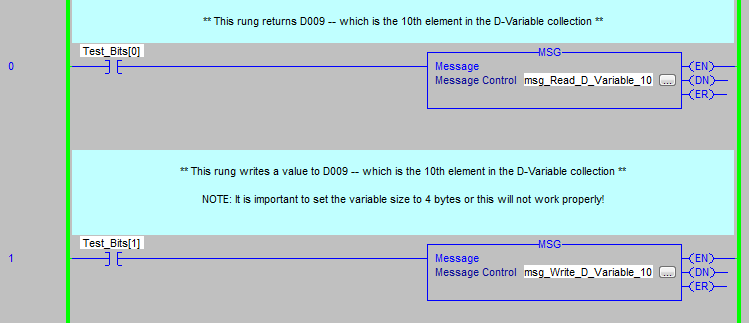
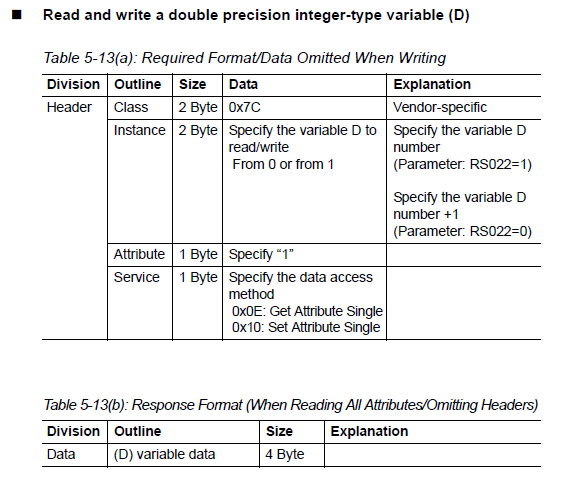
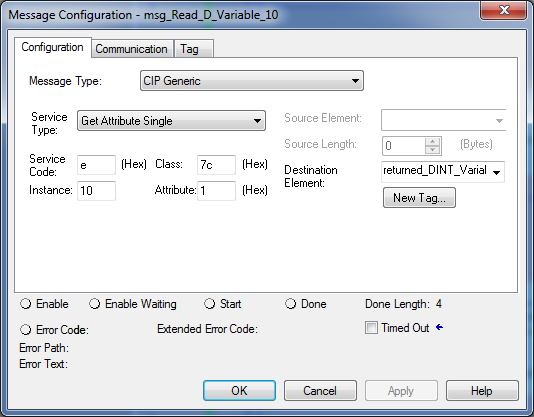
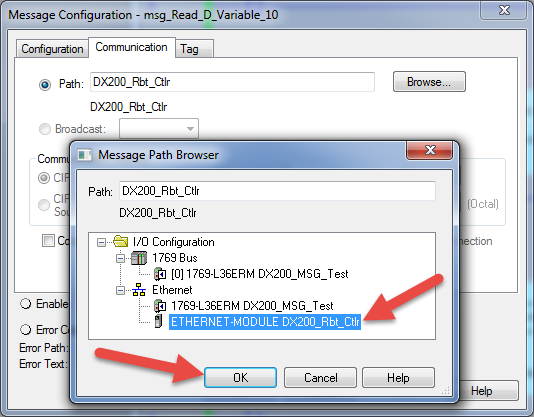
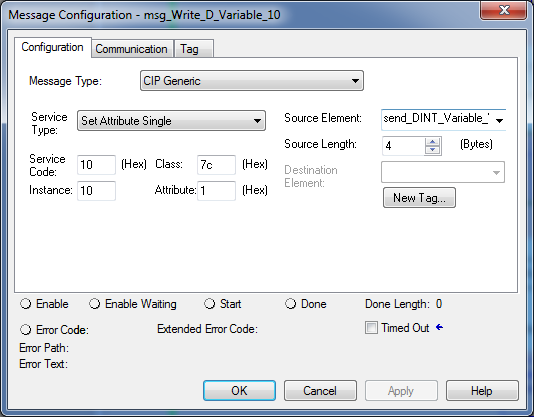
