LMI TECHNOLOGIES

**Title** EtherNet/IP communication between Gocator and Allen-Bradley PLC (explicit messaging)

**Revision** 1.6

**Purpose** This application note demonstrates how to interface to a Gocator from an Allen-Bradley PLC using the EtherNet/IP protocol and explicit messaging. The programming tool used to create the examples is RSLogix 5000.

**Equipment** Any Gocator model sensor

## Table of Contents




HEADQUARTERS

+1 604 636 1011

+1 604 516 8368

info@lmi3D.com

www.lmi3D.com

9200 Glenlyon Parkway

Burnaby, BC V5J 5J8

Canada

# 1 Overview

This application note demonstrates how to interface to a Gocator using the EtherNet/IP protocol, which is an industrial protocol that allows bidirectional data transfer with PLCs. It encapsulates the object oriented Common Industrial Protocol (CIP). The Gocator supports EtherNet/IP explicit messaging via TCP, as well as implicit messaging (also known as I/O messaging) via UDP. (For information on implicit messaging, see the app note guide at https://downloads.lmi3d.com/set-implicit-messaging-allen-bradley-plcs.)

The user must create ladder logic to poll the sensor for measurement results. There is no Add-On Profile (AOP) available for the Gocator and it is not possible to use the EDS file for automatic configuration.

EtherNet/IP communication enables the PLC to control a Gocator sensor to:

- Command the sensor to Start, Stop and run a Calibration.
- Switch active configuration.
- Receive sensor state.
- Receive measurement values and decisions.

*Before reading this note it is strongly recommended to read the EtherNet/IP section of the Gocator User's Manual.*

# 2 Configuring the Gocator

The Gocator supports many different output protocols. Before attempting to communicate to the sensor from a PLC, the sensor has to be configured to run the EtherNet/IP protocol. The Gocator sensor is configured through a web browser running on a computer. Figure 1 shows where this configuration takes place in the Output Panel. Refer to the Gocator Quick Start Guide for setup instructions, which can be downloaded here: http://www.lmi3d.com/product/gocator-2000-family/support/files/all

