ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

LABORATORIO DE COMUNICACIONES INDUSTRIALES Y SISTEMAS SCADA

Práctica

Comunicación Modbus RTU entre Micro800 y PowerFlex 4M

Autor:

Arturo Freire Veliz

GUAYAQUIL – ECUADOR

I PAO 2025

1. Objetivos

1.1. Objetivo general

Establecer una comunicación eficiente y funcional entre el variador de frecuencia PowerFlex 4M y el PLC Micro870 mediante el protocolo Modbus RTU utilizando Connected Components Workbench (CCW), permitiendo la operación y monitoreo remoto a través de una interfaz HMI.

1.2. Objetivos específicos

1. Configurar el variador PowerFlex 4M para la comunicación mediante Modbus RTU, ajustando los parámetros necesarios según su manual técnico, y definir las tramas requeridas para la lectura y escritura de registros.
2. Desarrollar un programa en CCW para el PLC Micro870 que implemente bloques de mensajería Modbus RTU, permitiendo la lectura y escritura de registros del variador para operaciones como encendido, apagado, cambio de dirección y ajuste de frecuencias.
3. Integrar la comunicación con una HMI en Connected Components Workbench (CCW), diseñando una interfaz gráfica que facilite la manipulación de los comandos y monitoreo del estado del variador.

2. Equipos y herramientas

- Connected Components Workbench (CCW).
- RsLinx Classic.
- Cables Ethernet.
- Stratix 2100.
- PC.
- PanelView 800 2711R-T7T.
- Cable RJ45 TIA-568B.
- Terminal RJ45 to RS485.
- Resistencias de Terminación de 120 Ohms.
- Allen Bradley USB-1203.
- Drive Powerflex 4M.
- Micro870 2080-L70E-24QWB.
- Módulo de Comunicación Serial 2080-SERIALISOL.

3. Marco teórico

3.1. Modelo Maestro-Esclavo

Es una aplicación particular de la arquitectura de comunicación cliente/servidor en que el Maestro, es decir, el cliente, *requiere y envía información hacia un Esclavo*, es decir, un servidor. Por otra parte, el Esclavo, *se limita a enviar la información solicitada y a efectuar las acciones de control dadas por el Maestro*.

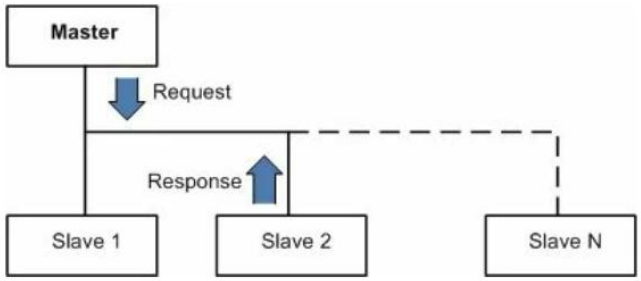


Figura 1. Modelo Maestro-Esclavo.

3.2. Estándar RS-485

RS-485 (actualmente conocido como EIA/TIA-485) es una interfaz estándar de la capa física de comunicación, un método de transmisión de señales, el *1er nivel del modelo OSI* (Interconexión de Sistemas Abiertos). El estándar RS-485 fue creado para ampliar las capacidades físicas de la interfaz RS-232.

La conexión serie EIA-485 es realizada utilizando un cable de *dos o tres hilos*: un hilo de datos, un hilo con datos invertidos y, a menudo, un hilo cero (tierra, 0 V). De este modo, los transmisores y los receptores intercambian los datos a través de un cable de par trenzado de hilos rígidos de 22 o 24 AWG.

Su principal función es transportar una señal a través de dos cables. Uno de los cables transmite la señal original y el otro transporta su copia inversa. Este método de transmisión *ofrece una gran resistencia a las interferencias en modo común*. El cable de par trenzado utilizado como línea de transmisión puede ser blindado o no blindado.

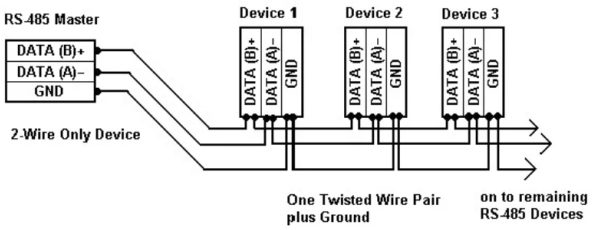


Figura 2. Conexión RS-485.

