ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

LABORATORIO DE COMUNICACIONES INDUSTRIALES Y SISTEMAS SCADA

Tema de Guía: Comunicación entre PLCs Allen-Bradley mediante Ethernet/IP: Uso de Modelo Productor/Consumidor, Mensajería Implícita y Explícita del Protocolo CIP

Autor:

Arturo Freire Veliz

GUAYAQUIL – ECUADOR

I PAO 2025

Table of Contents

Guía Teórica-Práctica

Tema: Comunicación entre PLCs Allen-Bradley mediante Ethernet/IP: Uso de Modelo Productor/Consumidor, Mensajería Implícita y Explícita del Protocolo CIP

1. Objetivos

1.1. Objetivo general

Implementar un sistema de comunicación entre los controladores lógicos programables (PLCs) MicroLogix, CompactLogix y ControlLogix, mediante el protocolo CIP sobre Ethernet/IP, utilizando etiquetas producidas/consumidas, instrucciones de mensajería implícita y explícita, con el fin de establecer un intercambio eficiente y confiable de datos en entornos industriales automatizados.

1.2. Objetivos específicos

- Analizar los fundamentos teóricos del Protocolo CIP, incluyendo el modelo productor/consumidor, el modelado de objetos, el direccionamiento de nodos en redes Ethernet/IP, y los mecanismos de comunicación mediante mensajería implícita y explícita.
- Configurar correctamente las variables tipo produced/consumed entre los PLCs en Studio 5000 Logix Designer.
- Configurar correctamente las instrucciones de mensajería necesarias para la comunicación entre PLCs, utilizando las instrucciones MSG_SYMBOLIC y MSG_GENERIC en Connected Components Workbench (CCW), así como la instrucción MSG en Studio 5000.

2. Equipos y herramientas

- Connected Components Workbench (CCW).
- Studio 5000 Logix Designer.
- RSLinx Classic o FactoryTalk Linx.
- Cables Ethernet.
- Stratix 5700 / 8300 / 2000.
- PCs.
- Micro850 y Micro870.
- CompactLogix 1769-L33ERM.
- ControlLogix L73 y Módulo 1756-EN2TR/C.
- Manuales de Programación: Guía de inicio de mensajería de cliente CIP, Instrucciones generales de controladores programables Micro800 y Logix 5000, Etiquetas producidas y etiquetas consumidas de los controladores Logix 5000.

3. Marco teórico

3.1. Protocolo Industrial Común (CIP)

Según [1], las redes de ODVA, incluyendo EtherNet/IP™, ControlNet® y DeviceNet® , entre otras, están conectadas mediante uno de los protocolos de comunicación más versátiles de la automatización industrial: el Protocolo Industrial Común (CIP™). CIP abarca un conjunto completo de mensajes y servicios para una amplia gama de aplicaciones de automatización industrial, como control, seguridad, energía, sincronización y movimiento, información y gestión de redes.

El Protocolo CIP define las características de las capas de aplicación para diferentes redes lo que ha dado lugar a llamar la *"Familia de Redes CIP"*. En la Figura 1 se muestra la arquitectura de red, de acuerdo al modelo de referencia OSI, para las distintas redes que implementan CIP en sus capas superiores.

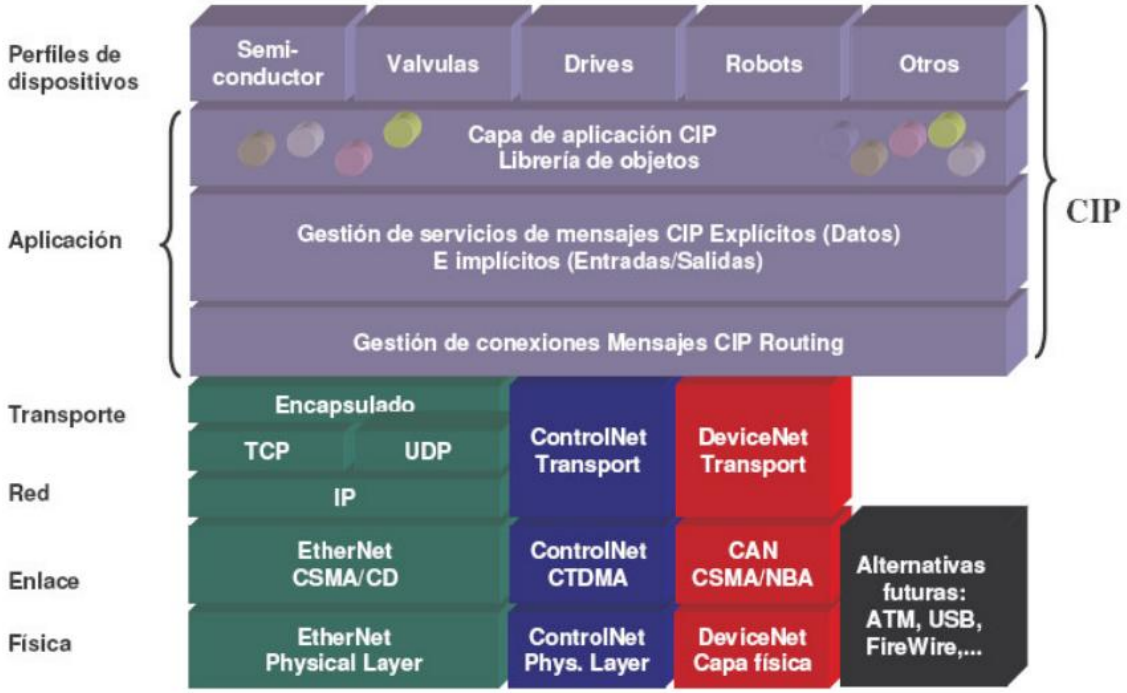


Figura 1. Arquitectura de Red que implementa CIP en sus capas superiores.

La **filosofía** del desarrollo del Protocolo CIP, es la de *proveer de una capa de aplicación que se abstraiga de la tecnología particular de una red industrial*, es decir, de las capas más bajas que definen las características físicas y propias del protocolo de comunicación. De esta forma, se puede aplicar esta capa común a distintas tecnologías de redes, cada una con sus capacidades y aplicaciones particulares.

3.1.1. Red Ethernet/IP

Esta red es una adaptación del estándar IEEE 802.3 para su aplicación como red industrial. En el nombre, la sección "IP" significa "Industrial Protocol", o Protocolo Industrial. Como se observa en la Figura 1, posee *las mismas bases que el EtherNet tradicional* (mismas capas física, de enlace, de red y de transporte). Por tanto, dentro de la familia de redes CIP, esta se utiliza como una red del **nivel de información**, para transportar **grandes volúmenes de información**.

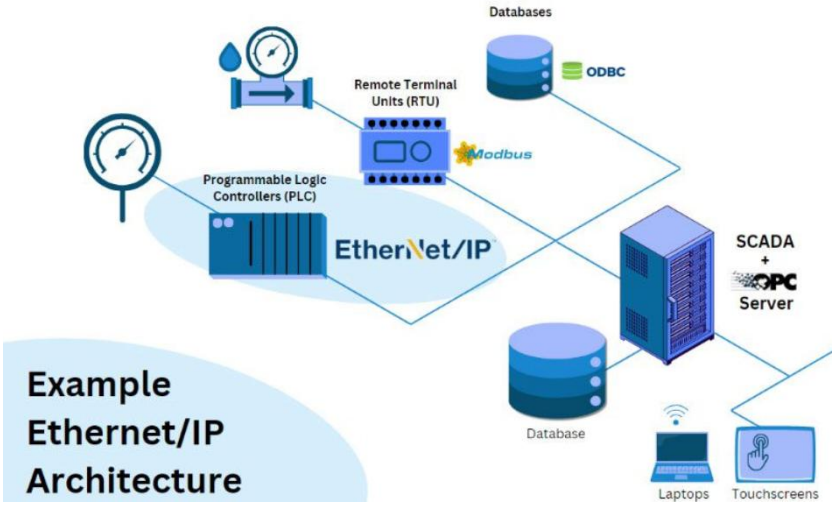


Figura 2. Ejemplo de Arquitectura Ethernet/IP.

3.2. Términos Generales utilizados por CIP

Modelo Fuente/Destino: Es un modelo de comunicación que establece la forma en que son compartidos los mensajes. En este modelo, *el dispositivo fuente envía un mensaje a un solo dispositivo destinatario específico*. Por otra parte, *el dispositivo destino es el que recibe el mensaje procedente desde el dispositivo fuente*. Por esta razón, se incluye en la cabecera de un mensaje, la **dirección de red del dispositivo fuente** y **la dirección de red del dispositivo destino**. Esta tipo de comunicación es *punto a punto*.

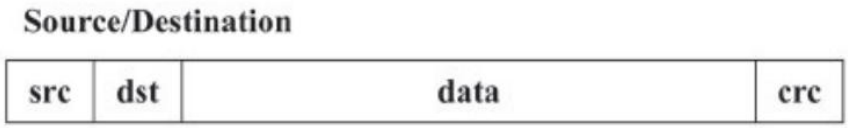


Figura 3. Paquete para Modelo Fuente/Destino.

Modelo Productor/Consumidor: Es un modelo de comunicación que establece la forma en que son compartidos los mensajes. En este modelo, el dispositivo **productor** coloca un *mensaje sobre la red para el consumo* por uno o varios consumidores. Generalmente, el mensaje producido no se dirige a un consumidor específico. Por otra parte, el **consumidor** es un dispositivo *que recoge (consume) un mensaje puesto en la red por un dispositivo productor*. El consumidor determina qué mensaje consumir por medio de un **identificador en la cabecera del mensaje**. Este modelo de comunicación es inherentemente *multicast* y puede soportar además comunicación punto a punto (por lo tanto, puede decirse que incorpora al modelo fuente/desino).

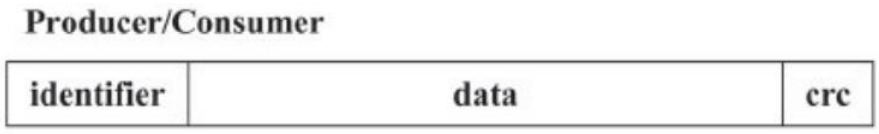


Figura 4. Paquete para Modelo Productor/Consumidor.

Modelo productor/consumidor para CIP: El Protocolo CIP *utiliza el modelo de comunicación productor/consumidor*, en contraste con el modelo tradicional de fuente/destino. Aprovecha su naturaleza *multicast*. Los nodos sobre la red determinan si se deben consumir los datos de un mensaje basándose en un valor de identificación (**identifier**) de la conexión, el cual está incluido en el paquete.

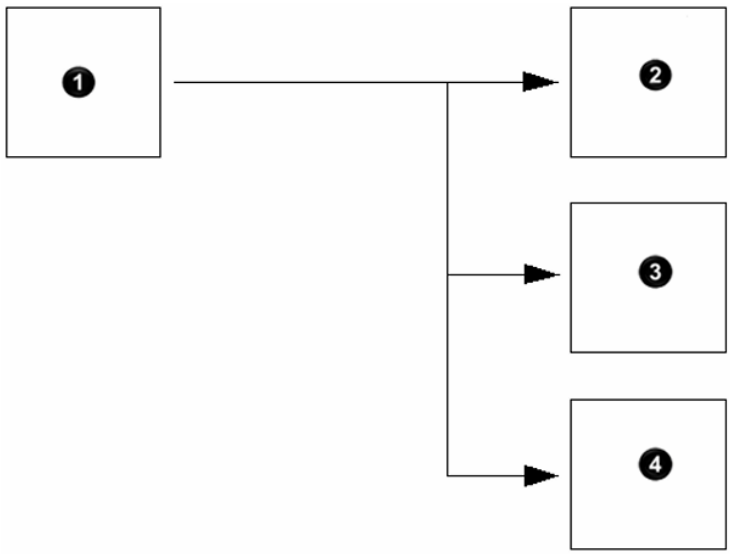


Figura 5. Comunicación entre un Productor y varios Consumidores.

Multicast: Es el envío de información desde un dispositivo emisor (por ejemplo, un cliente que envía un mensaje de solicitud de información) hacia varios dispositivos receptores determinados (por ejemplo, varios dispositivos servidores).

Mensaje explícito: Los mensajes explícitos contienen la *información de dirección* y de *servicio* que lleva al dispositivo receptor a realizar cierto servicio (una **acción**) en una parte específica de un dispositivo (por ejemplo, un **atributo**).

Mensaje I/O o implícito: Los mensajes I/O o implícitos no llevan la información de dirección y/o de servicio; *el nodo consumidor sabe qué hacer con los datos basados en la Conexión ID que fue asignada cuando la conexión fue establecida*. Los mensajes implícitos son nombrados así porque el significado de los datos está implícito por la conexión ID.

Modelo de Objetos: El protocolo CIP utiliza el modelo de objetos para la implementación de las capas de aplicación de las distintas redes de campo. El modelo de objetos se basa en *representaciones abstractas de las características de los dispositivos conectados a determinada red*. Dado que estos objetos se ubican en las capas de aplicación de acuerdo al modelo OSI, un objeto es creado a través del uso de algún lenguaje de programación apropiado (por ejemplo, C++) siguiendo las pautas de estructura y sintaxis entregadas por las especificaciones de CIP.

3.3. Modelado de Objetos

El Protocolo CIP utiliza un modelo de objetos abstractos para describir:

- El conjunto de servicios de comunicación disponibles.
- El *comportamiento visible externamente* de un nodo CIP.
- Un medio común para el *acceso e intercambio de información* entre productos CIP.

Todo *nodo CIP* es modelado como una *colección de objetos*. Un objeto provee una representación abstracta de un componente particular dentro de un producto. Cualquier cosa no descrita en forma de objeto no es visible a través de CIP.

Los objetos CIP están estructurados en los siguientes elementos:

- Clases
- Instancias
- Atributos
- Valores

Una **clase** es un conjunto de objetos donde todos representan el mismo tipo de componente de sistema. Un objeto **instancia** es la representación real de un objeto particular dentro de una clase. A su vez, cada instancia de una clase posee los mismos **atributos**, y que a su vez poseen su propio conjunto particular de **valores**.

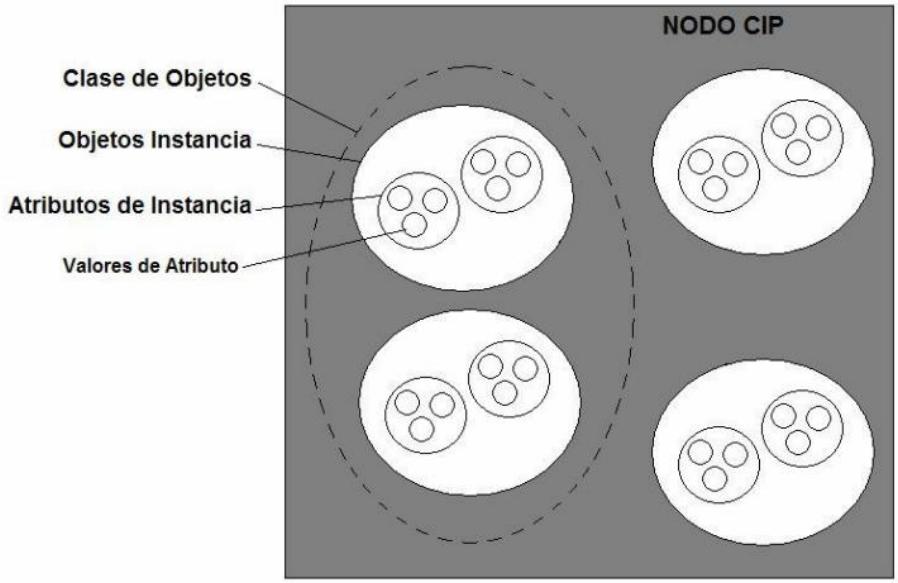


Figura 6. Un Nodo CIP según el modelo de objetos.