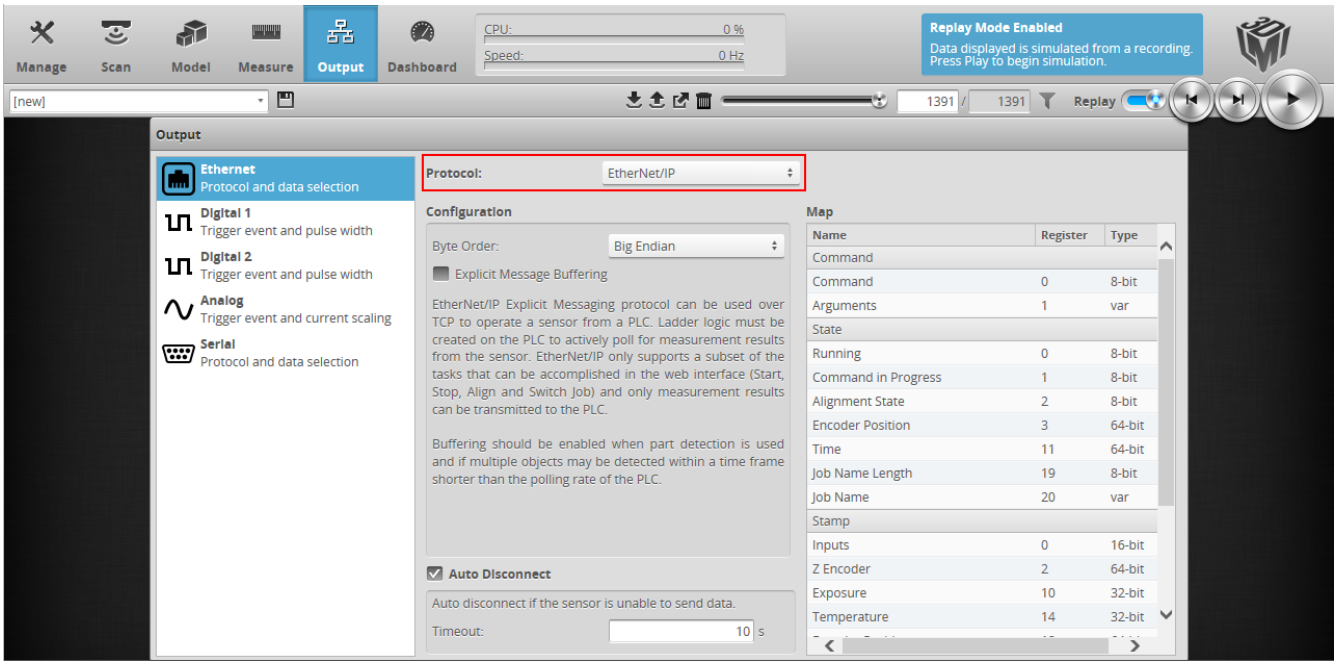



**Figure 1: Configuring the EtherNet/IP protocol in the Output Panel.**

# 3 I/O Configuration in RSLogix

The PLC has to be equipped with an Ethernet card configured to be on the same subnet as the Gocator sensor (factory default Gocator IP address is 192.168.1.10). The Gocator supports unconnected or connected TCP messaging.

The default EtherNet/IP ports are used.

The Ethernet card to which the Gocator is connected should be added as a module to the Backplane. Figure 2 shows an example what this should look like. Verify that the IP Address is on the correct subnet. Note the IP address should be that of the PLC's Ethernet module's, not that of the Gocator's. *SoftLogix 5800 EtherNet/IP was the Ethernet module was added for our particular PLC.*

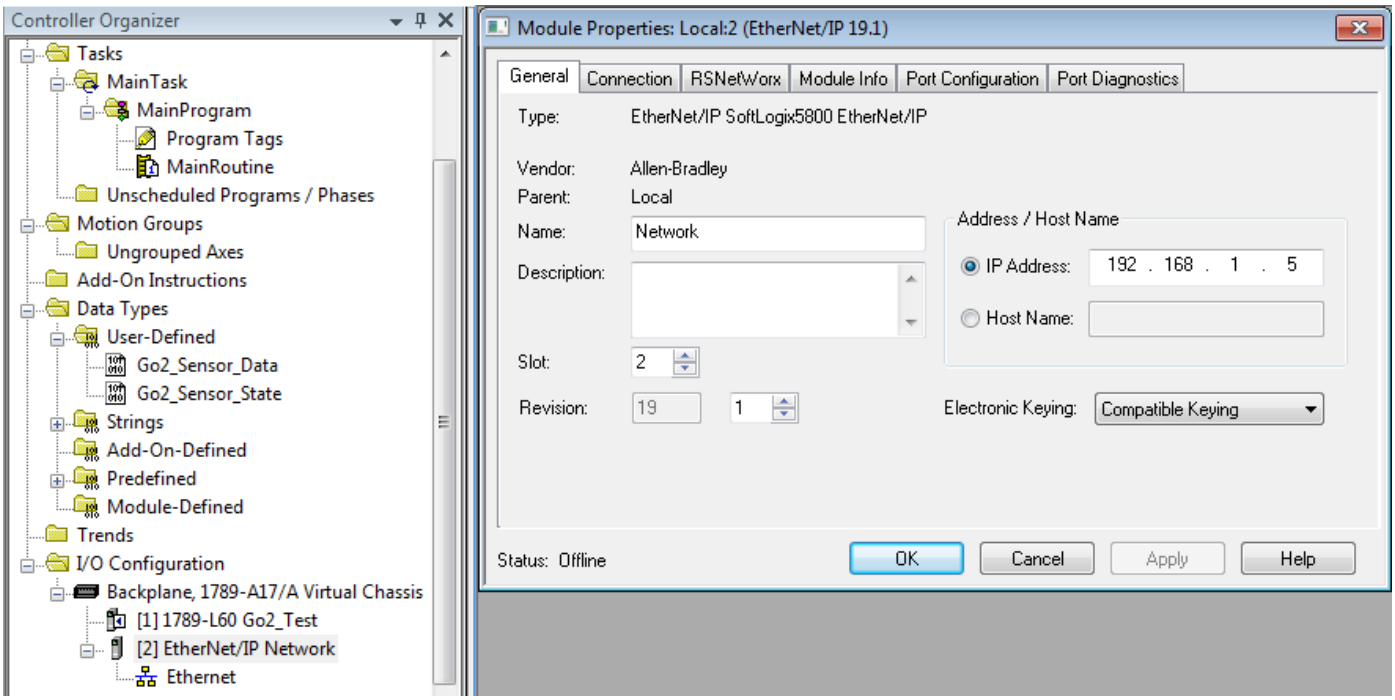



**Figure 2: I/O Configuration in RSLogix.**

# 4 Verifying Connection

Before attempting to control and run the Gocator from the PLC, *the user should always verify the connection first by reading an attribute from the Identity Object*, for example the sensor's serial number.