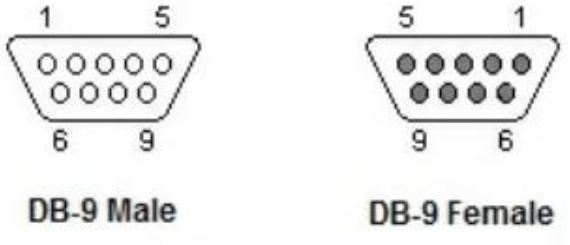
Debido a la transmisión diferencial de la señal, siempre hay una divergencia de potencial entre los cables. Esto garantiza una alta resistencia a las interferencias en modo común. Además, el par trenzado puede ser blindado, lo que garantiza la protección de los datos transmitidos. Todo esto permite enviar datos a largas distancias y a velocidades relativamente altas, que pueden alcanzar 100 kbits/s a 4000 pies.

La longitud máxima del cable utilizado en las comunicaciones RS-485 es de 4000 pies o unos 1200 metros. No obstante, se recomienda que el producto de la longitud de la línea (en metros) y la velocidad de datos (en bits por segundo) no sea superior a 10^8. Por ejemplo, un cable de 20 metros permite una velocidad de datos máxima de 5 Mbits/s.

RS-485 de *2 contactos* utiliza el modo de trabajo semidúplex (Half-Duplex) y solo un punto puede estar en estado de envío en cualquier momento, por lo que el circuito de envío debe ser controlado por la señal de habilitación. Por lo tanto, los dispositivos RS-485 no pueden transmitir y recibir datos al mismo tiempo, ya que se produce un conflicto de transmisores (colisión).

3.2.1. Pinout Conector RS-485

El pinout del conector DB9 para RS-485 se muestra a continuación:



DB-9	Designation	Name
1		Common Ground
2	CTS+	Clear To Send +
3	RTS+	Ready To Send +
4	RxD+	Received Data +
5	RxD-	Received Data -
6	CTS-	Clear To Send -
7	RTS-	Ready To Send -
8	TxD+	Transmitted Data +
9	TxD-	Transmitted Data -

Figura 3. Pinout RS-485.

A continuación, las definiciones de línea de señal RS-485:

- **Detector de Portador (CD)**: Esta señal de control se usa cuando un módem informa a un ordenador que ha detectado un operador que el ordenador puede usar para la transmisión de datos.
- **Recibir Datos (RXD)**: Esta línea se utiliza para la transmisión de datos entre dos fuentes. Un ejemplo son los datos recibidos de un módem transferidos a un ordenador.
- **Transmitir Datos (TXD):** Esta es la línea que realmente transporta los datos transmitidos.
- **Terminal de Datos Preparado (DTR):** Esta es la señal que muestra que un ordenador está listo para la transmisión.
- **Tierra del Sistema (GND):** Se refiere a una conexión física con la tierra, una línea base utilizada para medir voltajes en un circuito eléctrico o una ruta compartida para la corriente eléctrica de retorno.
- **Conjunto de Datos Preparado (DSR)**: A diferencia de la señal DTR, esta señal notifica a un ordenador o terminal que el módem está operativo y es capaz de recibir datos.
- **Solicitud de Envío (RTS)**: Es necesario un voltaje positivo para que esta señal permita que se realice la solicitud para enviar (RTS). Indica que es posible una transmisión sin interferencias entre el conjunto de datos y el terminal de datos.
- **Libre para Envío (CTS)**: Enviar esta señal después de que se haya establecido una conexión entre un terminal de datos y un módem confirma que el terminal de datos reconoce que las comunicaciones pueden comenzar.
- **Indicador de Llamada (RI)**: El propósito de esta señal es alertar a un módem que opera un conjunto de datos que se ha detectado una baja frecuencia. La señal simplemente alerta al terminal de datos pero no afecta la transmisión de datos entre los dispositivos.

El pinout se conecta a los conectores DB9 y DB25 con 2 y 4 contactos:

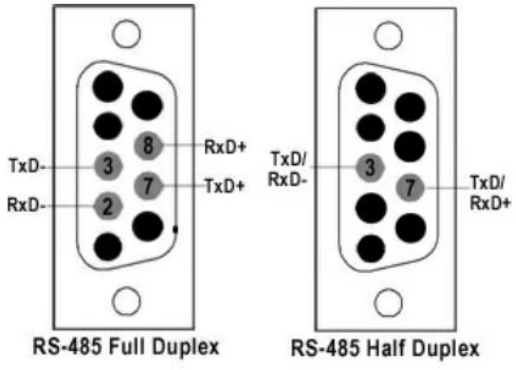


Figura 4. Pinout RS-485 Conector DB9.

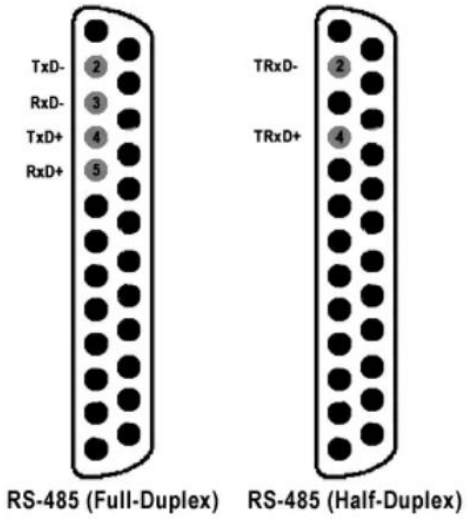


Figura 5. Pinout RS-485 Conector DB25.