3.4. Direccionamiento de Objetos

Los objetos y sus componentes son direccionados a través de un esquema uniforme compuesto de los siguientes elementos:

- **Node Address (o Node ID):** Es un valor entero de identificación asignado a cada nodo sobre una red CIP. En EtherNet/IP, la dirección de nodo es la dirección IP.
- **Class ID:** Es un valor entero de identificación asignado a cada clase de objetos accesible desde la red.
- **Instance ID:** Es un valor entero de identificación asignado a un objeto instancia que lo identifica entre todas las instancias de la misma clase.
- **Attribute ID:** Es un valor entero de identificación asignado a un atributo.
- **Service Code:** Es un valor entero de identificación que denota una solicitud de acción que puede estar dirigido a un objeto instancia particular o objeto clase.

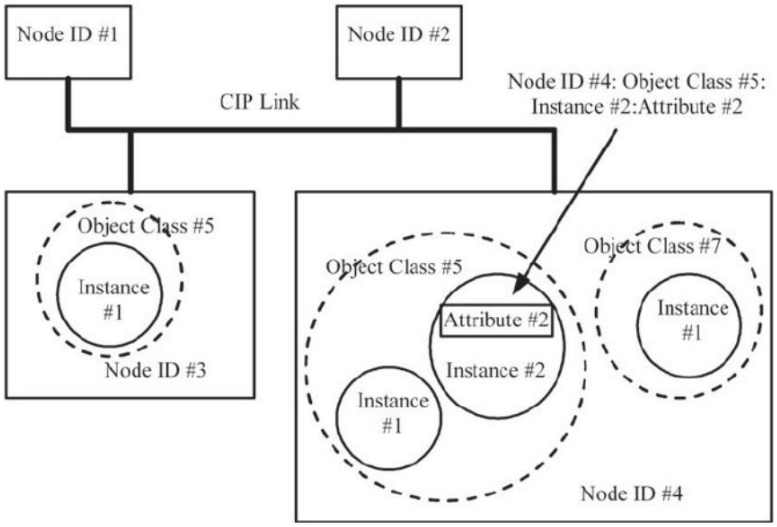


Figura 7. Esquema de direccionamiento CIP.

Se puede observar en la Figura 7 una red CIP (por ejemplo, una red Ethernet/IP). La red posee cuatro nodos. Poniendo atención en el Nodo #4, se puede observar que este posee 3 objetos instancia que se dividen en 2 clases de objetos. Siguiendo el esquema de direccionamiento definido por CIP, se puede observar en la Figura 8 la siguiente secuencia en el Nodo #4:

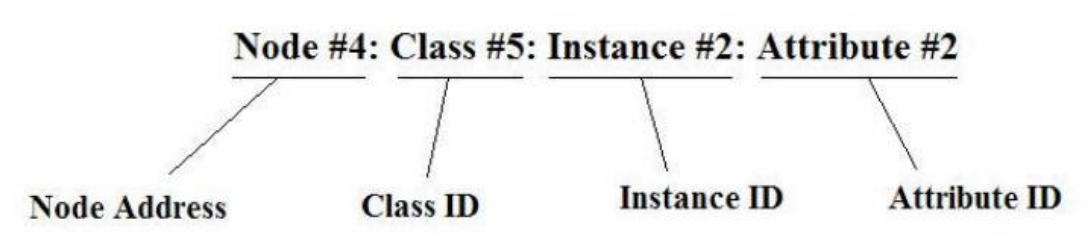


Figura 8. Ejemplo de Direccionamiento.

De esta forma, es *inequívoca la aplicación requerida*. Todos los nodos CIP deben ser direccionados siguiendo este esquema de direccionamiento.

3.5. Servicios

Los códigos de servicio son utilizados *para definir la acción solicitada* cuando un objeto es direccionado a través de mensajería explícita usando el esquema de direccionamiento descrito en la sección anterior. Algunos ejemplos de servicios:

- Get Attribute All
- Get Attribute Single
- Get and Clear
- Set Attribute Single
- Set Attribute List
- Create Object
- Delete Object
- Reset

3.6. Protocolo de Mensajería

El Protocolo Industrial Común (CIP) es basado en conexión. Esto significa que primero *debe ser establecida una conexión entre los nodos antes de comenzar a transmitir*. Una conexión CIP proporciona una trayectoria entre múltiples objetos de aplicación. Cuando se establece una conexión, las transmisiones asociadas a esa conexión se asignan a una **Conexión ID o CID**. Si la conexión implica un intercambio bidireccional, entonces son asignados dos valores CID. En la Figura 9 se representa el esquema de conexión CID.

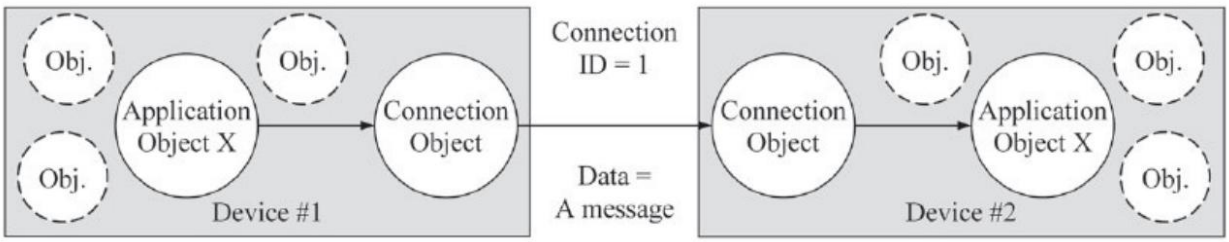


Figura 9. Conexión ID CIP.

3.7. Tipos de Conexiones de Mensajería CIP

Las conexiones CIP se dividen en las siguientes dos categorías:

- **Conexiones de Mensajería I/O o Implícita:** Las Conexiones de Mensajería I/O proporcionan trayectorias de comunicación dedicadas de propósito especial entre una aplicación **productor** y una o más aplicaciones **consumidor**. Los datos I/O específicos de aplicación se mueven a través de estos puertos, proceso llamado frecuentemente de Mensajería Implícita (debido a que se sobreentiende que son datos I/O). Estos mensajes son típicamente multicast.

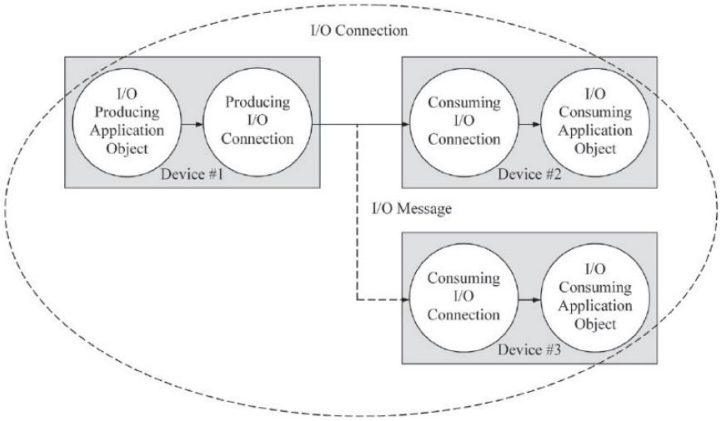


Figura 10. Conexión I/O CIP para Mensajería I/O multicast.

- **Conexiones de Mensajería Explícita:** Las Conexiones de Mensajería Explícitas proporcionan trayectorias de comunicación *genéricas*, multipropósito entre dos dispositivos. Estas conexiones se refieren a menudo a simples conexiones de mensajería. Los mensajes explícitos proporcionan la tradicional relación *solicitud/respuesta* orientada a las redes de comunicación. Este tipo de mensajes son **punto a punto**.

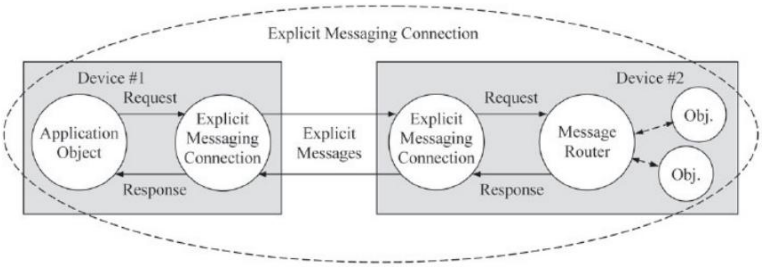


Figura 11. Conexión CIP para Mensajería Explícita punto a punto.

3.8. Manual de Programación: Etiquetas producidas y etiquetas consumidas de los controladores Logix 5000

Este apartado resalta los conceptos clave sobre el uso de *etiquetas producidas y consumidas* en los controladores **Logix 5000**, según el manual de programación. Se explica su función en la comunicación entre controladores y los parámetros básicos para su configuración.

3.8.1. Etiqueta Producida

Una etiqueta que un controlador *hace disponible para el uso de otros controladores*. Varios controladores pueden **consumir** (recibir) simultáneamente los datos. Una etiqueta **producida** envía sus datos a una o más etiquetas **consumidas** (consumidores) sin usar lógica.

3.8.2. Etiqueta Consumida

Una etiqueta que recibe datos de una etiqueta **producida**. El tipo de datos de la etiqueta **consumida** *debe coincidir* con el tipo de datos (incluida cualquier dimensión de matriz) de la etiqueta **producida**. El *RPI* de la etiqueta consumida determina el periodo de actualización de los datos.

Nota: Para que dos controladores *compartan etiquetas producidas o consumidas*, ambos controladores **deben estar** en el mismo backplane o conectados a la misma red de control.

3.8.3. Intervalo de Paquetes Solicitados (RPI)

El intervalo de paquetes solicitado para la conexión, medido por el tiempo (en ms) entre *actualizaciones de datos del controlador remoto*. Este valor es la **velocidad mínima** a la que el controlador local recibe o transfiere datos.

Nota: Cambiar el Intervalo de Paquetes Solicitados (RPI) mientras se está en línea *deshabilitará temporalmente la conexión*.

3.8.4. Limitaciones del RPI

En las versiones anteriores a la 28.00.00, una etiqueta **producida** generaba datos según el RPI del *consumidor más rápido*, permitiendo que múltiples consumidores con diferentes RPI se conectaran sin problema.

A partir de la versión 28.00.00, el **primer consumidor** que se conecta a una etiqueta producida *determina el RPI que usará el productor*. Todos los demás consumidores *deben coincidir exactamente con ese RPI o no podrán conectarse*, recibiendo el código de error **0112**.

El primer consumidor se define según el orden de encendido del sistema y **no puede configurarse manualmente**, lo que complica su identificación. Esto debe considerarse especialmente al usar **etiquetas producidas en modo multidifusión**.

Para evitar errores en sistemas con multidifusión la versión 28.00.00 en adelante, se recomienda: desde

- Asegurarse de que todos los consumidores usen *el mismo RPI*, o
- Habilitar la opción para que *las etiquetas consumidas acepten el RPI proporcionado por el productor*.

Versión de firmware	**Comportamiento del RPI**	**Condiciones y restricciones**
< 28.00.00	El productor usa el RPI del consumidor más rápido.	Se permite que múltiples consumidores tengan distintos RPI.
≥ 28.00.00	El primer consumidor determina el RPI.	Todos los consumidores deben usar el **mismo RPI** o se genera error **0112**. No se puede elegir quién es el primer consumidor.
Multidifusión (≥ 28.00.00)	Igual que en la versión correspondiente.	• Usar el mismo RPI. • O estar configurados para **aceptar el RPI del productor**.

Tabla 1. Resumen Limitaciones del RPI según Versión de Firmware.

3.8.5. Pautas de Organización de las Etiquetas

- Crear las etiquetas en el alcance del controlador (variables globales). **Sólo comparte** tags de alcance de controlador.
- Utilizar uno de los siguientes tipos de dato: DINT, REAL, Tabla de DINT o REAL, UDT (User Defined Tag).
- Limitar el tamaño de la etiqueta a *500 Bytes*. Si se excede, crear lógica para transferir por paquetes.
- Utilizar el RPI **más alto permitido** para la aplicación.
- Combinar los datos que se dirigen al mismo controlador. Si se está produciendo varios tags para el mismo controlador, agrupar los datos en uno o más tipos de datos definidos por el usuario. (Esto utiliza **menos conexiones** que la producción de cada etiqueta de forma separada.)

3.9. Manuales de Programación: Guía de inicio de mensajería de cliente CIP e Instrucciones generales de controladores programables Micro800.

Este apartado resalta los conceptos clave sobre el uso de *las instrucciones de mensajería* en los controladores Micro800 de Allen Bradley, según los manuales de programación. Se explicará cada uno de los parámetros solicitados por las instrucciones. Para los controladores de la familia MicroLogix se utilizará el software *Components Connected Workbench (CCW).*

3.9.1. Ruta de Destino o Path CIP

La ruta de destino (Path) para mensajes CIP contiene parámetros que determinan la ruta y el destino del mensaje CIP. El parámetro de cadena de ruta de destino utiliza la siguiente sintaxis:

"<puerto local>,<dirección del primer destino> ,<puerto local del primer destino>,

<dirección del segundo destino> "

A continuación, una descripción de los elementos de la sintaxis:

Elemento de cadena	Descripción
Puerto local	Puerto local utilizado para enviar el mensaje. Debe ser un puerto serie CIP o Ethernet/IP activo (no se admiten puertos USB).
Dirección del primer destino	Dirección de destino del primer salto. • Para EIP, se debe especificar la dirección IP de destino. Esta debe ser una dirección unicast y no puede ser 0, de multidifusión, de difusión, local ni de bucle invertido (127.x.x.x). • Para serie CIP, se debe especificar la dirección de nodo de destino. El valor admitido es 1.
Puerto local del primer destino	Puerto local utilizado para enviar el mensaje.
Dirección del segundo destino	Dirección de destino del segundo salto.

Figura 12. Elementos de la ruta de destino CIP.

A continuación, unos ejemplos de ruta de destino:

- **0,0:** El dispositivo de destino es el dispositivo local.
- **6,1:** A través del puerto 6 (identificador puerto serie UPM Micro800), se alcanza el nodo 1.
- **4,192.168.1.100 (Micro a Micro/Compact):** A través del puerto 4 (identificador puerto Ethernet integrado Micro800), se alcanza el nodo en 192.168.1.100 (dirección IP Compact/MicroLogix).
- **4,192.168.1.100,1,0 (Micro a Control):** A través del puerto 4 (identificador puerto Ethernet integrado Micro800), se alcanza el nodo en 192.168.1.100 (módulo ENET de Logix). Desde el módulo ENET, a través del puerto del backplane (identificador puerto 1), se alcanza el controlador Logix en la ranura/slot 0.
- **2,192.168.1.100,1,1 (Compact a Control):** A través del puerto 2 (identificador puerto Ethernet integrado CompactLogix), se alcanza el nodo en 192.168.1.100 (módulo ENET de Logix). Desde el módulo ENET, a través del puerto del backplane (identificador puerto 1), se alcanza el controlador Logix en la ranura/slot 1.
- **2,192.168.1.100 (Compact a Compact/Micro):** A través del puerto 2 (identificador puerto Ethernet integrado CompactLogix), se alcanza el nodo en 192.168.1.100 (dirección IP Compact/MicroLogix).
- **1,6,2,192.168.1.100,1,0 (Control a Control):** A través del puerto del backplane 1 (identificador puerto 1), se alcanza el módulo ENET Logix 1 en la ranura/slot 6. A través del puerto 2 (identificador puerto Ethernet integrado módulo ENET Logix 1), se alcanza el nodo en 192.168.1.100 (módulo ENET de Logix 2). Desde el módulo ENET Logix 2, a través del puerto del backplane 2 (identificador puerto 1), se alcanza el controlador Logix 2 en la ranura/slot 0.

- **1,6,2,192.168.1.100 (Control a Compact/Micro):** A través del puerto del backplane (identificador puerto 1), se alcanza el módulo ENET Logix en la ranura/slot 6. A través del puerto 2 (identificador puerto Ethernet integrado módulo ENET Logix), se alcanza el nodo en 192.168.1.100 (dirección IP Compact/MicroLogix).