The EtherNet/IP communication is performed with a MSG block in the ladder logic. The user has to create a Controller Tag for the message block of data type MESSAGE and another tag to hold the serial number returned by the sensor of data type DINT. Figure 3 shows the simplest form of ladder logic with a MSG block to read the serial number from the sensor.

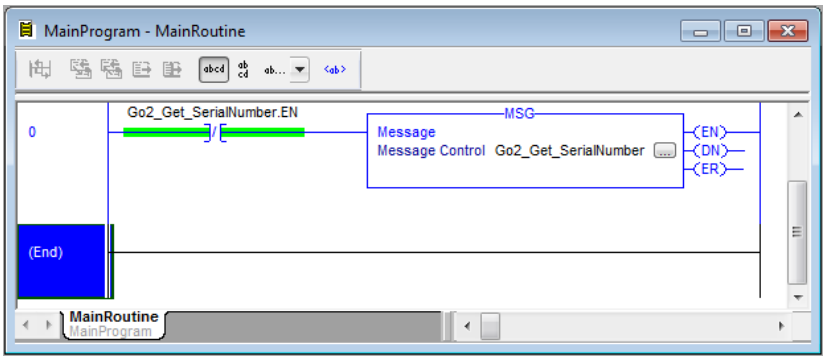



**Figure 3: Ladder logic to read the sensor's serial number.**

Figure 4 shows how to configure the message block. The service code to get a single attribute is 0x0e, the Identity Object is class 0x01 and the Instance is 1. The Gocator User's Manual defines the serial number as Attribute 6. The Destination is set to the tag created to hold the result.

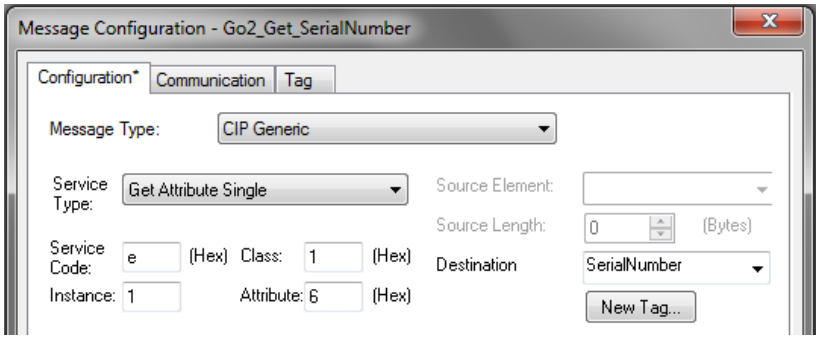



**Figure 4: Message block configuration to read the serial number.**

Finally the communication path has to be set to the IP address of the Gocator, via the Ethernet module in the backplane. Note that the path in Figure 5 is only an example. This path will have to be set according to the particular I/O configuration on site in the factory. *Note especially that the number 2 in this path example is not the slot number, but the Ethernet port number of the particular PLC model.*

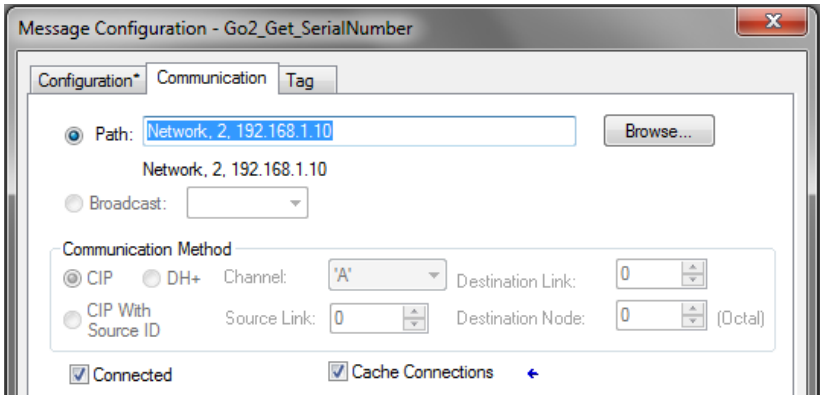



**Figure 5: Message block communication settings.**

When this simple ladder logic is downloaded to the PLC and then executed, the serial number tag should be monitored to verify the success of the communication. This confirms that the communication path is correct and can be used in all other MSG blocks with the Gocator.