Las líneas TxD+ y TxD- transportan los datos de transmisión, mientras que RxD+ y RxD- contienen los datos de recepción. Las distancias que llevan estas señales son mayores debido a las señales diferenciales.

Como se muestra en el pinout del cable RS485, la interfaz tiene todas las señales en configuraciones diferenciales:

- Las señales CTS+ y CTS- y RTS+ and RTS- son empleadas como señales de control de handshake.
- TxD+ y TxD- realizan transmisión de datos.
- RxD+ and RxD- realizan recolección de datos.

3.2.2. Descripción Interfaz RS-485

Tipos

Hay dos tipos de RS485:

- RS485 en modo semidúplex (Half-Duplex) con 2 contactos.
- RS485 en modo duplex completo (Full-Duplex) con 4 contactos.

El modo dúplex completo se usa cuando se necesita poder transmitir y recibir datos al mismo tiempo. En el modo semidúplex, solo se puede transmitir o recibir datos en un momento dado.

No se utiliza ningún tipo específico de conector para implementar el protocolo RS485, pero en la mayoría de los escenarios, se emplea un conector DB9 o un bloque de terminales.

Los conectores RS485 específicos pueden tener diferentes pines. Puede determinar la configuración real en función de la documentación que acompaña al dispositivo.

Conectar dispositivos RS485 con 2 contactos:

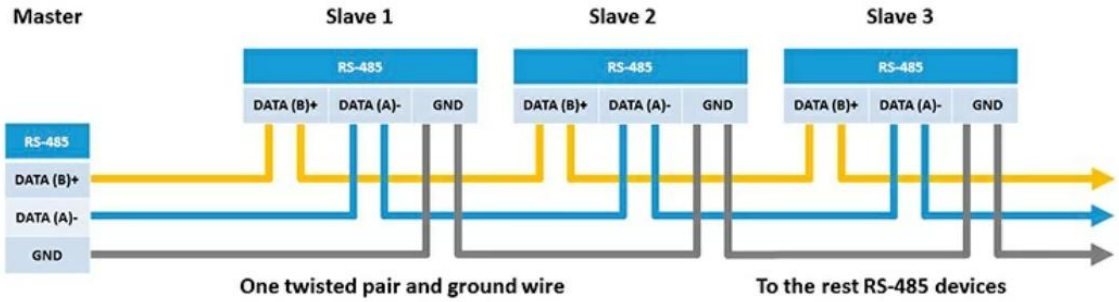


Figura 6. Conexión RS-485 de 2 contactos.

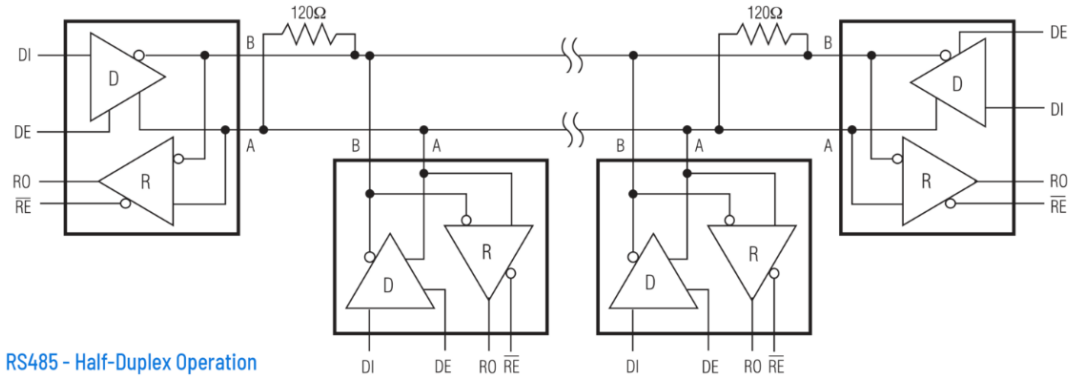


Figura 7. Conexión RS-485 Half-Duplex.