3.9.2. Objetos CIP Micro800

En la Figura 13 se muestran los objetos CIP que pueden ser utilizados con los controladores Micro800:

Objeto	Código de clase
Identity Object	0x01 (01h)
Wall-Clock Time Object	0x8B (8Bh)
Modbus Serial Link Object	0x46 (46h)
TCP/IP Object	0xF5 (F5h)
Ethernet Link Object	0xF6 (F6h)
USB Object	0x33A (33Ah)
Symbol Object	0x6B (6Bh)

Figura 13. Objetos CIP Micro800.

A continuación, se proporcionará información del Identity Object, ya que este se utilizará en la implementación de la práctica:

Identity Object **Código de clase: 0x01 (01h)**

Este objeto proporciona la identificación e información general del dispositivo. El Identity Object está presente en todos los productos CIP.

Figura 14. Descripción Identity Object.

Atributo de clase

ID de atributo	Regla de acceso	Nombre	Tipo de datos	Descripción	Valores	Valores predeterminados
1	Solo lectura	Revision	UINT	Revisión		0x01
2	Solo lectura	Max Instance	UINT	Número máximo de instancias de un objeto actualmente creado en este nivel de clase del dispositivo	01	0x01
3	Solo lectura	Number of Instances	UINT	Número de instancias en esta jerarquía de clases	01	0x01

Figura 15. Atributo de Clase Identity Object.

Atributo de instancia

Se acepta solo una instancia (Instance ID 01).

ID de atributo	Regla de acceso	Nombre	Tipo de datos	Descripción	Valores	Valores predeterminados
1	Solo lectura	Vendor ID	UINT	Identificación de cada proveedor por número	01 (Allen-Bradley)	0x01
2	Solo lectura	Device Type	UINT	Identificación del tipo general de producto	0x0E (controlador lógico programable)	0x0E
3	Solo lectura	Product Code	UINT	Identificación de un determinado producto de un proveedor individual		
4	Solo lectura	Revision	Struct of:	Revisión del ítem que el objeto de identidad representa		Producto/revisión específicos
		Major Revision	USINT			
		Minor Revision	USINT			
5	Solo lectura	Status	WORD	Estado de resumen del dispositivo	Vea las especificaciones CIP, Volumen 1	0x04 (configurado)
6	Solo lectura	Serial Number	UDINT	Número de serie del dispositivo		4 bytes únicos
7	Solo lectura	Product Name	SHORT-STRING	Identificación legible por el usuario		Ejemplo: 2080-LC30-16QVB
8	Solo lectura	State	WORD	Estado actual del dispositivo representado por el diagrama de transición de estado	Vea las especificaciones CIP, Volumen 1	0x02 (en espera)
11[1]	Lectura/ escritura	Active Language	Struct of:	Idioma actualmente activo del dispositivo	"eng" (English)	
			USINT	El campo language1 del tipo de datos STRING		"e"
			USINT	El campo language2 del tipo de datos STRING		"n"
			USINT	El campo language3 del tipo de datos STRING		"g"

Figura 16. Atributo de Instancia Identity Object.

ID de atributo	Regla de acceso	Nombre	Tipo de datos	Descripción	Valores	Valores predeterminados
12(2)	Solo lectura	Lista de idiomas compatibles	Array of Struct of:	Lista de idiomas compatibles por cadenas de carácter del tipo de datos STRING dentro del dispositivo	Controladores Micro830 y Micro850 "eng" (inglés) Controlador Micro820 1 = "eng" (inglés) 2 = "deu" (alemán) 3 = "fra" (francés) 4 = "spa" (español) 5 = "ita" (italiano) 6 = "por" (portugués) 7 = "dut" (holandés) 8 = "swe" (sueco) 9 = "pol" (polaco) 10 = "tur" (turco) 11 = "cze" (checo) 12 = "hun" (húngaro) 13 = "chi" (chino) 14 = "jpn" (japonés)	
			USINT	El campo language1 del tipo de datos STRING		
			USINT	El campo language2 del tipo de datos STRING		
			USINT	El campo language3 del tipo de datos STRING		
15(1)	Lectura/ escritura	Assigned_Name	STRINGI	Nombre asignado al usuario	Unicode de 2 bytes del tipo de datos STRING2 incorporado. 64 bytes máx. (32 caracteres Unicode)	En blanco
16(1)	Lectura/ escritura	Assigned_Description	STRINGI	Descripción asignada al usuario	Unicode de 2 bytes del tipo de datos STRING2 incorporado. 100 bytes máx. (50 caracteres Unicode)	En blanco
17(1)	Lectura/ escritura	Assigned_Geographic Descripción	STRINGI	Lugar asignado al usuario	Unicode de 2 bytes del tipo de datos STRING2 incorporado. 64 bytes máx. (32 caracteres Unicode)	En blanco

(1) No volátil
(2) Volátil

Figura 17. Atributo de Instancia Identity Object.

Servicios

Código de servicio	Implementado para:		Nombre de servicio
	Clase	Instancia	
0x01	Sí	Sí	Get_Attribute_All
0x05	Sí	Sí	Reset
0x0E	Sí	Sí	Get_Attribute_Single
0x10	Sí	Sí	Set_Attribute_Single
0x18	Sí	Sí	Get_Member

Figura 18. Servicios Identity Object.

3.9.3. Bloque de Función MSG_CIPSYMBOLIC

En CCW, para realizar la comunicación CIP mediante Mensajería Implícita, se utiliza la instrucción de mensajería simbólica.

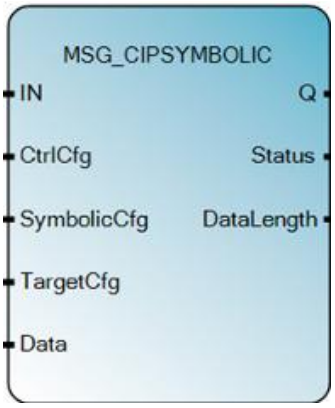


Figura 19. Instrucción MSG_CIPSYMBOLIC.

A continuación, los parámetros de la instrucción MSG_CIPSYMBOLIC:

Parámetros MSG_CIPSYMBOLIC

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
IN	Entrada	BOOLEAN	En caso de flanco ascendente (IN cambia de 0 a 1), inicie el bloque de funciones con la condición previa de que la última operación se haya concluido. Cuando se habilita el bloque de funciones, se borran los búferes de recepción para las operaciones de lectura con el flanco ascendente de IN.
CtrlCfg	Entrada	CIPCONTROLCFG	Configuración de control. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • CtrlCfg.Cancel (Boolean) – Cuando este valor es verdadero, se anula la ejecución de MSG_CIPSYMBOLIC – Se borra el bit cuando se habilita el mensaje • CtrlCfg.TriggerType (UDINT) – Si se establece en "0", el mensaje se dispara una sola vez cuando "IN" es verdadero – Si se establece en 1 a 65535 (unidad: milisegundos), se dispara periódicamente cuando "IN" es verdadero. Por ejemplo, un valor de 100 significa que cuando "IN" es verdadero, el mensaje se dispara cada 100 milisegundos • CtrlCfg.StrMode (USINT) – Reservado.

Figura 20. Parámetros Instrucción MSG_CIPSYMBOLIC.

Parámetros MSG_CIPSYMBOLIC

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
SymbolicCfg	Entrada	CIPSYMBOLICCFG	Información de símbolo a leer/escribir. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • SymbolicCfg.Service (USINT) Servicio de lectura/escritura 0 – Lectura (valor predeterminado) 1 – Escritura • SymbolicCfg.Symbol (STRING) Nombre de la variable a leer/escribir. Este campo no puede estar en blanco. Se puede introducir un máximo de 80 caracteres. *Ejemplo 1*: para leer una matriz MyArray[1..100] debemos introducir SymbolicCfg.Symbol := 'MyArray'; *Ejemplo 2*: para leer una matriz MyArray[1..100], comience a partir del quinto elemento de la matriz y seguidamente SymbolicCfg.Symbol := 'MyArray[5]'; • SymbolicCfg.Count (UINT) Número de elementos variables a leer/escribir. 1...65535, 1 se usa si este valor se establece en 0 (valor predeterminado). • SymbolicCfg.Type (USINT) Valor de tipo de datos de los datos que se leen/escriben. Este campo no puede estar en blanco para escritura. Para lectura, la ejecución del bloque de funciones llena este campo con el tipo de datos recibido. Para escritura, el usuario especifica el tipo de datos a escribir. Vea Tipos de datos de mensaje CIP en la página 65. • SymbolicCfg.Offset (USINT) Reservado para uso futuro. Offset de byte de la variable a leer/escribir. Este campo se usa para leer/escribir una variable de gran tamaño que no se puede procesar en un solo mensaje. Rango: 0...255.

Figura 21. Parámetros Instrucción MSG_CIPSYMBOLIC.

Data type	Data type value (hexadecimal)	Description
BOOL	193 (0xC1)	Logical Boolean with values TRUE (1) and FALSE (0)
SINT	194 (0xC2)	Signed 8-bit integer value
INT	195 (0xC3)	Signed 16-bit integer value
DINT	196 (0xC4)	Signed 32-bit integer value
LINT	197 (0xC5)	Signed 64-bit integer value
USINT	198 (0xC6)	Unsigned 8-bit integer value
UINT	199 (0xC7)	Unsigned 16-bit integer value
UDINT	200 (0xC8)	Unsigned 32-bit integer value
ULINT	201 (0xC9)	Unsigned 64-bit integer value
REAL	202 (0xCA)	32-bit floating point value
LREAL	203 (0xCB)	64-bit floating point value
STRING	218 (0xDA)	Character string

Figura 22. Tipos de datos de mensajes CIP.

Nota: Esta tabla sirve para el atributo *Type* para la operación de escritura simbólica.

Parámetros MSG_CIPSYMBOLIC

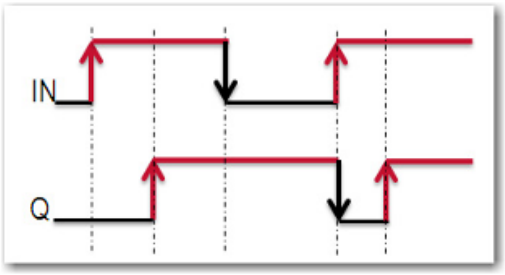
Parámetro	Tipo de parámetro	Tipo de datos	Descripción
TargetCfg	Entrada	CIPTARGETCFG	Configuración del dispositivo de destino. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • TargetCfg.Path (String) Información de destino. Se puede especificar un máximo de dos saltos. {"<port>,<node/slot address>"} *Ejemplo 1*: Desde el puerto Ethernet incorporado en Micro850 (puerto 4), envíe un mensaje a un dispositivo dentro de la misma red de subconjunto con la dirección IP 192.168.1.100. La ruta es TargetCfg.Path := '4, 192.168.1.100' *Ejemplo 2*: Desde el puerto serial incorporado en Micro830 (puerto 2), para alcanzar un dispositivo en nodo 1. TargetCfg.Path := '2, 1' • TargetCfg.CipConnMode (USINT) Tipo de conexión CIP. '0' – Desconectado (predeterminado), '1' – Conexión de clase 3 • TargetCfg.UcmmTimeout (UDINT) Tiempo de espera de mensaje desconectado (en milisegundos) Cantidad de tiempo que se espera para recibir una respuesta para mensajes desconectados (incluso el establecimiento de conexión para un mensaje conectado) Rango: 250...10,000; establezca en 0 para usar el valor predeterminado de 3000. Si se especifica cualquier valor menor que 250 se establece en 250, y cualquier valor mayor que el valor máximo se establece en 10,000. • TargetCfg.ConnMsgTimeout (UINT) Tiempo de espera de conexión de clase 3 (en milisegundos) Cantidad de tiempo que se espera para recibir una respuesta para mensajes conectados. Se cierra la conexión CIP si expira este tiempo de espera. Al usarse la comunicación en puente o serial, el valor se debe establecer en 880 o más. Rango: 800...10,000; establezca en 0 para usar el valor predeterminado de 10,000. Si se especifica cualquier valor menor que 800 se establece en 800, y cualquier valor mayor que el valor máximo se establece en 10,000. • TargetCfg.ConnClose (UINT) Comportamiento de cierre de conexión. Verdadero: Cierre la conexión CIP al concluirse el mensaje. Falso: No cierre la conexión al concluirse el mensaje [predeterminado].
Data	Entrada	Matriz USINT	Este es un parámetro de entrada/salida. El comando de lectura almacena los datos devueltos del servidor. El comando de escritura sirve como búfer de los datos que se envían al servidor. Cuando se dispara (o se vuelve a disparar) un MSG, se borran los datos del comando de lectura de MSG. Rango: 1...490
Q	Salida	BOOLEAN	Las salidas de esta instrucción se actualizan asíncronamente desde el escán de programa. La salida Q no se puede usar para volver a disparar la instrucción ya que IN se dispara con flanco. TRUE: se concluyó debidamente la instrucción MSG. FALSE: no se ha concluido la instrucción MSG. Después de que 'Q' se hace verdadero, incluso cuando 'IN' se haga falso, Q permanece verdadero hasta que 'IN' se vuelva a disparar. 

Figura 23. Parámetros Instrucción MSG_CIPSYMBOLIC.

Parámetros MSG_CIPSYMBOLIC

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
Status	Salida	CIPSTATUS	Estado de ejecución de la instrucción. Cuando se dispara (o se vuelve a disparar) un MSG, se restablecen todos los elementos dentro de Status. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • Status.Error (Boolean) Este bit se establece en TRUE cuando la ejecución de un bloque de funciones detecta una condición de error. • Status.ErrorID (UINT) Valor de código de error. Vea Códigos de mensaje de error CIP en la página 63. • Status.SubErrorID (UDINT) Valor de código de suberror. Vea Códigos de mensaje de error CIP en la página 63. • Status.ExtErrorID (UINT) Valor de código de error de estado extendido CIP • Status.StatusBits (UINT) - - - - - - - - - - - 4 3 2 1 0 Este parámetro se puede usar para verificar varios bits de control. – Bit 0: EN – Habilitar – Bit 1: EW – Habilitar espera – Bit 2: ST – Arrancar – Bit 3: ER – Errores (mismo estado que Q) – Bit 4: DN – Concluido (mismo estado que Q) – Los otros bits están reservados.
DataLength	Salida	UINT	Longitud de datos de respuesta de lectura de mensaje CIP y longitud de datos de solicitud de escritura de mensaje de CIP (unidad: byte). Cuando se dispara (o se vuelve a disparar) una lectura de MSG, se restablece DataLength en 0 para el comando de lectura de MSG. Rango: 0…490

Figura 24. Parámetros Instrucción MSG_CIPSYMBOLIC.

3.9.4. Bloque de Función MSG_CIPGENERIC

En CCW, para realizar la comunicación CIP mediante Mensajería Explícita, se utiliza la instrucción de mensajería genérica.

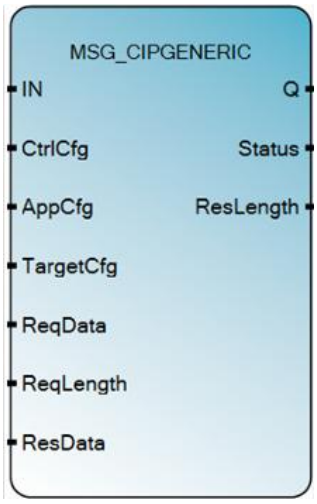


Figura 25. Instrucción MSG_CIPGENERIC.

A continuación, los parámetros de la instrucción MSG_CIPGENERIC:

Parámetros MSG_CIPGENERIC

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
IN	Entrada	BOOLEAN	En caso de flanco ascendente (IN cambia de 0 a 1), inicie el bloque de funciones con la condición previa de que la última operación se haya concluido. Cuando se habilita el bloque de funciones, se borran los búferes de recepción para las operaciones de lectura con el flanco ascendente de IN.
CtrlCfg	Entrada	CIPCONTROLCFG	Configuración de control. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • CtrlCfg.Cancel (Boolean) – Cuando este valor es verdadero, se anula la ejecución de MSG_CIPGENERIC. – Se borra el bit cuando se habilita el mensaje. • CtrlCfg.TriggerType (UDINT) – Si se establece en "0", el mensaje se dispara una sola vez cuando "IN" es verdadero. – Si se establece en 1 a 65535 (unidad: milisegundos), se dispara periódicamente cuando "IN" es verdadero. Por ejemplo, un valor de 100 significa que cuando "IN" es verdadero, el mensaje se dispara cada 100 milisegundos • CtrlCfg.StrMode (USINT) – Reservado.