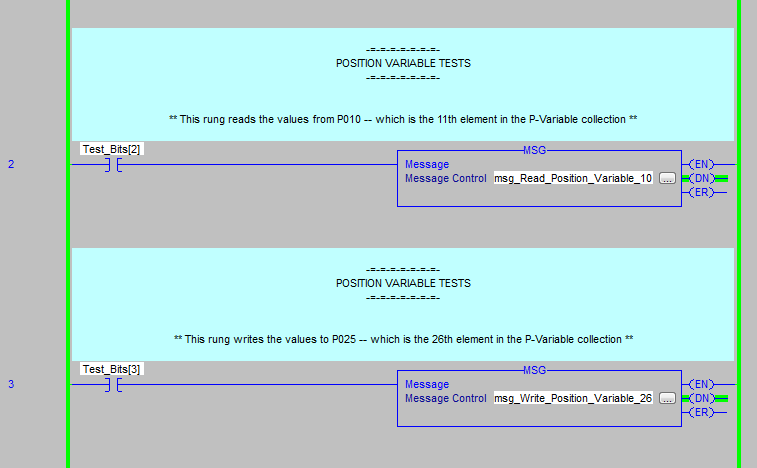
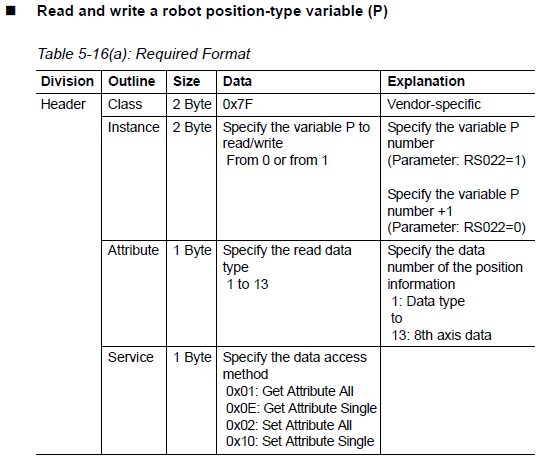
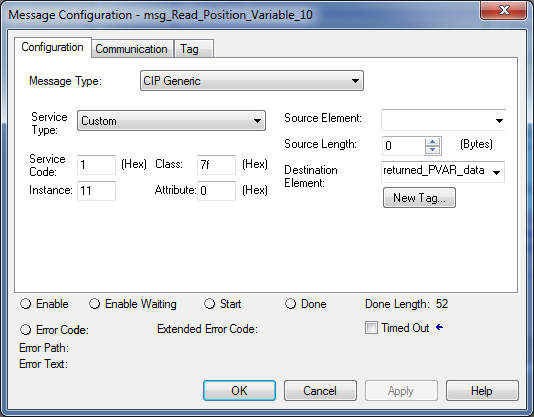
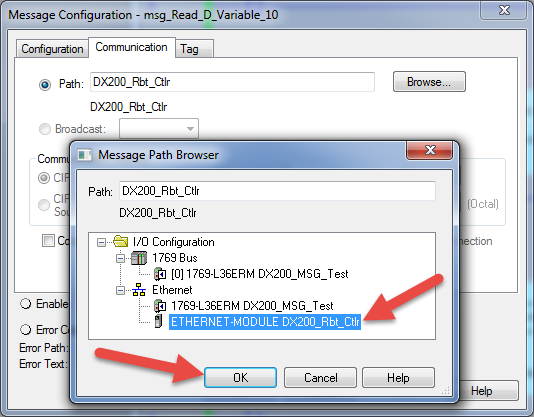
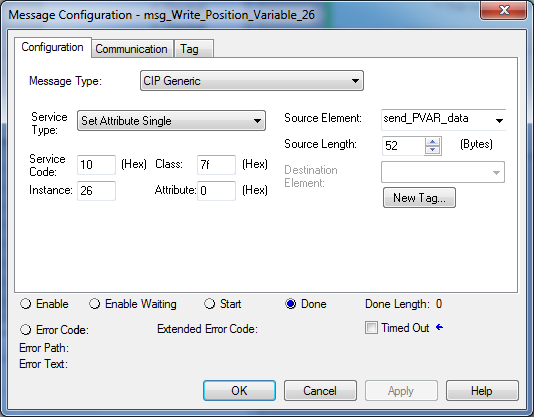
**Parameter Change:**