Conectar dispositivos RS485 con 4 contactos:

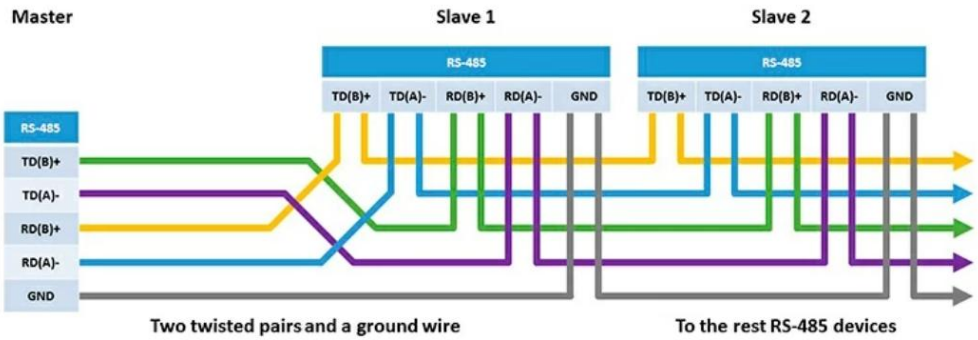


Figura 8. Conexión RS-485 de 4 contactos.

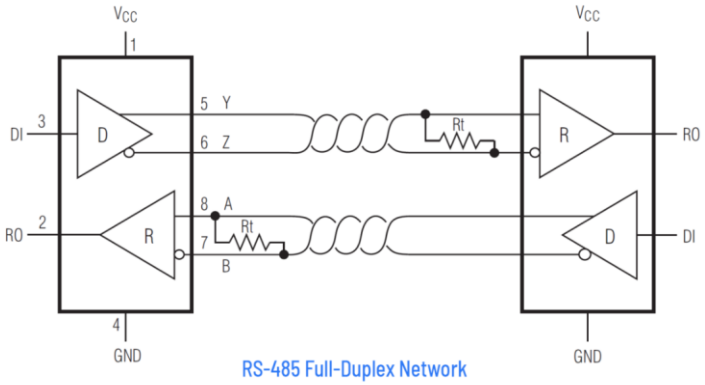


Figura 9. Conexión RS-485 Full-Duplex.

Transceptor

Cada dispositivo conectado a un bus RS-485 puede actuar como un controlador (transmisor) o un receptor . El circuito que controla o lee la línea es en realidad un amplificador diferencial de amplificador operacional. Es por eso que los controladores se representan mediante un triángulo con dos entradas y una salida. El controlador es un dispositivo que actualmente utiliza el bus para transmitir datos. Aunque puede haber 32 controladores en una línea, solo uno puede estar activo a la vez. Esto significa que solo un dispositivo puede transmitir a través de un par de líneas RS-485. Todos los demás dispositivos en la red actuarán como receptores en este momento y pueden leer todos los datos que pasan por el bus. Este tipo de línea de comunicación se llama línea multipunto.

RS-485 también puede considerarse una línea multipunto, ya que cualquier dispositivo también puede actuar como controlador/transmisor cambiando su función dinámicamente.

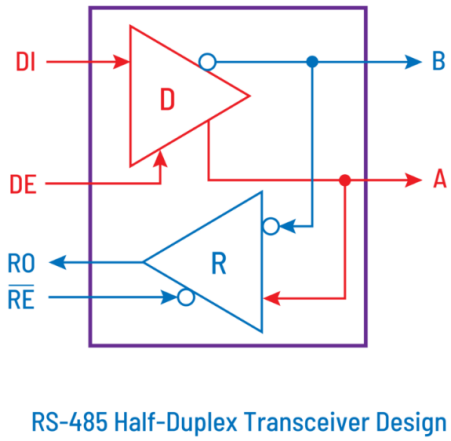


Figura 10. Transceptor Half-Duplex.

Voltaje de Modo Común (VCM)

El voltaje VCM (Voltaje de Modo Común) es la suma de las diferencias de potencial de GND entre los participantes RS-485, el voltaje de offset del controlador y el ruido de modo común (V-ruido) que actúa sobre la línea de bus. Los fabricantes de controladores RS-485 proporcionan un rango de voltaje para VCM de -7 a 12 V. En caso de problemas de comunicación, este rango de voltaje, resultante de las diferencias de potencial entre el transmisor y el receptor, se ve frecuentemente afectado si la interfaz no está separada galvánicamente por configuración o no existe una línea común. La Imagen 6 muestra el cálculo del voltaje de modo común.