Figura 26. Parámetros Instrucción MSG_CIPGENERIC.

Parámetros MSG_CIPGENERIC

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
AppCfg	Entrada	CIPAPPCFG	Configuración de servicio CIP y ruta de aplicación. Este es un tipo de datos estructurado que consiste en los elementos siguientes: • AppCfg.Service (USINT) – Servicio de código • AppCfg.Class (UINT) – Valor de ID de clase del segmento lógico • AppCfg.Instance (UDINT) – Valor e ID de instancia del segmento lógico • AppCfg.Attribute (UINT) – Valor de atributo del segmento lógico • AppCfg.MemberCnt (UINT) – Máximo de valores de ID de miembro usados (normalmente no se usa este parámetro, lo que significa que se establece en 0); 0 = No se usa una ID de miembro • AppCfg.MemberId (UINT) – Valores de ID de miembro (normalmente no se usa este parámetro)

Figura 27. Parámetros Instrucción MSG_CIPGENERIC.

TargetCfg	Entrada	CIPTARGETCFG	Configuración del dispositivo de destino. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • TargetCfg.Path (String) Información de destino. Se puede especificar un máximo de dos saltos. {"<port>,<node/slot address>"} *Ejemplo 1*: Desde el puerto Ethernet incorporado en Micro850 (puerto 4), envíe un mensaje a un dispositivo dentro de la misma red de subconjunto con la dirección IP 192.168.1.100. La ruta es TargetCfg.Path := '4, 192.168.1.100' *Ejemplo 2*: Desde el puerto serial incorporado en Micro830 (puerto 2), para alcanzar un dispositivo en nodo 1. TargetCfg.Path := '2, 1' • TargetCfg.CipConnMode (USINT) Tipo de conexión CIP. '0' – Desconectado (predeterminado), '1' – Conexión de clase 3 • TargetCfg.UcmmTimeout (UDINT) Tiempo de espera de mensaje desconectado (en milisegundos) Cantidad de tiempo que se espera para recibir una respuesta para mensajes desconectados (incluso el establecimiento de conexión para un mensaje conectado) Rango: 250…10,000; establezca en 0 para usar el valor predeterminado de 3000. Si se especifica cualquier valor menor que 250 se establece en 250, y cualquier valor mayor que el valor máximo se establece en 10,000. • TargetCfg.ConnMsgTimeout (UINT) Tiempo de espera de conexión de clase 3 (en milisegundos) Cantidad de tiempo que se espera para recibir una respuesta para mensajes conectados. Se cierra la conexión CIP si expira este tiempo de espera. Al usarse la comunicación en puente o serial, el valor se debe establecer en 880 o más. Rango: 800…10,000, establezca en 0 para usar el valor predeterminado de 10,000. Si se especifica cualquier valor menor que 800 se establece en 800, y cualquier valor mayor que el valor máximo se establece en 10,000. • TargetCfg.ConnClose (UINT) Comportamiento de cierre de conexión. Verdadero: Cierre la conexión CIP al concluirse el mensaje. Falso: No cierre la conexión al concluirse el mensaje [predeterminado].
ReqData	Entrada	Matriz USINT	Datos de solicitud de mensaje CIP (los datos a escribir al destino). El tamaño de la matriz no debe ser menor que el tamaño de 'ReqLength'.
ReqLength	Entrada	UINT	Longitud de datos de solicitud de mensaje CIP (unidad: byte). Número de bytes a escribir al destino. Rango: 0…490

Figura 28. Parámetros Instrucción MSG_CIPGENERIC.

Parámetros MSG_CIPGENERIC

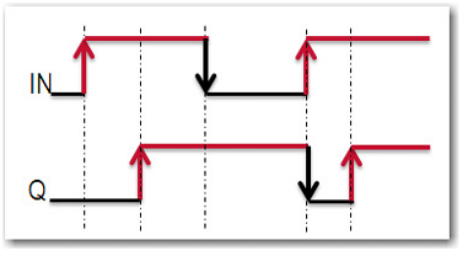
Parámetro	Tipo de parámetro	Tipo de datos	Descripción
ResData	Entrada	USINT	Datos de respuesta de mensaje CIP. El tamaño de la matriz no debe ser menor que el tamaño de 'ResLength'. Cuando se dispara (o se vuelve a disparar) un MSG, se borran los datos en la matriz ResData. Este es un parámetro de entrada/salida.
Q	Salida	BOOLEAN	Las salidas de esta instrucción se actualizan asíncronamente desde el escán de programa. La salida Q no se puede usar para volver a disparar la instrucción ya que IN se dispara con flanco. TRUE: se concluyó debidamente la instrucción MSG. FALSE: no se ha concluido la instrucción MSG. Después de que 'Q' se hace verdadero, incluso cuando 'IN' se haga falso, Q permanece verdadero hasta que 'IN' se vuelva a disparar. 

Figura 29. Parámetrs Instrucción MSG_CIPGENERIC.

Status	Salida	CIPSTATUS	Estado de ejecución de la instrucción. Cuando se dispara (o se vuelve a disparar) un MSG, se restablecen todos los elementos dentro de Status. Este es un tipo de datos estructurado. Consiste en los elementos siguientes: • Status.Error (Boolean) – Este bit se establece en TRUE cuando la ejecución de un bloque de funciones detecta una condición de error. • Status.ErrorID (UINT) – Valor de código de error. Vea Códigos de mensaje de error CIP en la página 63. • Status.SubErrorID (UDINT) – Valor de código de suberror. Vea Códigos de mensaje de error CIP en la página 63. • Status.ExtErrorID (UINT) – Valor de código de error de estado extendido CIP • Status.StatusBits (UINT) - - - - - - - - - - - 4 3 2 1 0 Este parámetro se puede usar para verificar varios bits de control. – Bit 0: EN – Habilitar – Bit 1: EW – Habilitar espera – Bit 2: ST – Arrancar – Bit 3: ER – Error (mismo estado que Q) – Bit 4: DN – Concluido (mismo estado que Q) – Los otros bits son reservados.
ResLength	Salida	UINT	Longitud de datos de respuesta de mensaje CIP (unidad: byte). Cuando se dispara (o se vuelve a disparar) un MSG, se restablece ResLength en 0 Rango: 0...490

Figura 30. Parámetros Instrucción MSG_CIPGENERIC.

3.9.5. Bloque de Función COP

En CCW, esta instrucción se puede usar para *copiar* los datos binarios en la matriz de origen (Src) a la matriz de destino (Dest). *No se cambia el origen*. Esta función se utilizará *conjuntamente con las instrucciones de mensaje CIP* para convertir los datos del original, por ejemplo, tipo de datos de matriz DINT, REAL, al formato de datos de envío con el apoyo de la instrucción de mensaje CIP (es decir, matriz **USINT**).

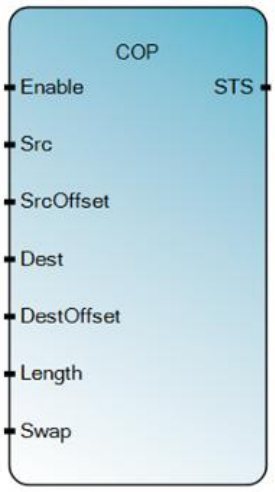


Figura 31. Instrucción COP.

Parámetros COP

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
Enable	Entrada	BOOLEAN	Habilitación de bloque de funciones. Este bloque de funciones se dispara por nivel. Cuando Enable=TRUE, haga una copia. Cuando Enable=FALSE, no se ejecuta el bloque de funciones.

Figura 32. Parámetos Instrucción COP.

Parámetros COP

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
Src	Entrada	Tipos de datos compatibles:(1) Boolean, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, DATE, STRING, LWORD, ULINT, LINT, LREAL	Elemento inicial a copiar. Si el origen o el destino es String Data, la otra parte debe ser String Data o USINT (UCHAR y BYTE se consideran el mismo tipo de datos). Si no es así, se indica una incompatibilidad del tipo de datos. Cuando se copie a o de String Data, se usará el formato String corto ODVA para los datos en la matriz USINT. Para la instrucción COP entre cualquier par de tipos de datos (sin datos String como origen o destino), se considera válida la operación de copiar, aunque es posible que los datos en el destino estén en un formato no válido. El usuario necesita manejar la lógica en el nivel de aplicación. Para copiar la matriz USINT a una matriz String, los datos en la matriz USINT tienen que estar en el formato: – Byte1: Longitud de la primera cadena, – Byte2: Primer carácter de byte, – Byte3: Segundo carácter de byte, – Byte n: Último carácter de byte, – Byte (n+1): Longitud de la segunda cadena, – Byte (n+2): Primer carácter de byte para la segunda cadena, y así sucesivamente.
SrcOffset	Entrada	UINT	Offset de elemento de origen. Offset de elemento si el origen es un tipo de datos de matriz; si no es así, se debe establecer en 0. En el caso de un tipo de datos de matriz, para copiar desde el primer elemento el offset se debe establecer en 0.
Dest	Entrada	Tipos de datos compatibles(1): Boolean, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, DATE, STRING, LWORD, ULINT, LINT, LREAL	Destino de copia.
DestOffset	Entrada	UINT	Offset de elemento de destino. Offset de elemento si el destino es de tipo de datos de matriz; si no es así, se debe establecer en 0. En el caso de un tipo de datos de matriz, para copiar al primer elemento el offset se debe establecer en 0.
Length	Entrada	UINT	Número de elementos de destino a copiar. Cuando el destino es de tipo de datos de cadena, significa el número de cadenas a copiar.
Swap	Entrada	BOOLEAN	TRUE: Intercambie los bytes según el tipo de datos. Si el origen o el destino son datos de cadena, no hay intercambio. Si el origen y el destino son datos de longitud de un byte, no hay intercambio.
STS	Salida	UINT	0: Bloque de funciones no habilitado. Sin operación. 1: Éxito de la operación COP 2: El destino tiene bytes sobrantes al copiar desde la cadena (si se considera que la copia se realizó debidamente). 3: Se truncan los datos de origen (si se considera que la copia se realizó debidamente). 4: La longitud de copia no es válida. 5: Incompatibilidad de tipo de datos cuando hay un tipo de datos de cadena como origen o destino. 6: El tamaño de datos de origen es demasiado pequeño para copiar. 7: El tamaño de datos de destino es demasiado pequeño para copiar. 8: El offset de datos de origen no es válido. 9: El offset de datos de destino no es válido 10: Los datos no son válidos en el origen o el destino.

(1) Src y Dest deben estar en el formato de matriz únicamente si se requiere copiar una variable; posteriormente es necesario definirla en formato de matriz (por ejemplo, variable[1..1]). La instrucción COP no es compatible con las variables estructuradas.

Figura 33. Parámetos Instrucción COP.

3.10. Manuales de Programación: Manual de referencia Instrucciones generales de los controladores Logix 5000

Para los controladores de las familias Compact y ControlLogix se utilizará el software *Studio 5000 Logix Designer.*

3.10.1. Bloque de Función MSG

En Studio 5000, la instrucción MSG lee de manera asíncrona o escribe un bloque de datos en otro módulo de red. Se puede configurar para varios tipos de mensajería, incluyendo la simbólica y genérica.

Figura 34. Instrucción MSG.

.ER	BOOL	El bit de error se establece cuando el controlador detecta que ha fallado una transferencia. El bit .ER se restablecerá la próxima vez que EnableIn cambie de falso a verdadero. **Importante:** No cambie el bit .ER. El controlador ignora el cambio y usa los valores del los bit almacenados de manera interna.
.DN	BOOL	El bit de efectuado se establece cuando el último paquete de mensaje se ha transferido con éxito. El bit .DN se restablecerá la próxima vez que EnableIn cambie de falso a verdadero. **Importante:** No cambie el bit .DN. El controlador ignora el cambio y usa los valores del los bit almacenados de manera interna.
.ST	BOOL	El bit de comienzo se establece cuando el controlador comienza a ejecutar la instrucción MSG. El bit .ST se restablece cuando se activa el bit .DN o el .ER. **Importante:** No cambie el bit .ST. El controlador ignora el cambio y usa los valores del los bit almacenados de manera interna.
.EN	BOOL	El bit de habilitación se establece cuando EnableIn se establece en verdadero y permanece establecido hasta que se establezca el bit .DN o el .ER y EnableIn sea falso. Si EnableIn se borra y se establece en falso, pero los bits .DN y .ER se borran, el bit .EN permanece activado. **Importante:** No cambie el bit .EN. El controlador ignora el cambio y usa los valores del los bit almacenados de manera interna.

Figura 35. Parámetros Instrucción MSG.

.Destination Link	INT	Para cambiar el vínculo de Destination de DH+ o CIP con el mensaje de ID de Source, establezca este miembro en el valor necesario.	
.Destination Node	INT	Para cambiar el nodo de Destination de DH+ o CIP con el mensaje de ID de Source, establezca este miembro en el valor necesario.	
.SourceLink	INT	Para cambiar el vínculo de Source de DH+ o CIP con el mensaje de ID de Source, establezca este miembro en el valor necesario.	
.Class	INT	Para cambiar la clase del parámetro de un mensaje de CIP genérico, establezca este miembro en el valor necesario.	
.Attribute	INT	Para cambiar el atributo del parámetro de un mensaje de CIP genérico, establezca este miembro en el valor necesario.	
.Instance	DINT	Para cambiar el parámetro Instance de un mensaje de CIP genérico, se establece este miembro en el valor necesario.	
.LocalIndex	DINT	Si usa un asterisco (*) para designar un número de elemento en la matriz local, LocalIndex le dará el número de elemento. Para cambiar el número de elemento, establezca este miembro en el valor necesario.	
		Si el mensaje:	**La matriz local es:**
		Lee datos	Elemento de Destination
		Escribe datos	Elemento de Source

Figura 36. Parámetros Instrucción MSG.

.Path	STRING	Para mandar el mensaje a un controlador diferente, se establece este miembro en la nueva ruta. Se introduce la ruta como un valor hexadecimal. Omita los puntos (.). Por ejemplo, para una ruta que sea 1, 0, 2, 42, 1, 3, escriba $01$00$02$2A$01$03. Para examinar un dispositivo y crear automáticamente una porción o un todo de la nueva cadena, haga clic en la etiqueta de cadena y elija Ir al editor de ruta de mensaje (Go to Message Path Editor).	
.RemoteIndex	DINT	Si usa un asterisco (*) para designar un número de elemento en la matriz remota, RemoteIndex le dará el número de elemento. Para cambiar el número de elemento, establezca este miembro en el valor necesario.	
		Si el mensaje	**Entonces la matriz remota es**
		Lee datos	Elemento de Source
		Escribe datos	Elemento de Destination

Figura 37. Parámetros Instrucción MSG.

3.10.2. Bloques de Función COP y CPS

Las instrucciones COP (Copiar Archivo) y CPS (Copiar Archivo Sincrónico) copian los valores de Source en Dest. Source permanece sin cambios.

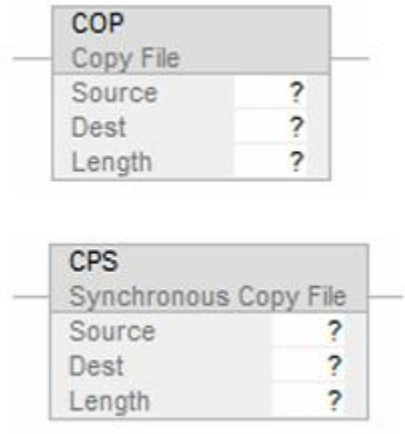


Figura 38. Instrucciones COP y CPS.

Operando	Tipo de datos	Formato	Descripción
Source	SINT INT DINT LINT REAL Tipo de cadena estructura	etiqueta	Elemento inicial para copiar
Dest	SINT INT DINT LINT REAL Tipo de cadena estructura	etiqueta	Elemento inicial a sobrescribir por Source
Length	SINT INT DINT	inmediato etiqueta	Número de elementos de Destination para copiar

Figura 39. Parámetros Instrucciones COP y CPS.