**Figure 6: Controller tag receiving the serial number of the sensor.**

## 5 Sending commands to the Gocator

The Command Assembly Object is used to start, stop, calibrate and switch configuration on the sensor. Like with all other communication the commands are sent to the sensor with a MSG block in the ladder logic. *Note that the ladder logic should be designed so that the command is not sent continuously with every scan of the logic*.

The command is written to the sensor as a 32 element byte array. The user should define a Controller Tag of data type SINT[32]. The first byte is the command itself, whereas bytes 1-31 hold the name of the configuration file, which only applies if the command is Load Configuration (5).

Figure 7 shows how to configure the message block to send a command. The service code to set a single attribute is 0x10, the Assembly Object is class 0x04 and the Instance for the Command Assembly is 784 (0x310). The Source Element should be set to the Controller Tag holding to command to be sent to the sensor.

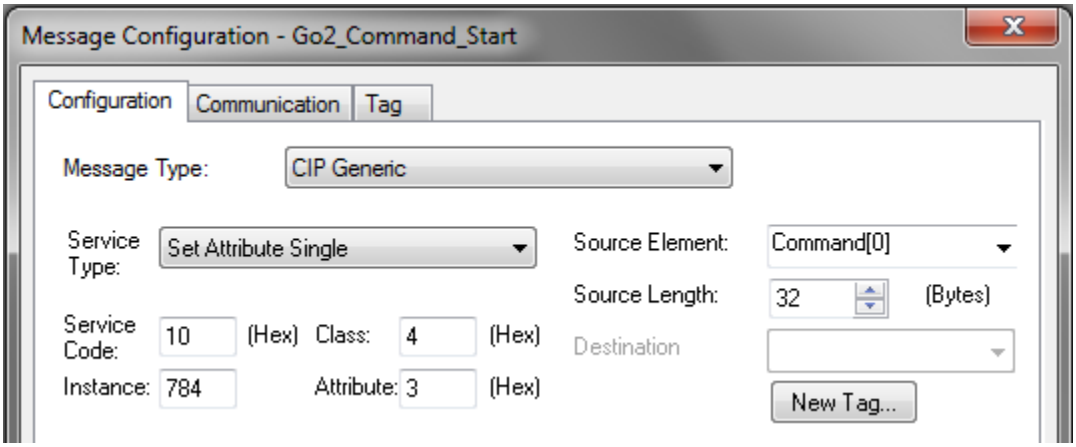



**Figure 7: Message block configuration to write a command to the sensor.**

## 6 Receiving Gocator measurement results

The Sample State Assembly Object is used to receive measurement results from the Gocator. The user should define a Controller Tag to hold the raw data stream from the sensor. The data type of this Controller Tag should be as follows:

- For Gocator firmware version 3.6 or earlier: SINT[180]
- For Gocator firmware version 4.x and later: SINT[380]

*Do not create a user defined data type, since RSLogix will pad the data structure to satisfy internal 4-byte boundaries*.

Figure 8 shows how to configure the message block. The service code to get a single attribute is 0x0e, the Assembly Object is class 0x04 and the Instance for the Sample Assembly is 801 (0x321). The Destination should be set to the Controller Tag to hold the raw data stream.