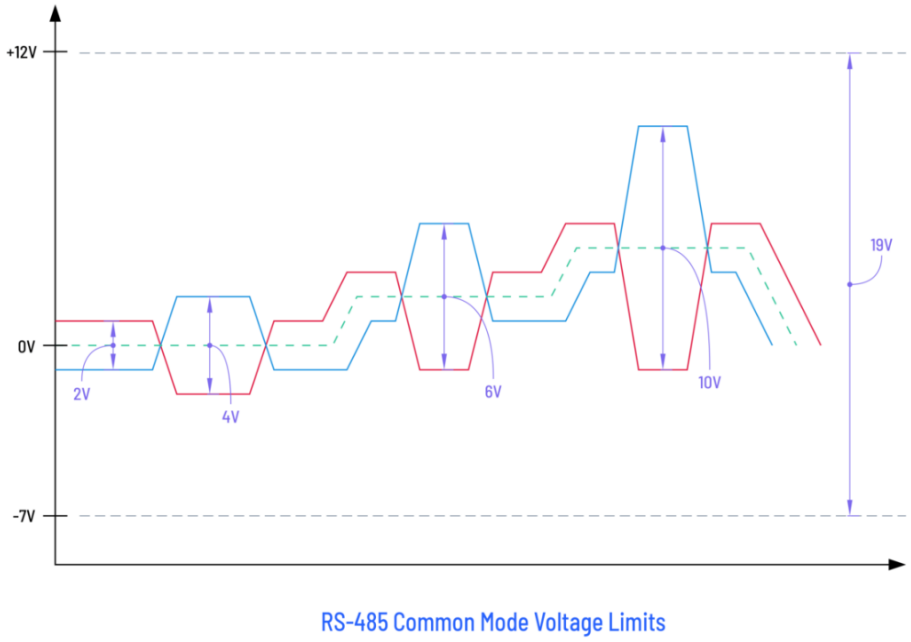
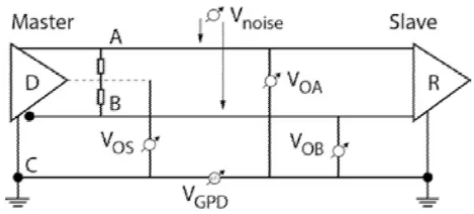


Figura 11. Rango de Voltaje VCM.



$$V_{OS} = \frac{V_{OA} + V_{OB}}{2}$$

$$V_{CM} = V_{OS} + V_{noise} + V_{GPD}$$

Figura 12. Cálculo del voltaje de modo común.

Topología de Bus

El bus es multipunto y permite conectar hasta 32 participantes sin repetidor. La mejor topología de red es la conexión en cadena. Esto significa que el cable del bus se conecta directamente de esclavo a esclavo. Es importante tener en cuenta que, en general, se deben evitar las derivaciones. Estas derivaciones causan reflexiones en el bus. En teoría, es posible calcular una posible derivación según el transceptor utilizado. Sin embargo, en la práctica, esto resulta complejo.

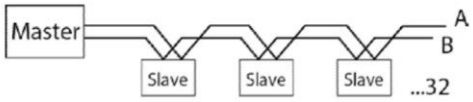


Figura 13. Topología de Bus.

Terminación

Otra causa de interrupciones de comunicación son las *reflexiones del bus*. Una reflexión se produce si la señal del transmisor no ha sido absorbida completamente por la carga. La impedancia de la fuente *debe reflejar la impedancia de carga y la impedancia de sobretensión de la línea*, ya que de esta manera se alcanza la potencia máxima de la señal y solo se producen reflexiones mínimas.

La comunicación serie de la interfaz RS-485 funciona de forma más eficiente cuando la impedancia de la fuente y la de carga están armonizadas a **120 ohmios**. Por esta razón, el estándar RS-485 recomienda una línea de bus con una impedancia de sobretensión de línea de Z0 = 120 ohmios. Para evitar reflexiones en el bus, esta debe estar equipada con una resistencia de terminación al inicio y al final, que debe reflejar la impedancia de sobretensión de la línea.

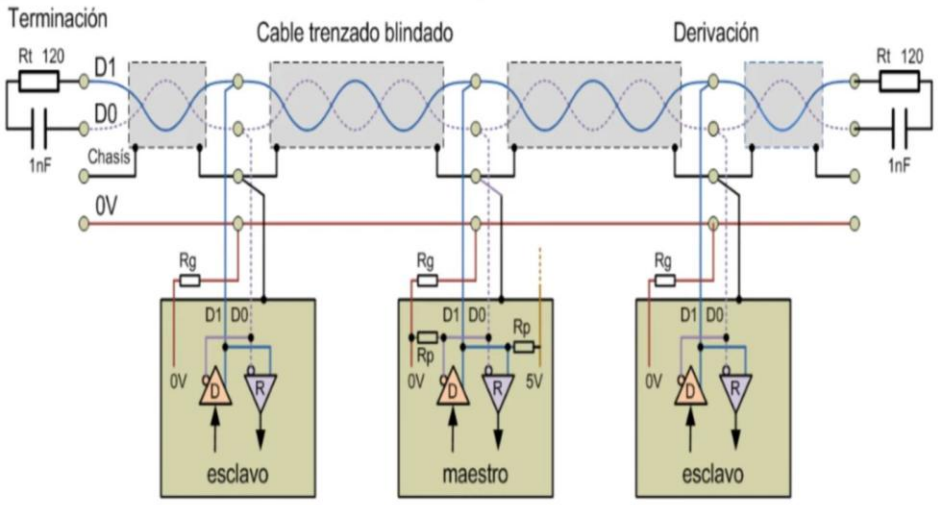


Figura 14. Terminanación con Filtrado Adicional.