Durante la ejecución de las instrucciones COP y CPS, otras acciones del controlador pueden intentar interrumpir la operación de copia y cambiar el origen:

Si el origen o el destino es:	Y necesita:	Entonces selecciona:	Notas
• etiqueta producida • etiqueta consumida • datos de E/S • datos que otra tarea puede sobrescribir	Evitar que los datos de origen cambien durante la operación de copia	CPS	Las tareas que intentan interrumpir una instrucción CPS se retardan hasta que se realiza la instrucción. Para calcular el tiempo de ejecución de la instrucción CPS, consulte el Manual del usuario del sistema ControlLogix, publicación 1756-UM001.
	Permitir que los datos de origen cambien durante la operación de copia	COP	
Ninguno de los anteriores	---------->	COP	

Figura 40. Escenarios de Uso COP y CPS.

Para efectos de esta práctica se utilizará la instrucción COP, pero en un escenario más crítico debería utilizarse CPS.

4. Arquitectura de Comunicación

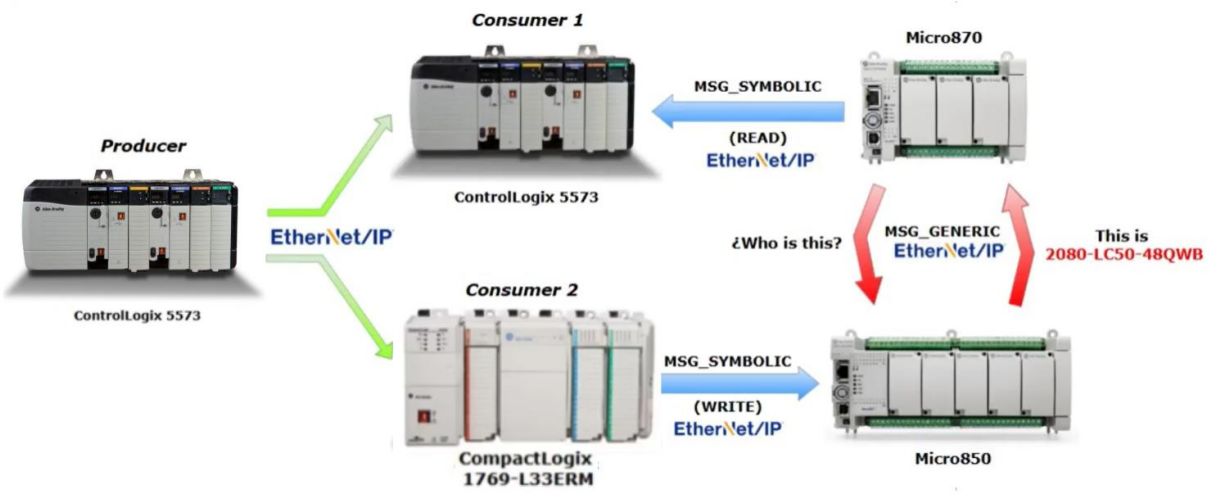


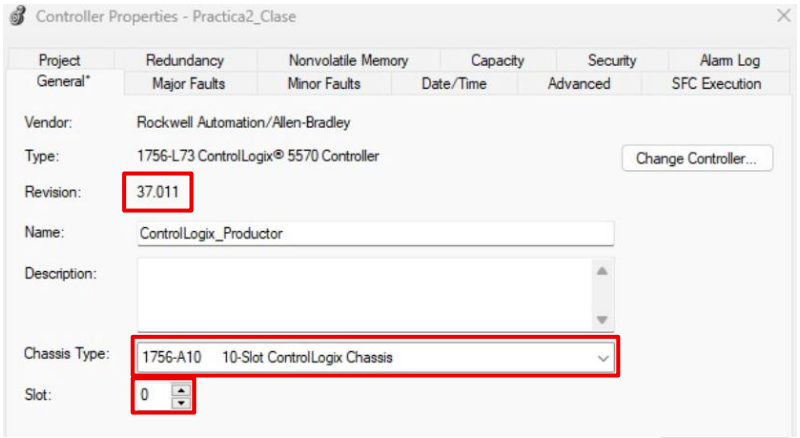
Figura 41. Arquitectura Protocolo CIP.

5. Procedimiento

En este apartado se explica la configuración paso a paso de las etiquetas producidas/consumidas, de las instrucciones de mensajería simbólica (lectura y escritura) y genérica, tanto en *CCW* como en *Studio 5000 Logix Designer*, con el objetivo de implementar la arquitectura de comunicación propuesta.

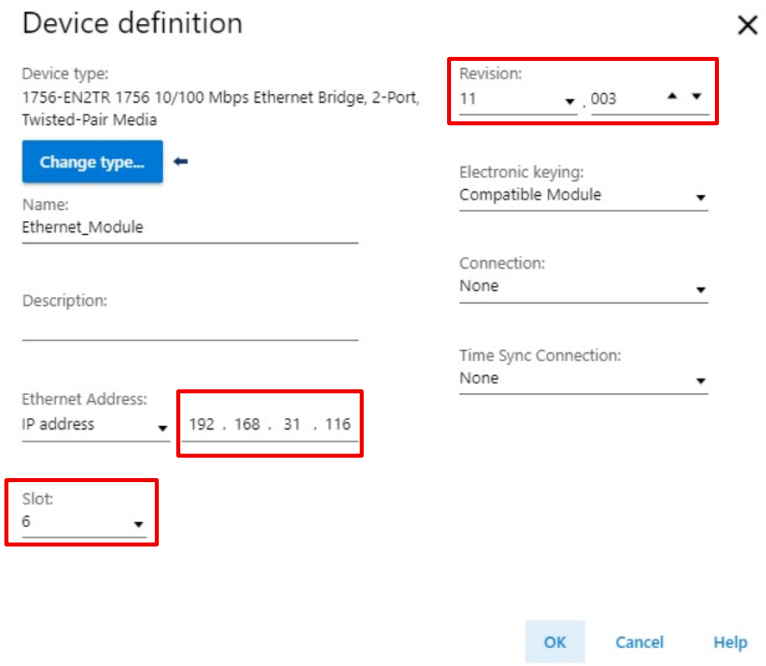
5.1. Etiqueta Producida

- Inicialmente, se debe crear un proyecto en Studio 5000 Logix Designer para el Productor, el cual es un ControlLogix con CPU modelo Logix5573 o L73. Considerar la Revisión de Firmware, Slot del Controlador y Tamaño del Chasis.

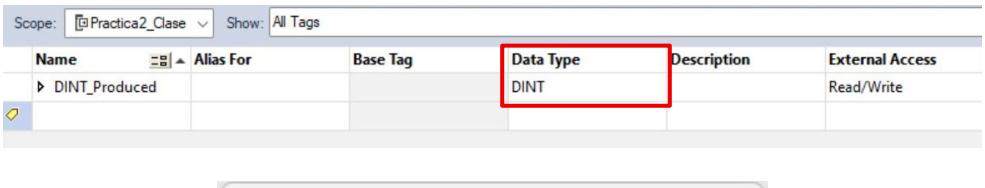


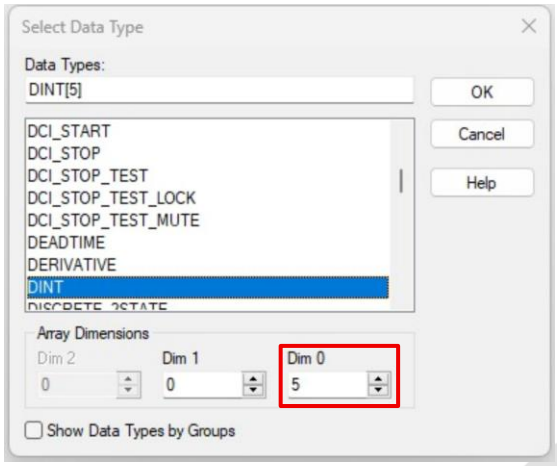
Nota: Apóyese del uso del software RSLinx Classic o FactoryTalk Linx para revisar las Propiedades de los Dispositivos/Configuración del Módulo (Número de Catálogo, Revisión de Firmware, Serie, Slot, Direccionamiento IP, etcétera).

- Luego de crear el proyecto, añadir el Módulo de Ethernet de nuestro PLC y configurarlo correctamente. En este caso se añade el módulo 1756-EN2TR/C del ControlLogix.



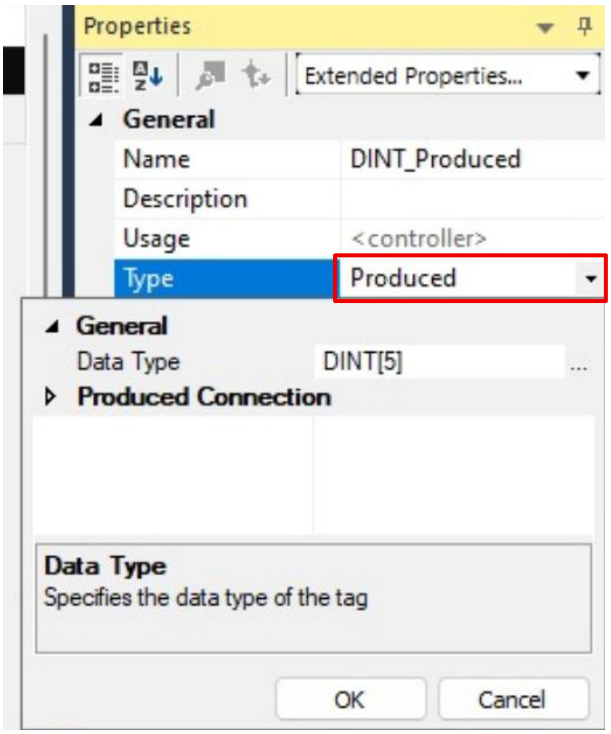
- Dirigirse a las Etiquetas del Controlador (Controller Tags), y crear una matriz de tipo DINT de 5 posiciones titulada “DINT_Produced”.



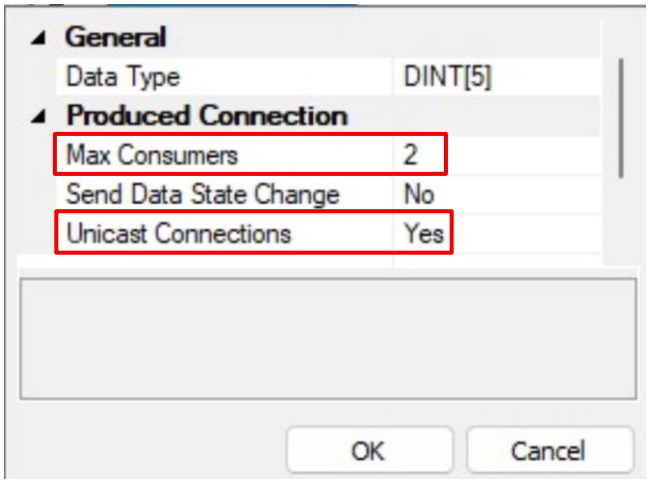


Nota: Recordar que las *etiquetas producidas/consumidas* **sólo pueden definirse como variables globales** (controller tags) o del alcance del controlador (scope de todo el proyecto).

- Dirigirse a las Propiedades de la etiqueta creada, seleccionar la propiedad *Type* y escoger el tipo **Produced**.



- Configurar la *Máxima Cantidad de Consumidores* que pueden acceder a esta etiqueta producida (en nuestro caso serían 2). Dejar por default la opción *Send Data State Change*. Seleccionar la opción *Yes* para *Unicast Connections* (automáticamente se coloca en Yes si Max Consumers > 1).



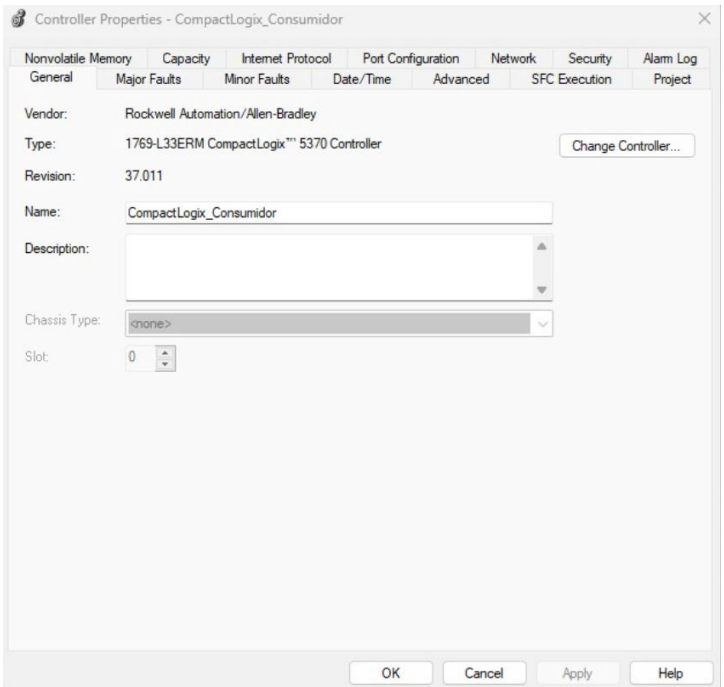
- Realizar la configuración del RPI en la sub pestaña de *Multicast*. Dejar por default el mínimo y máximo RPI. Seleccionar *Yes* en la opción *Use Default* para que el Productor asigne automáticamente el RPI a los Consumidores (el cual se configura en Default RPI). Escoger un valor de RPI de 100 ms.

Nota: Si el Productor será el encargado de asignar el RPI a la red, cada Consumidor *debe habilitar una opción* para que esto sea posible. Esto se encuentra en las *propiedades avanzadas* del controlador.

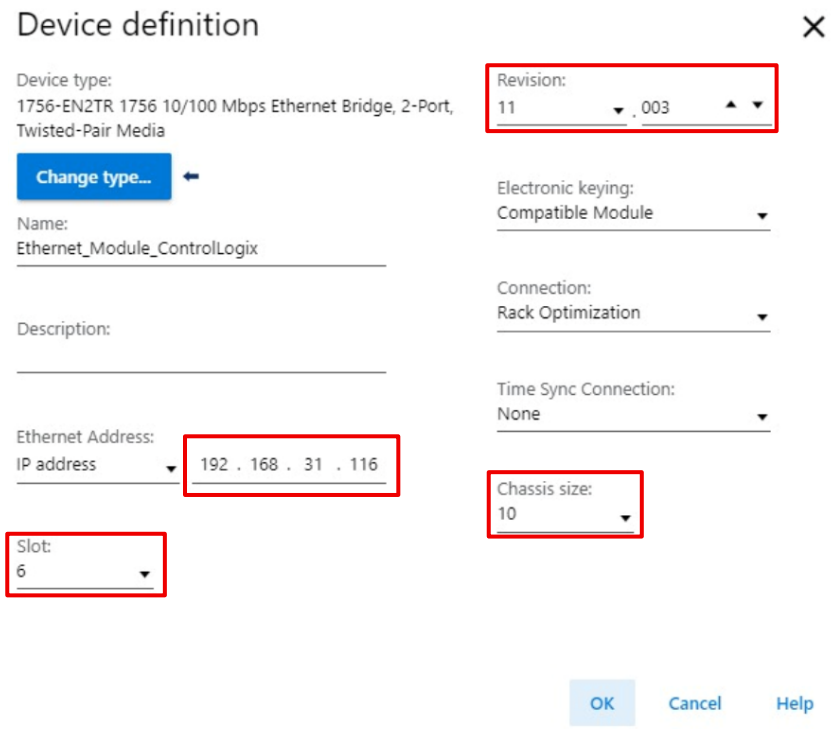
Si por el contrario *Use Default* se selecciona en *No*, cada Consumidor *debe configurar manualmente su RPI*, y para que la sincronización exista en la red **todos deben tener el mismo valor**.