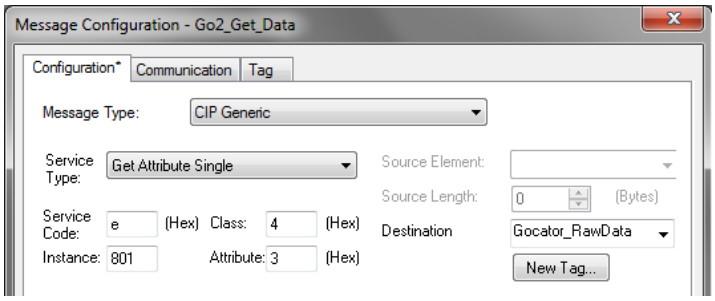
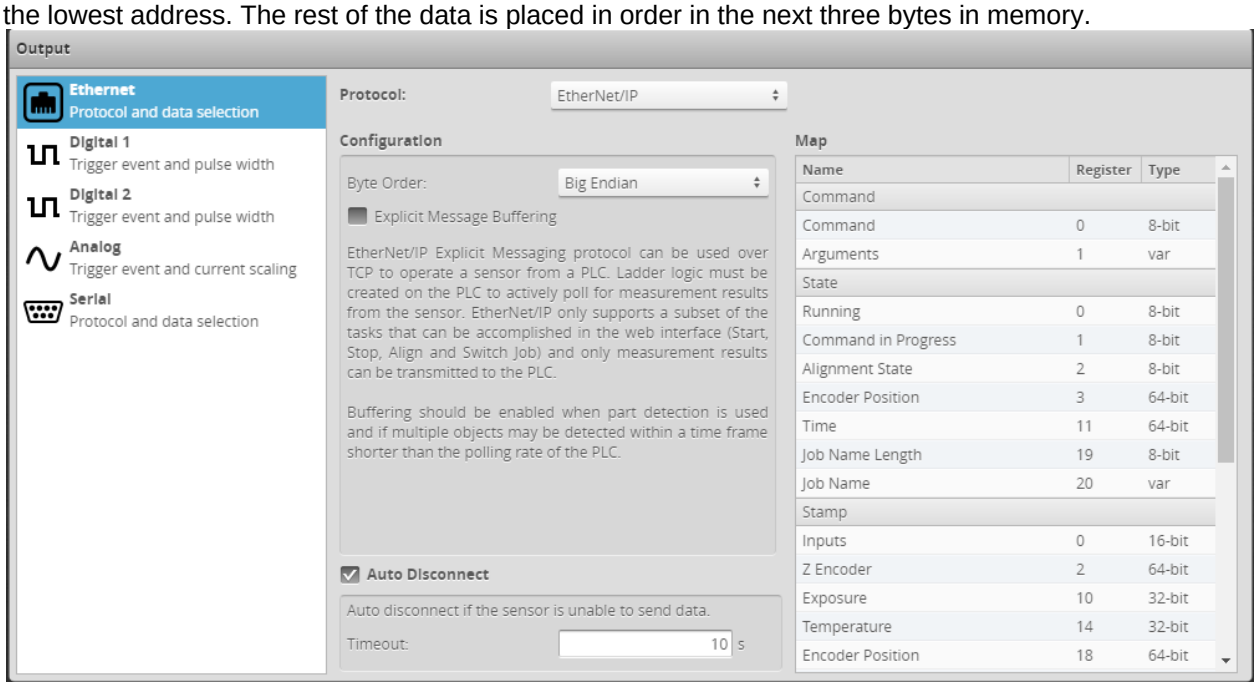



**Figure 8: Message block configuration to read measurement results.**

The user will now have to create ladder logic to copy the correct bits in the raw data stream into Controller Tags holding the individual results. This can be done with the Bit Field Distribute (BTD) block, as illustrated in the example in Figure 9 below. In this example the user has created a Controller Tag called Measurement0_Value of data type DINT. According to the Gocator User's Manual, bytes 80 to 83 hold this specific information.

## 6.1 Byte Order Options

Gocator supports outputting in either Big Endian or Little Endian byte ordering options.
**Big Endian Byte Order:** The most significant byte (the "big end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.
**Little Endian Byte Order:** The least significant byte (the "little end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.



*Note that the raw bytes have to be copied to the correct bits inside the DINT tag*. Below is an example when results are presented in Big Endian format. The lowest byte in the raw data is copied to the highest bits. In the example below, byte 80 is copied to bits 24 to 31; byte 81 is copied to bits 16 to 23 etc. The Gocator is sending measurement values in thousands of a millimetre. In the example below, the measurement result is 51.408 mm.