Funcionamiento de la Interfaz

Los datos durante la transmisión 485 tienen tres estados:

1. Cuando la diferencia de voltaje entre A y B VAB = UA-UB es mayor que +200 mV, la salida lógica del transceptor 485 es 1.
2. Cuando la diferencia de voltaje entre A y B VAB = UA-UB es inferior a -200 mV, la lógica de salida del transceptor 485 es 0.
3. Cuando la diferencia de voltaje entre A y B VAB = UA-UB está entre -200 mV ~ +200 mV, el transceptor 485 puede generar un nivel alto o bajo, lo cual es un estado incierto.

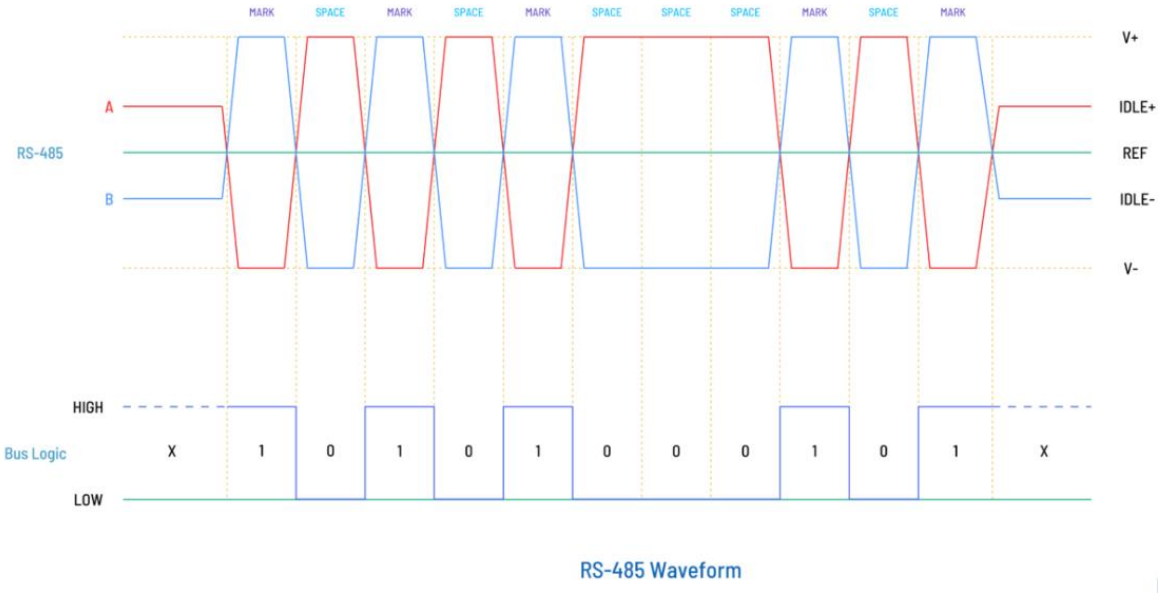


Figura 15. Forma de la Señal RS-485.

3.3. Protocolo Modbus RTU

Modbus RTU opera mediante interfaces seriales RS-232 y RS-485. RS-232 admite la comunicación punto a punto en distancias cortas, ideal para configuraciones básicas. RS-485, por otro lado, permite la conexión en red multipunto, lo que permite el uso de múltiples dispositivos a largas distancias con mayor inmunidad al ruido.

Figura 16. Protocolo Modbus RTU.

3.4. Trama de Datos Modbus RTU

Una trama Modbus RTU consta de varios componentes que permiten una transmisión de datos eficiente y precisa. La estructura de la trama incluye los siguientes elementos:

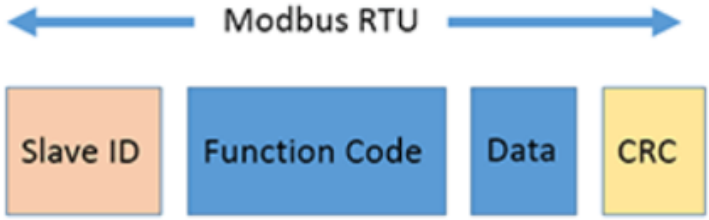


Figura 17. Trama Modbus RTU.

El marco Modbus RTU consta de cuatro componentes principales:

1. **Campo de Dirección (1 byte)**: Identifica el dispositivo esclavo de destino. El valor oscila entre 1 y 247, con 0 reservado para la transmisión.
2. **Código de Función (1 byte)**: Indica el tipo de acción que solicita el maestro, como leer o escribir datos.
3. **Campo de Datos (N bytes)**: Contiene la información relevante para la función, incluidas las direcciones y valores de los registros.
4. **Campo CRC (2 bytes)**: Proporciona verificación de errores mediante una verificación de redundancia cíclica.