5.2. Etiqueta Consumida

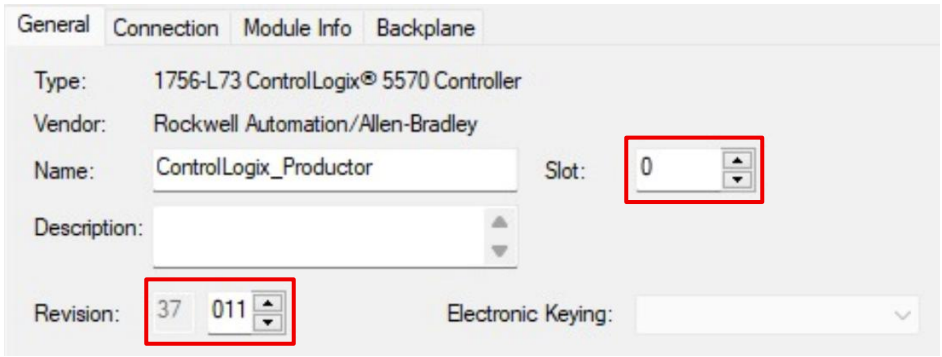
- Inicialmente, se debe crear un proyecto en Studio 5000 Logix Designer para el/los Consumidor/es, el cual es un CompactLogix con CPU modelo 1769-L33ERM. Considerar la Revisión de Firmware y Slot del Controlador.



- Debido que el/los Consumidor/es *necesitan la ruta de destino (Path) CIP del Productor de la Red*, se debe añadir dicho dispositivo desde la tarjeta de ethernet embebida del CompactLogix. Dar click en Ethernet, y seleccionar New Module. Buscar inicialmente el Módulo de Ethernet del ControlLogix (1756-EN2TR/C), y realizar correctamente su configuración.



- Luego de agregar el Módulo de Ethernet, aparecerá disponible el Backplane del ControlLogix. Dar click en el Backplane, y seleccionar New Module. Buscar el Controlador L73 para añadirlo al proyecto. Realizar correctamente su configuración.

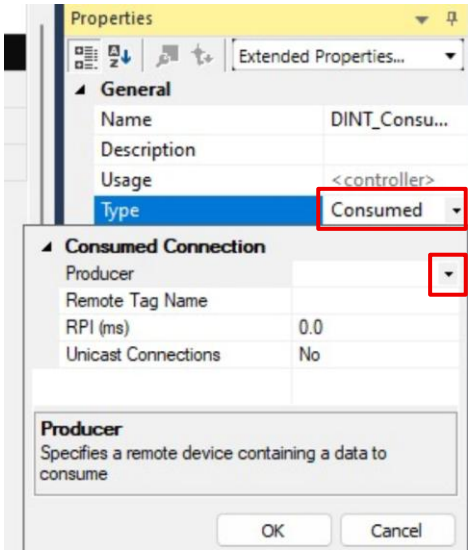


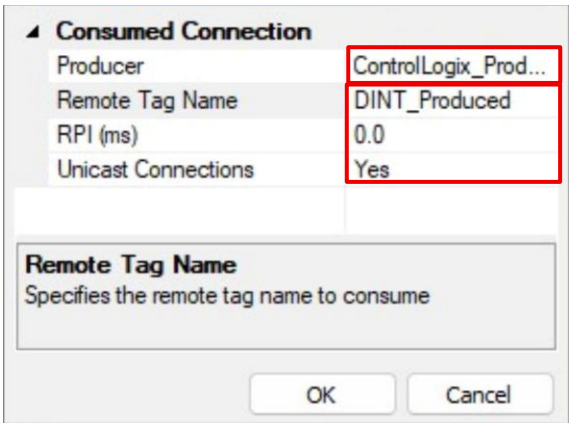
- Dirigirse a las Etiquetas del Controlador (Controller Tags), y crear una matriz de tipo DINT de 5 posiciones titulada “DINT_Consumed”.

Name	Alias For	Base Tag	Data Type
DINT_Consumed			DINT[5]

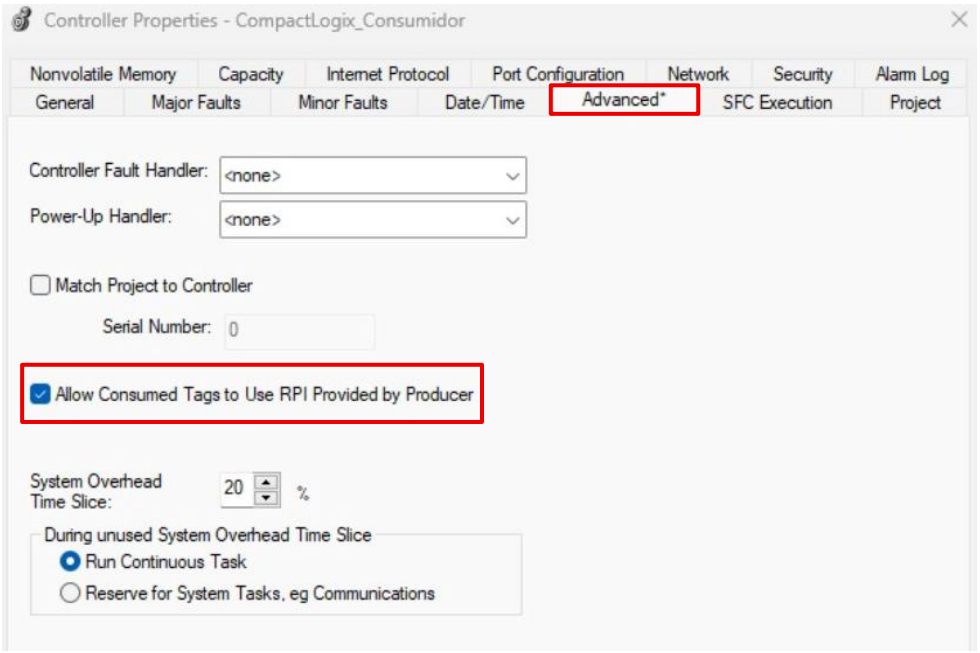
Nota: Para que la etiqueta DINT_Consumed sea la consumida de DINT_Produced ambas deben de tener *el mismo Data Type y ser Etiquetas Globales*.

- Dirigirse a las Propiedades de la etiqueta creada, seleccionar la propiedad *Type* y escoger el tipo **Consumed**. Aparecerá la pestaña de configuración *Consumed Connection*.
- En la opción *Producer* seleccionar el dispositivo Productor (ControlLogix_Productor). En *Remote Tag Name* escribir el nombre de la **variable producida que se desea consumir**, en este caso DINT_Produced. En *RPI* no colocar ningún valor, ya que este será asignado por el Productor de la Red (100 ms). La opción *Unicast Connections* automáticamente se pondrá en *Yes*.





- Dirigirse a las Propiedades del Controlador Consumidor (CompactLogix) y seleccionar la pestaña *Advanced*. Dar click en la opción *Allow Consumed Tags to Use RPI Provided by Producer*, de tal manera que ahora al dispositivo Consumidor se le asignará el RPI configurado del Productor (100 ms).

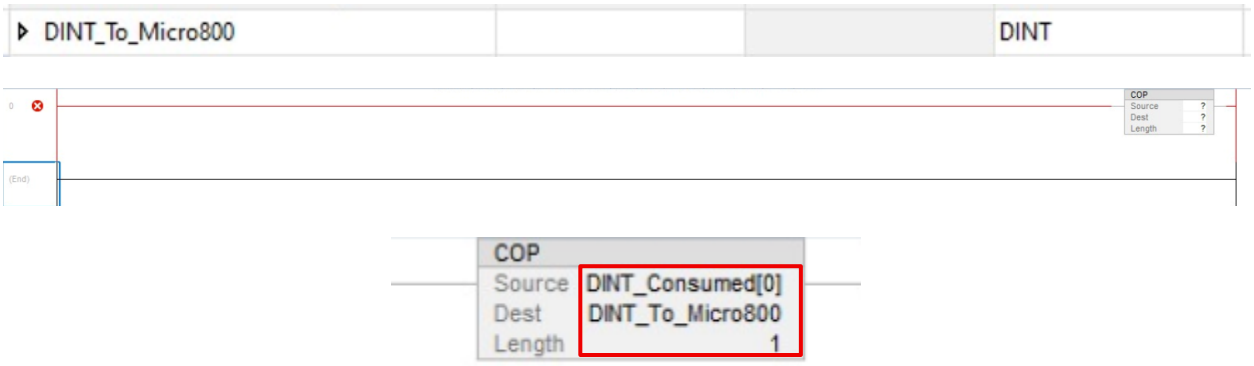


Nota: Si existen más consumidores para un mismo productor en la red, recordar que **todos deben tener el mismo valor de RPI**, si no *existirá desincronización en la comunicación*.

Nota: La razón por la cual se produce y se consume una matriz del tipo DINT, es porque *nos permite encapsular en una sola variable varios datos DINT*, y a su vez, *por cada DINT, varios valores BOOL*. En Modelo Productor/Consumidor es recomendable *transmitir datos en formato matriz/array para asegurar eficiencia en la comunicación*.

5.3. MSG_SYMBOLIC (WRITE)

- En esta sección se explicará cómo realizar la escritura mediante mensajería simbólica desde un CompactLogix a un Micro850. Para ello, antes de configurar la mensajería, inicialmente se debe definir **la fuente (source)** con el contenido a ser escrito en el **destinatario**. De la matriz DINT_Consumed se extraerá *el primer elemento* (DINT_Consumed[0]), y su contenido se copiará en una nueva etiqueta del tipo DINT llamada DINT_To_Micro800.
- Dirigirse a MainTask, luego a MainProgram y luego a MainRoutine. Agregar un nuevo renglón. Para realizar dicha operación de copiado binario utilizaremos la instrucción **COP**. En *Source* colocamos a DINT_Consumed[0], en *Dest* colocamos a DINT_To_Micro800 y en *Length* colocamos el valor de 1 (debido a que sólo se copiará un elemento de source a dest).



Nota: Recordar que el tipo de dato de Source (CompactLogix) debe corresponder con el tipo de dato de Destination (Micro800), o al menos ser equivalente. Un ejemplo de equivalencia es el siguiente (el cual se utilizará para el apartado de lectura de Micro a Control):

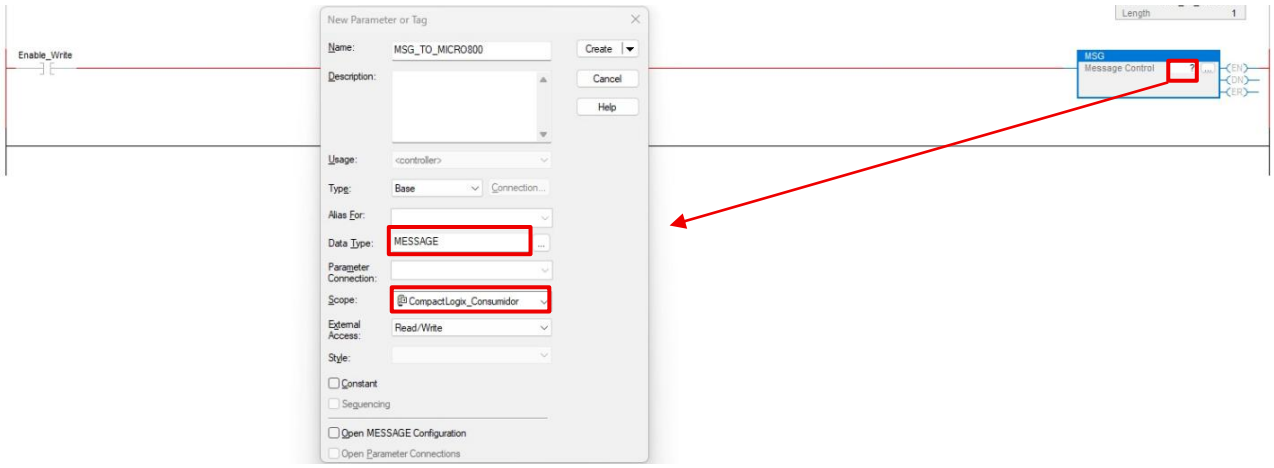
$$Unsigned\ Single\ Integer: 1\ USINT\ \vee\ 1\ SINT\ =\ 1\ Byte\ =\ 8\ bits$$

$$Double\ Integer: 1\ DINT\ =\ 4\ Bytes\ =\ 32\ bits$$

$$\therefore\ \mathbf{1\ DINT\ \equiv\ 4\ USINT\ \vee\ 4\ SINT}$$

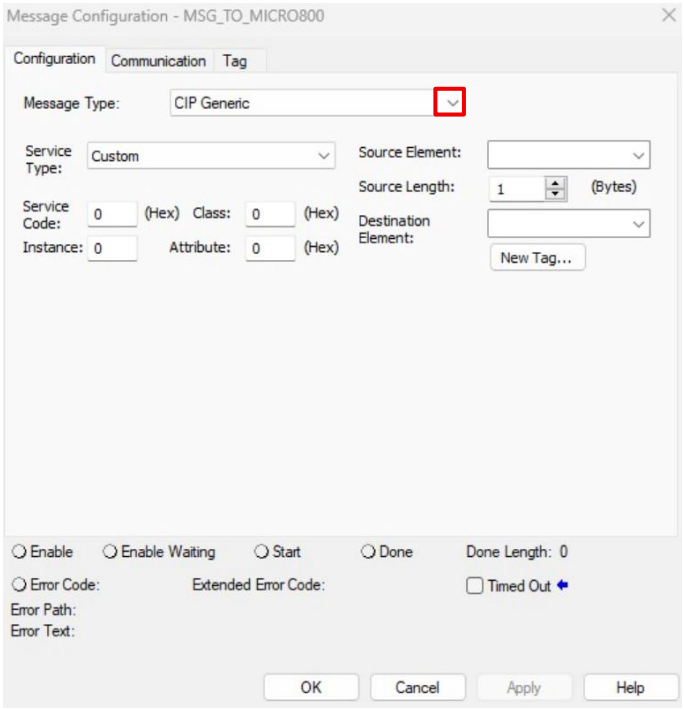
También considerar que tanto la escritura como lectura se logra para **etiquetas globales** o del **alcance del controlador**.

- A continuación añadir otro renglón a la rutina principal, donde se añadirá un permisivo NA (se encuentra en el apartado *Bit*) llamado Enable_Write y luego la instrucción **MSG** (se encuentra en el aparado de *Input/Output*). En *Message Control* crear una etiqueta de tipo MESSAGE llamada MSG_To_Micro800.

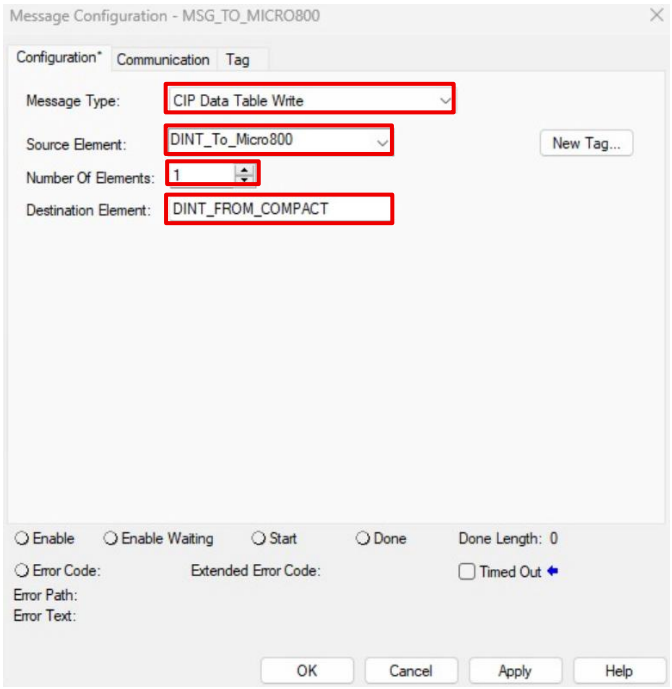


Nota: La etiqueta tipo MESSAGE debe ser una **etiqueta global**. Revisar que en *Scope* se encuentre seleccionado el controlador, el cual tendrá el nombre del proyecto.