* The validation of this communication interface was done with RS022 = 0, set in the robot controller. The controller often ships with the parameter set to 1. This means the message commands will offset the variable number by 1. Either the controller parameter can be changed, or the PLC code can be changed by 1, either is easy once this caveat is understood.   
  + The RS022 parameter is described in the EtherNet/IP manual:  
    

**Supporting PLC Project File:**

* An example PLC file was also developed. This file contains the exact example code that follows. It can be provided by request from a Yaskawa representative.
  + DX200\_Explicit\_Message\_Example\_Variable\_Exchange\_PLC\_Ladder.acd

**Steps/Process:**

1. First establish a successful EtherNet/IP connect between the PLC and the DX200 robot controller. (For DX200 reference manual 165022-1CD for assistance.)  
     
   
   1. This is required so there is a valid communication path mapped in the PLC for the MSG commands later.
2. Next define some “MSG” variables. *It appears that the MSG variable scope needs to be “controller” scope and cannot be refined any more local, such as to individual tasks.*  
   
3. If P-Variables are going to be exchanged it would be helpful to create a User Defined data type. This is the version created for this test:   
   
   1. This is actually a series of 13 individual DINTs – it may be quicker to create variables with datatype DINT[13]. The advantage of using the above user defined data type is abstraction – all users can easily understand the purpose of the 13 DINTs in the data to be sent or returned.
   2. The definition for this structure came from the manual:  
      
4. Add some variables to receive or send the data. In this example D Variables (DINT) and P-Variables are exchanged. 4 total were created – 2 of each type for reach and write operations:  
   
5. Add 2 rungs, each with a MSG command and a test bit to trigger the message.   
   
6. Next, configure the data for each MSG.
   1. The first rung will do a **READ** of D-Variable from DX200. First is the section from the manual which defines the values to use.  
      
   2. Now click on the […] in the MSG instruction to define properties for this instruction.  
      
      1. Change **Service Type** to “Get Attribute Single” (GET = READ)
      2. **Service** **Code** will be set to “e” automatically. (This matches the manual.)
      3. **Class** should be set to “7c” (this is for D-Variable)
      4. Set **Instance** to 10 (this means the 10th element, D009)
      5. Set **Attribute** to 1 (this is the default value specified in the manual)
      6. Pick a **Destination Element** – this is a controller variable (DINT) to hold the returned value from the DX200.
   3. Change to the “Communication” tab and set the path to the DX200 Controller  
      
   4. The next rung/MSG will do a **WRITE** of D-Variable from DX200. Refer back to the manual text above if required.
   5. Now click on the […] in the MSG instruction to define properties for this instruction.  
      
      1. Change **Service Type** to “Set Attribute Single” (SET = WRITE)
      2. **Service** **Code** will be set to “10” automatically. (This matches the manual.)
      3. **Class** should be set to “7c” (this is for D-Variable)
      4. Set **Instance** to 10 (this means the 10th element, D009)
      5. Set **Attribute** to 1 (this is the default value specified in the manual)
      6. Pick a **Source Element** – this is a controller variable (DINT) to send a value to the DX200.
         1. The next step is IMPORTANT! You also need to define the data size.
         2. Set **Source Length** = 4 bytes. (DINT = 4 bytes)
   6. Change to the “Communication” tab and set the path to the DX200 Controller as shown above.
7. Save the project. Upload it to the PLC and test it.
8. When everything works as expected it is appropriate to move on to a more complex example, which is working with Position Variables (P-Variables).
9. Now P-Variable read/write will be tested. Add 2 more rungs, each with a MSG command and a test bit to trigger the message  
   
10. Next, configure the data for each MSG.
    1. The first rung will do a **READ** of P-Variable from DX200. First is the section from the manual which defines the values to use.  
       
    2. Now click on the […] in the MSG instruction to define properties for this instruction.  
       
       1. In this case we want to get every value from the P-Variable at the same time, we can not use the “Get Single” service.
          1. Change **Service Type** to “Custom” (this is different from D-Variable)
       2. **Service** **Code** should be set to “1”. (This allows retrievable of ALL attributes.)
       3. **Class** should be set to “7f” (this is for P-Variable)
       4. Set **Instance** to 11 (this means the 10th element, P010)
       5. Set **Attribute** to 0 (value not important if we are getting all elements)
       6. Pick a **Destination Element** – this is a controller variable (User Defined P-Variable data type) to hold the returned value from the DX200.
    3. Change to the “Communication” tab and set the path to the DX200 Controller  
       
    4. The next rung/MSG will do a **WRITE** of P-Variable from DX200. Refer back to the manual text above if required.
    5. Now click on the […] in the MSG instruction to define properties for this instruction.  
       
       1. Change **Service Type** to “Set Attribute Single” (SET = WRITE)
       2. **Service** **Code** will be set to “10” automatically. (This matches the manual.)
       3. **Class** should be set to “7f” (this is for P-Variable)
       4. Set **Instance** to 26 (this means the 25th Position Variable, P025)
       5. Set **Attribute** to 0 (value not important if we are getting all elements)
       6. Pick a **Source Element** – this is a position variable (P-Var) to send a value to the DX200.
          1. The next step is IMPORTANT! You also need to define the data size.
          2. Set **Source Length** = 42 bytes. (13 DINT’s = 52 bytes)
    6. Change to the “Communication” tab and set the path to the DX200 Controller as shown above.
11. Save the project. Upload it to the PLC and test it.