Modbus RTU Frame Format

Start	Address	Function	Data	CRC	End
≥3.5 char	8 bit	8 bit	N * 8 bit	16 bits	≥3.5 char

Modbus ASCII Frame Format

Start	Address	Function	Data	LRC	End
;	2chars	2chars	N * 1 chars	2chars	CR, LF

Figura 18. Comparación entre Modbus RTU y Modbus ASCII.

3.5. Objetos, Códigos de Función y Direccionamiento Modbus

Comandos a nivel de Bit

Código de Función	Nombre	Descripción	Valores Soportados	Rango Modbus
01	Leer Bobinas	Este código de función se utiliza para leer el estado contiguo de bobinas en un dispositivo remoto (direcciones 0xxxx). Las bobinas en el mensaje de respuesta se empaquetan como una bobina por bit del campo de datos.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 256 bobinas	**Dirección Local:** 00001-01024 **Dirección del Servidor:** 000001-065536
02	Leer Entradas Discretas	Este código de función se utiliza para leer el estado contiguo de entradas discretas en un dispositivo remoto (direcciones 1xxxx). Las entradas en el mensaje de respuesta se empaquetan como una bobina por bit del campo de datos.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 256 Entradas	**Dirección Local:** 10001-11024 **Dirección del Servidor:** 10001-165536
05	Escribir Bobina Simple	Este código de función se utiliza para escribir una sola bobina a ON o OFF en un dispositivo remoto (direcciones 0xxxx).	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535	**Dirección Local:** 00001-01024 **Dirección del Servidor:** 000001-065536
15	Escribir Múltiples Bobinas	Este código de función se utiliza para escribir en una o más bobinas en una secuencia de bobinas a ON o OFF en un dispositivo remoto (direcciones 0xxxx).	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 256 bobinas	**Dirección Local:** 00001-01024 **Dirección del Servidor:** 000001-065536

Comandos a nivel de Palabra

Código de Función	Nombre	Descripción	Valores Soportados	Rango Modbus
03	Leer Registros de Retención	Este código de función se utiliza para leer el contenido de un bloque contiguo de registros de retención (direcciones 4xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 120 registros	**Dirección Local:** 40001-41024 **Dirección del Servidor:** 400001-465536
04	Leer Registros de Entrada	Este código de función se utiliza para leer el contenido de un bloque contiguo de registros de entrada (direcciones 3xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 120 registros de entrada	**Dirección Local:** 30001-31024 **Dirección del Servidor:** 300001-365536
06	Escribir un Solo Registro de Retención	Este código de función se utiliza para escribir en un solo registro de retención (direcciones 4xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535	**Dirección Local:** 40001-41024 **Dirección del Servidor:** 400001-465536
16	Escribir Múltiples Registros de Retención	Este código de función se utiliza para escribir en registros de retención contiguos (direcciones 4xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 120 registros	**Dirección Local:** 40001-41024 **Dirección del Servidor:** 400001-465536

3.6. Manejo de Errores Modbus RTU

Modbus RTU utiliza una comprobación de redundancia cíclica (CRC) para detectar errores en los datos transmitidos. La CRC es un algoritmo matemático que calcula una suma de comprobación basada en el contenido de la trama. El emisor añade esta suma de comprobación a la trama y el receptor recalcula la CRC al recibirla. Si la CRC calculada coincide con la recibida, la trama se considera libre de errores. De lo contrario, se detecta un error y el receptor puede solicitar una retransmisión. Modbus RTU utiliza un algoritmo CRC de 16 bits, concretamente CRC-16.

3.7. Códigos de Error Modbus RTU

Cuando un dispositivo esclavo detecta un error al procesar una solicitud del dispositivo maestro, responde con un mensaje de excepción que contiene un código de error. Estos códigos de error proporcionan información sobre la naturaleza del error, lo que permite al dispositivo maestro tomar las medidas pertinentes. Algunos códigos de error comunes de Modbus RTU incluyen:

- **Función ilegal (0x01):** El código de error indica que el dispositivo esclavo no admite el código de función solicitado.
- **Dirección de datos ilegal (0x02):** El código de error indica que la dirección de datos solicitada no es válida o está fuera del rango permitido para el dispositivo esclavo.
- **Valor de datos ilegal (0x03):** El código de error indica que el valor de datos proporcionado en la solicitud no es válido o no está permitido por el dispositivo esclavo.
- **Error del dispositivo esclavo (0x04):** El código de error indica que el dispositivo esclavo encontró un error interno al procesar la solicitud.
- **Reconocimiento (0x05):** El error lo envía el dispositivo esclavo para indicar que recibió la solicitud pero necesita tiempo adicional para procesarla.
- **Dispositivo esclavo ocupado (0x06):** El dispositivo esclavo envía este error para indicar que está ocupado ejecutando otro comando. El maestro debe enviar la solicitud una vez que el dispositivo esclavo esté disponible.