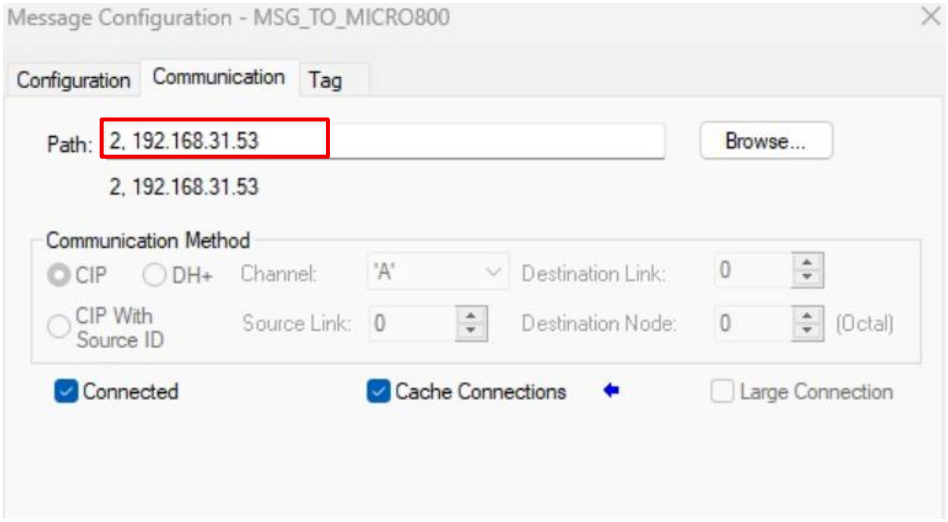
- Dar click en ... para configurar la instrucción de mensajería. Aparecerá por default el *Message Type* como *CIP Generic*. Debido a que se pretende realizar la escritura mediante mensajería simbólica, dar click en la lista de *Message Type* y seleccionar la opción *CIP Data Table Write*.



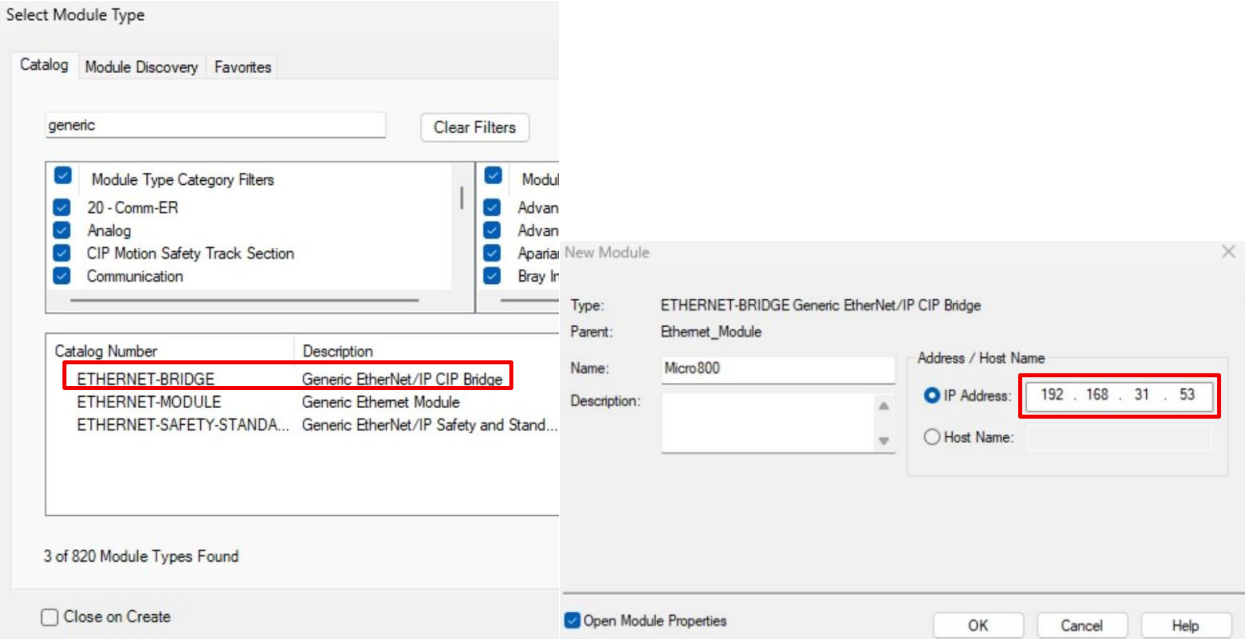
- Aparecerá la pestaña de configuración correspondiente. En *Source Element* colocar a DINT_To_Micro800. En *Number of Elements* colocar el valor de 1. En *Destination Element* colocar **el mismo nombre de la etiqueta remota** (la cual se encuentra definida como *etiqueta global en el Micro800*) llamada DINT_From_Compact.



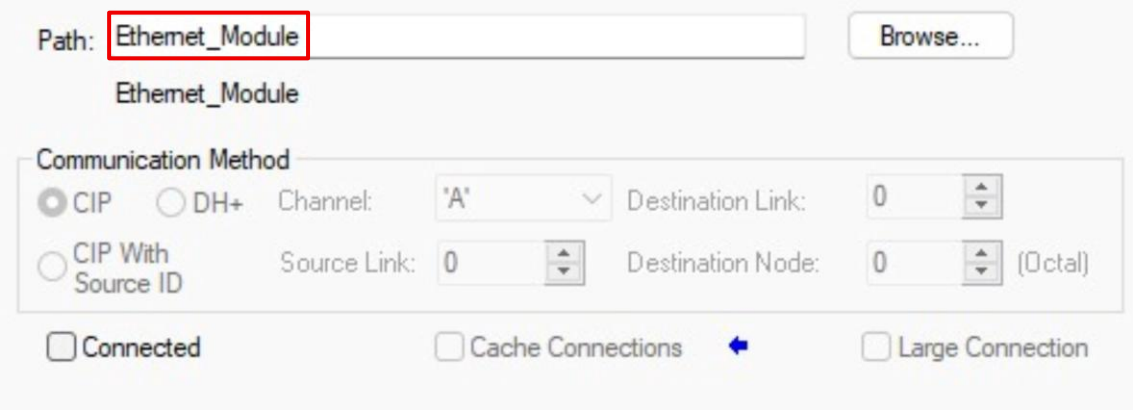
- Dirigirse a la pestaña de *Communication*. En *Path* escribir la ruta de destino CIP para realizar la comunicación de Compact a Micro (2 es el id del puerto ethernet embebido del CompactLogix, y lo siguiente es la dirección IP del Micro850).



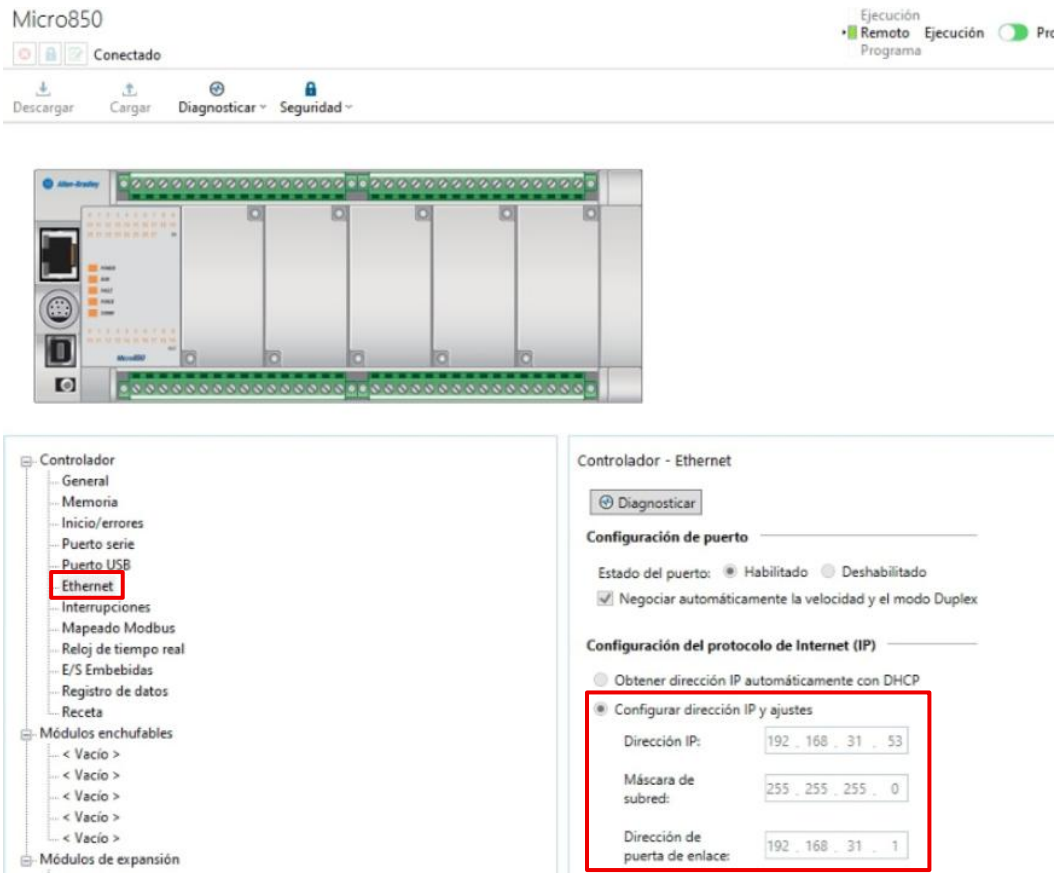
Nota: Otra alternativa para la Path es añadir un *Generic Ethernet-Bridge*, debido a que los Micro800 no se encuentran disponibles para añadir por Ethernet. La configuración del Bridge solo necesita la dirección IP del dispositivo de destino.



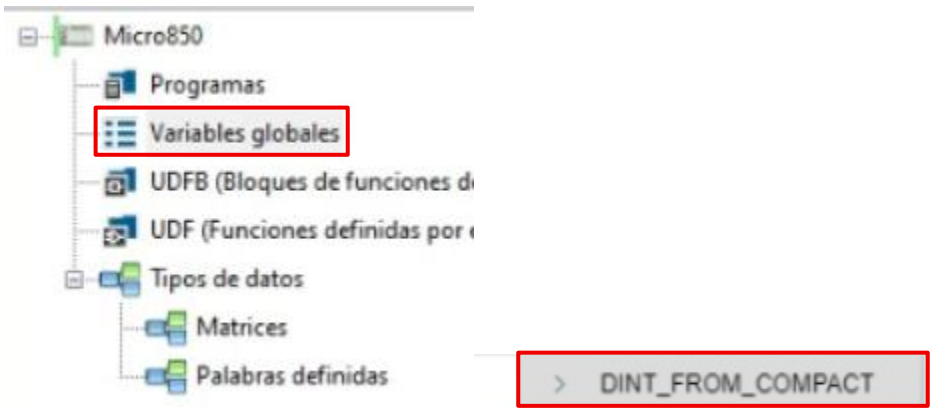
Luego en la Pestaña de Communication de la instrucción MSG dar click en *Browse*, y seleccionar el Ethernet_Module.



- Se debe crear un proyecto en Connected Components Workbench (CCW) para el Micro850. Considerar el Modelo Respectivo y la Revisión de Firmware. En la sección *Micro850* dirigirse a la pestaña *Ethernet* y realizar la configuración del direccionamiento IP.



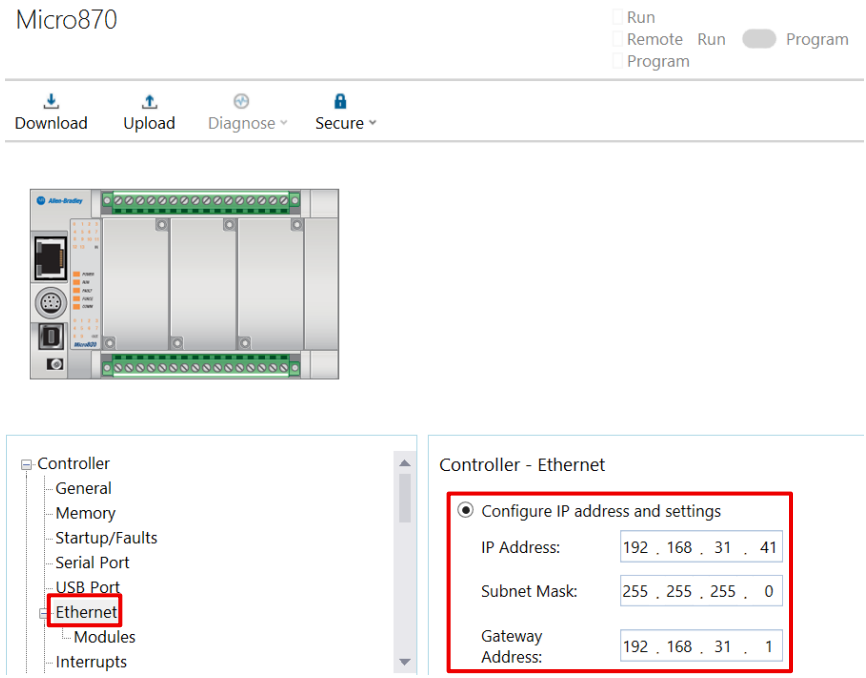
- En la sección *Variables Globales* crear la etiqueta de tipo DINT llamada DINT_From_Compact.



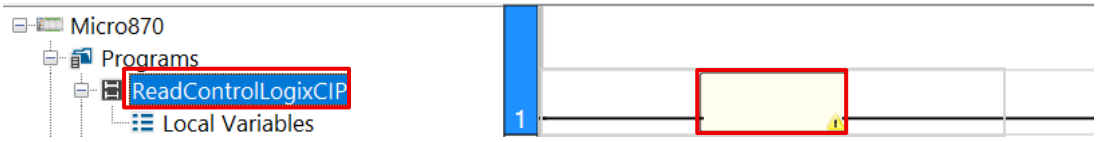
- Habilitar la instrucción MSG mediante Enable_Write y comprobar que la bandera DN se active y no exista ningún error. Corroborar monitoreando la variable DINT_From_Compact.

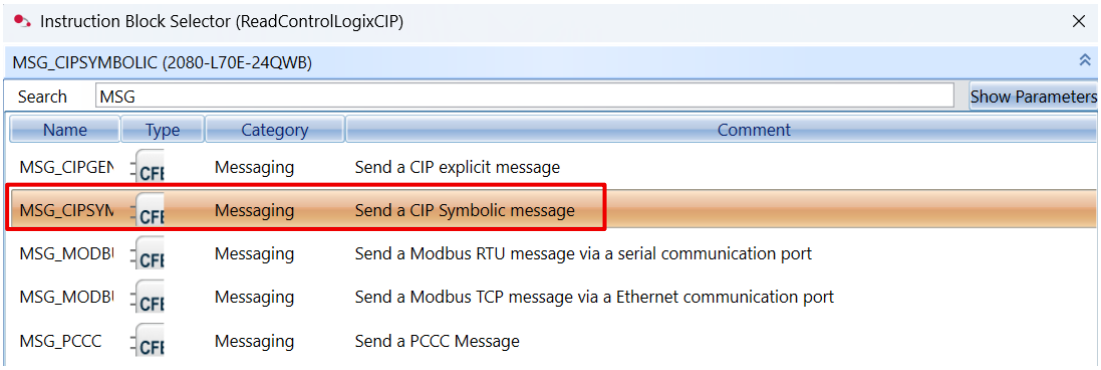
5.4. MSG_SYMBOLIC (READ)

- En esta sección se explicará cómo realizar la lectura mediante mensajería simbólica desde un Micro870 a un ControlLogix. Inicialmente se debe crear un proyecto en Studio 5000 Logix Designer y crear una etiqueta global de tipo DINT llamada DintToMicro870.
- Luego se debe crear un proyecto en Connected Components Workbench (CCW) para el Micro870. Considerar el Modelo Respectivo y la Revisión de Firmware. En la sección *Micro870* dirigirse a la pestaña *Ethernet* y realizar la configuración del direccionamiento IP.