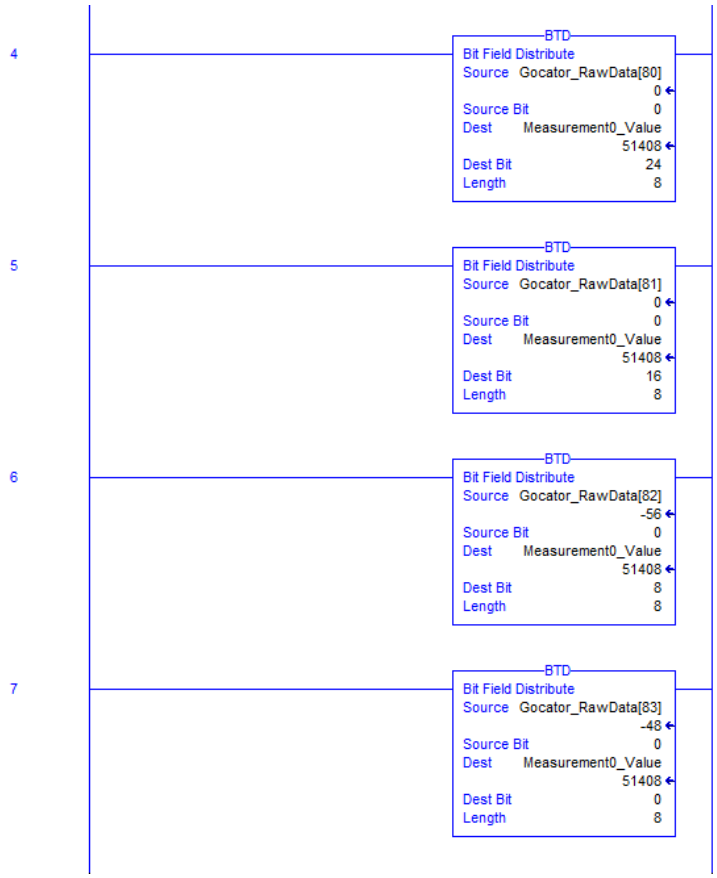



**Figure 9: Ladder logic to convert the Measurement0 value into a DINT tag.**

## 6.2 Managing 64-bit values

The Gocator is using 64-bit values to represent encoder positions, time stamps and frame counts. Most PLC hardware does not support 64-bit data types, so the user can choose to either ignore the most significant bits, or manually handle the values as two DINT tags.

As an example, let's consider the frame count, which according to the Gocator User's Manual is sent in bytes 34 to 41. In order to read the frame count as a 32-bit value, first create a Controller Tag of type DINT called FrameCount. Then use the BTD block to copy byte 38 to 41 to the FrameCount tag, as these are the 4 least significant bytes. Figure 10 shows what the ladder logic should look like for this example.

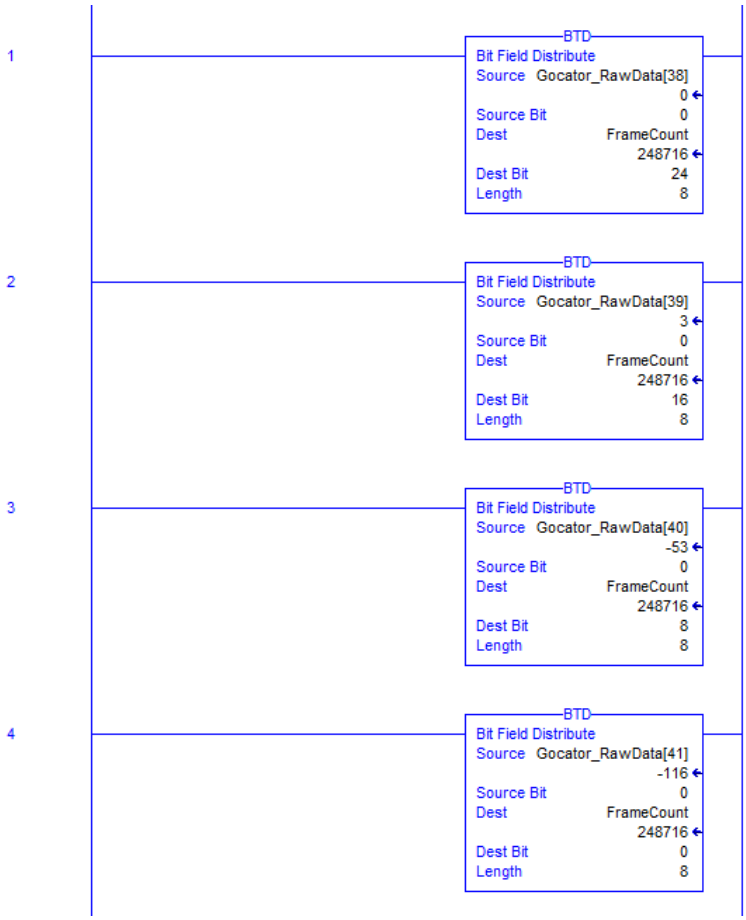



**Figure 10: Ladder logic to convert the 4 least significant bytes of the frame count into a DINT tag.**

# 7 Other Communication

The Gocator supports several more EtherNet/IP object classes in order to read network configuration attributes (such as sensor IP address) and sensor state. This communication is performed in the exact same way as outlined in this document.

The key rule is to not create user defined data types and instead use ladder logic to interpret the raw data stream using the BTD block.