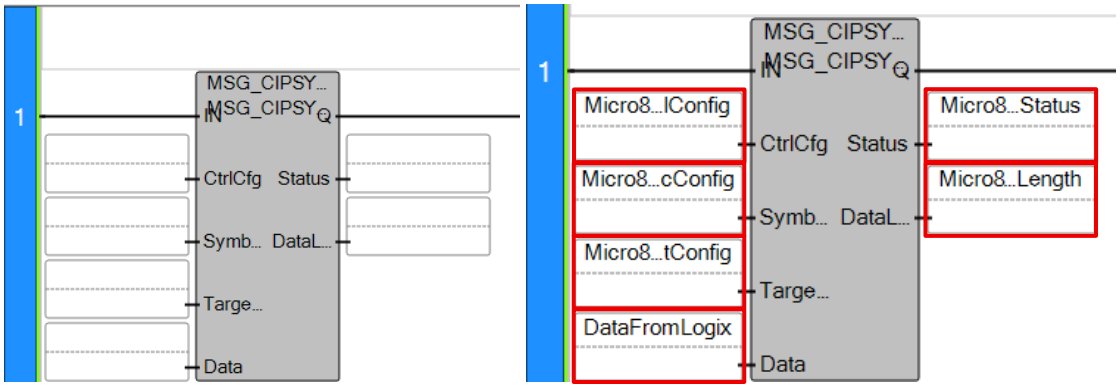


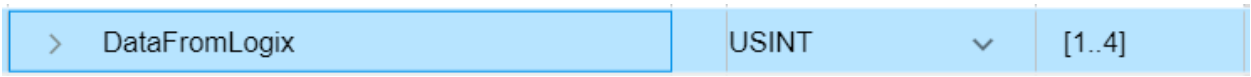
- Dar click derecho en la sección *Programs*, seleccionar *Add New LD* (Ladder Diagram) y darle un nombre. Agregar un renglón y un bloque de instrucción. Dar click en el bloque genérico, buscar la instrucción MSG_CIPSYMBOLIC y seleccionarla.





- Crear los parámetros de la instrucción MSG_CIPSYMBOLIC. El parámetro *Data* debe ser una matriz tipo USINT. Dado que se pretende leer un dato tipo DINT, la etiqueta global DataFromLogix será una matriz USINT[1..4].





- Realizar la configuración del parámetro CIPCONTROLCFG. Para el atributo *TriggerType* escribir en *Initial Value* el valor de 100 (la mensajería se ejecutará periódicamente cada 100 ms).

Micro800ControlConfig	CIPCONTROLCFG	...	...	
Micro800ControlConfig.Cancel	BOOL			Abort the execution of message
Micro800ControlConfig.TriggerType	UDINT		100	0 - Trigger once, n - Cyclic trigger
Micro800ControlConfig.StrMode	USINT			reserved parameter

- Realizar la configuración del parámetro CIPSYMBOLICCFG. Para el atributo *Service* dejarlo por default 0 (Read). Para el atributo *Symbol* escribir en *Initial Value* el nombre de la etiqueta a leer DintToMicro870. Para el atributo *Count* dejar por default 0 (se leerá un elemento). El atributo *DataType* no es necesario especificarlo ya que no se realizará un servicio de escritura.

Micro800SymbolicConfig	CIPSYMBOLICCFG	...	...	
Micro800SymbolicConfig.Service	USINT			0 - Read, 1 - Write
Micro800SymbolicConfig.Symbol	STRING		'DintToMicro870'	Symbol name to read / write
Micro800SymbolicConfig.Count	UINT			Num of variables to read/ write.
Micro800SymbolicConfig.DataType	USINT			Symbol data type
Micro800SymbolicConfig.Offset	USINT			Byte offset of variable to read / w

- Realizar la configuración del parámetro CIPTARGETCFG. Para el atributo *Path* escribir la ruta de destino CIP de un Micro800 a un ControlLogix (4 es el id del puerto ethernet embebido en el Micro800, 192.168.31.111 es la dirección IP del módulo ENET, 1 es el id del backplane, 0 es el slot del controlador L73). Para el atributo *CipConnMode* dejar el valor de default de 0.

Micro800TargetConfig	CIPTARGE...	...	...	
Micro800TargetConfig.Path	STRING		'4,192.168.31.111,1,0'	CIP destination path
Micro800TargetConfig.CipConnMode	USINT			0 - Unconnected, 1 - Class3 connection
Micro800TargetConfig.UcmmTimeout	UDINT			Unconnected message time out.
Micro800TargetConfig.ConnMsgTi...	UDINT			Connected message time out.
Micro800TargetConfig.ConnClose	BOOL			TRUE: Close CIP connection upon messa

- Añadir un nuevo renglón con la instrucción COP para copiar el contenido binario de DataFromLogix (USINT[1..4]) a DintFromLogix (UDINT[1..1]). Los *Offsets* respectivos dejarlos en 0 y *Length* dejarlo en 1 (la longitud de DinFromLogix es de 1 elemento). Activar el parámetro *Swap* con un valor de True debido que *durante la comunicación ocurrirá un cambio en el orden del contenido binario* (**Endianness**). Adicionalmente añadir un *Direct Contact* con la bandera Q de la instrucción MSG_CIPSYMBOLIC.

DintFromLogix	UDINT	[1..1]

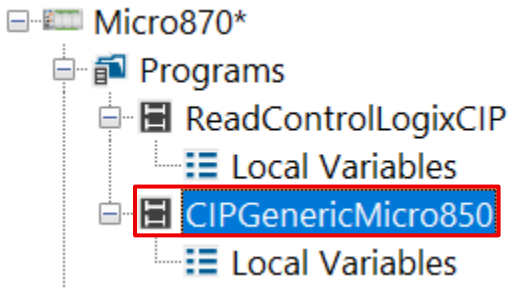
- Monitorear la instrucción MSG_CIPSYMBOLIC si no existe algún error. Comprobar que el contenido de DintToMicro870 se haya copiado correctamente en DintFromLogix.

Nota: Otra razón por la cual se utiliza la instrucción COP es debido a que la variable DataFromLogix **no es estática**, ya que *en ciertos intervalos de la periodicidad* el valor será **0**. Luego de realizar el copiado del contenido binario, el dato en DintFromLogix será fijo.

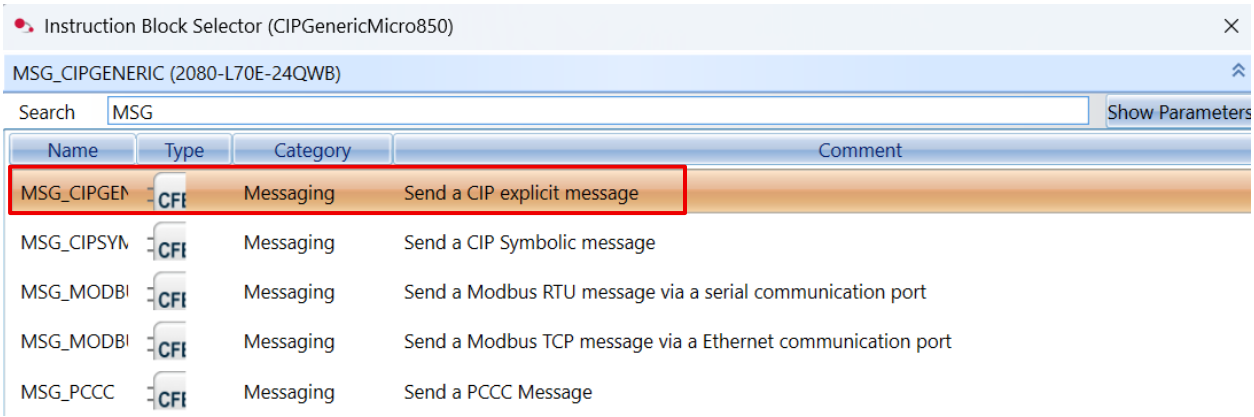
Recomendación: Dado que ya se explicaron los servicios de lectura y escritura mediante mensajería implícita (simbólica), uno se preguntaría *cuál se debería utilizar en un contexto de comunicación cíclica* (Clase de Conexión 1 del Protocolo CIP). La respuesta por criterio de seguridad es **lectura**. Esto debido a que *tengo información de dónde proviene la data de interés*. En el caso de la escritura, el dispositivo que ha sido escrito *no tiene información de dónde provino esa data*, lo cual podría dar pie a un problema de seguridad.

5.5. MSG_GENERIC

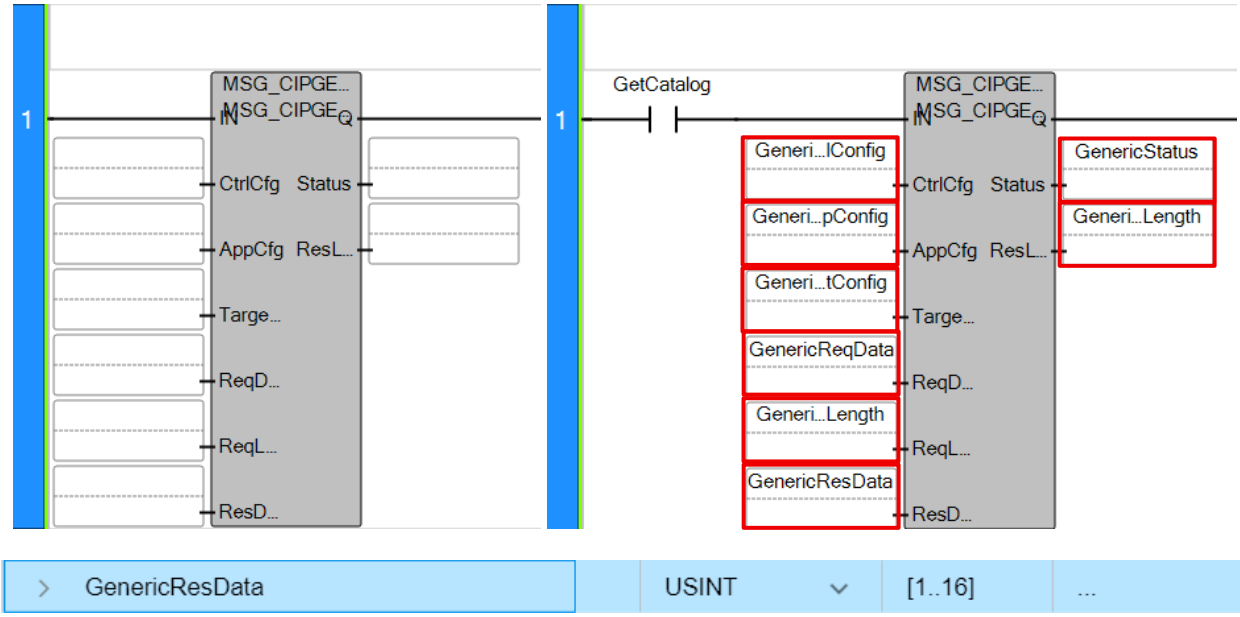
- En esta sección se explicará cómo realizar la lectura mediante mensajería genérica desde un Micro870 a un Micro850, de tal manera que se obtenga el *Product Name* de este último. En el mismo proyecto crear un nuevo *Ladder Diagram*.



- Agregar un renglón y un bloque de instrucción. Dar click en el bloque genérico, buscar la instrucción MSG_CIPGENERIC y seleccionarla.



- Crear los parámetros de la instrucción MSG_CIPGENERIC. El parámetro *ResData* debe ser una matriz de tipo USINT. Dado que se pretende leer "2080-LC50-48QWB" que es un dato tipo STRING de 16 caracteres (16 contando el Null Terminator), la etiqueta global GenericResData será una matriz USINT[1..16]. Adicionalmente se añade un *Direct Contact* llamado GetCatalog para habilitar la mensajería.



GenericResData	USINT	[1..16]	...

- Realizar la configuración del parámetro CIPCONTROLCFG. Para el atributo *TriggerType* dejar por default el valor de 0 (se ejecutará sólo una vez la instrucción). Esto debido que *Product Name* es un dato estático (propiedad del dispositivo).

GenericControlConfig	CIPCONTR...	...	...	
GenericControlConfig.Cancel	BOOL			Abort the execution of message
GenericControlConfig.TriggerType	UDINT		0	0 - Trigger once, n - Cyclic trigger
GenericControlConfig.StrMode	USINT			reserved parameter

- Realizar la configuración del parámetro CIPAPPCFG (Revisar el apartado de Objetos CIP para Micro800). Para el atributo *Service* escribir el valor 14 (Get_Attribute_Single). Para el atributo *Class* escribir el valor 01 (Código de Clase). Para el atributo *Instance* escribir el valor 01 (ID). Para el atributo *Attribute* escribir el valor 07 (Product_Name).

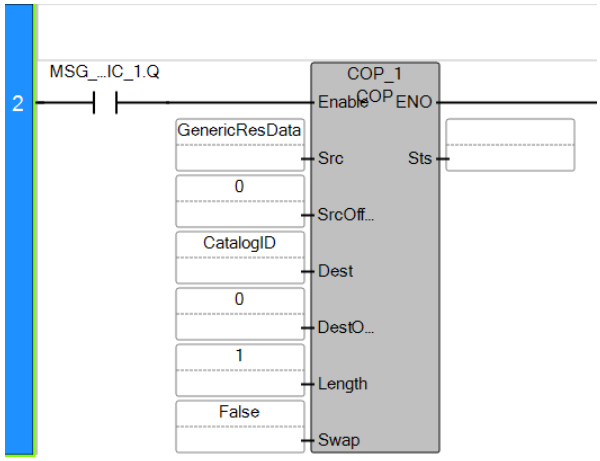
GenericAppConfig	CIPAPPCFG	...	...	
GenericAppConfig.Service	USINT		14	CIP Service code: 1 - 127
GenericAppConfig.Class	UINT		01	CIP Class ID: 1 - 65535
GenericAppConfig.Instance	UDINT		01	CIP Instance ID: 0 - 0xFFFFFFFF
GenericAppConfig.Attribute	UINT		07	CIP Attribute: 1 - 65535, 0 - No attribute
GenericAppConfig.MemberCnt	USINT			CIP Member ID count: 1 - 3, 0 - None
GenericAppConfig.MemberId	CIPMEMBERID	...	...	CIP Member ID: 0 - 65535

- Realizar la configuración del parámetro CIPTARGETCFG. Para el atributo *Path* escribir la ruta de destino CIP entre Micro800 (4 es el id del puerto ethernet embebido en el Micro870, 192.168.31.53 es la dirección IP del Micro850). Para el atributo *CipConnMode* escribir el valor de 1, debido que la comunicación explícita es una Clase de Conexión 3 del Protocolo CIP.

GenericTargetConfig	CIPTARGE...	...	...	
GenericTargetConfig.Path	STRING		'4,192.168.31.53'	CIP destination path
GenericTargetConfig.CipConnMode	USINT		1	0 - Unconnected, 1 - Class3 connection
GenericTargetConfig.UcmmTimeout	UDINT			Unconnected message time out.
GenericTargetConfig.ConnMsgTim...	UDINT			Connected message time out.
GenericTargetConfig.ConnClose	BOOL			TRUE: Close CIP connection upon message ...

Nota: Por lo general las comunicaciones explícitas/genéricas (Clase 3) se utilizan para configuración, diagnóstico o lectura de parámetros de forma *no periódica* (no es crítica con respecto al tiempo).

- Añadir un nuevo renglón con la instrucción COP para copiar el contenido binario de GenericResData (USINT[1..16]) a CatalogID (STRING[1..1]). Los *Offsets* respectivos dejarlos en 0 y *Length* dejarlo en 1 (la longitud de CatalogID es de 1 elemento). Para este caso no se necesita activar el parámetro *Swap*. Adicionalmente añadir un *Direct Contact* con la bandera Q de la instrucción MSG_CIPGENERIC.



CatalogID	STRING	[1..1]

- Habilitar la instrucción MSG_CIPGENERIC mediante GetCatalog y comprobar que no exista ningún error en *Status*. Corroborar con el dato obtenido en CatalogID.

References

[1] ODVA, «Protocolo Industrial Común (CIP™),» [En línea]. Available: https://www.odva.org/technology-standards/key-technologies/common-industrial-protocol-cip/.