3.8. Tiempo de espera y retransmisión

En la comunicación Modbus RTU, el dispositivo maestro espera una respuesta del dispositivo esclavo dentro de un plazo específico, conocido como tiempo de espera. Si el dispositivo maestro no recibe respuesta dentro de este plazo, asume que se ha producido un error, como una trama perdida o un dispositivo esclavo que no responde. En tales casos, el dispositivo maestro puede intentar retransmitir la solicitud o tomar otras medidas adecuadas, como informar del error al usuario o iniciar un proceso de recuperación de fallos.

3.9. Solución de Problemas Modbus RTU

A continuación, se presentan algunos problemas típicos y sus soluciones:

1. Error de comunicación entre los dispositivos maestro y esclavo:

- Verifique el cableado y las conexiones entre los dispositivos, asegurando la terminación y el blindaje adecuados.
- Verificar la configuración de los dispositivos maestros y esclavos, incluida la velocidad en baudios, la paridad y los bits de detención.
- Asegúrese de que la dirección del dispositivo esclavo en la solicitud del maestro coincida con la dirección real del dispositivo esclavo.

2. Valores de datos incorrectos o inconsistentes:

- Inspeccione el esquema de direccionamiento de datos, asegurándose de que el dispositivo maestro esté solicitando datos de los registros o bobinas correctos en el dispositivo esclavo.
- Verificar los tipos de datos y los factores de escala utilizados en la comunicación, garantizando la coherencia entre los dispositivos maestros y esclavos.
- Compruebe si hay posibles fuentes de ruido o interferencias eléctricas que puedan dañar los datos transmitidos.

3. Tiempos de espera o tiempos de respuesta lentos:

- Ajuste la configuración de tiempo de espera en el dispositivo maestro para tener en cuenta los retrasos en la comunicación o los dispositivos esclavos de respuesta lenta.
- Inspeccione la topología de la red y las distancias de comunicación, asegurándose de que estén dentro de los límites especificados por el protocolo Modbus RTU y la interfaz de comunicación serial elegida.
- Optimice el mecanismo de sondeo utilizado por el dispositivo maestro, reduciendo el número de solicitudes o priorizando los dispositivos críticos para minimizar los tiempos de respuesta.

4. Códigos de error o mensajes de excepción:

- Analice los códigos de error devueltos por los dispositivos esclavos para identificar la naturaleza del problema, como códigos de función ilegales, direcciones de datos no válidas o fallas del dispositivo.
- Revise las solicitudes del dispositivo maestro para asegurarse de que sean válidas y compatibles con los dispositivos esclavos.
- Inspeccione los dispositivos esclavos para detectar posibles problemas de hardware o software que puedan causar errores o fallas.

3.10. Requisitos de Hardware - Software Modbus RTU

Especificaciones técnicas para la implementación de RS-485

- **Niveles de Voltaje:** Señalización diferencial con una salida diferencial mínima de 1,5 V.
- **Carga de Bus:** Hasta 32 nodos por segmento con transceptores estándar.
- **Velocidades de Datos:** Hasta 10 Mbps en distancias cortas; las velocidades típicas son de 9600 bps a 115,2 kbps para aplicaciones industriales.
- **Impedancia del Cable:** Impedancia característica de 120 ohmios.
- **Conectores:** Normalmente se utilizan conectores DB-9 o RJ-45.

Requisitos de terminación y sesgo

- Utilice resistencias de terminación de 120 ohmios en ambos extremos del bus RS-485.
- Incluye resistencias pull-up y pull-down (normalmente 10 kΩ) para polarizar el estado inactivo.
- Asegúrese de que la polarización mantenga un voltaje diferencial mínimo de 200 mV cuando el bus esté inactivo.

Requisitos y limitaciones del cable

- **Tipo de Cable:** Cables de par trenzado blindado (STP) o de par trenzado sin blindaje (UTP).
- **Longitud Máxima:** 1200 metros a 9600 bps; longitudes más cortas para velocidades más altas.
- **Blindaje:** utilice cables blindados en entornos ruidosos para reducir la EMI.
- **Calibre del Cable:** 24 AWG o más grueso.

Pautas de Conexión a tierra

- Conecte los blindajes del cable a tierra en un extremo para evitar bucles de tierra.
- Evite la conexión a tierra en múltiples puntos para minimizar la interferencia.
- Asegúrese de que los transceptores RS-485 tengan una conexión a tierra adecuada para mantener la integridad de la señal.

Mecanismos de manejo de errores

- Compruebe si hay desajustes de CRC y errores de registro.
- Implementar tiempos de espera para la recepción de respuestas.
- Detectar y manejar códigos de función o datos no válidos.
- Mecanismos de reintento para solicitudes fallidas.

Criterios de selección de la velocidad en baudios

- 9600 bps: Para comunicaciones de larga distancia (hasta 1200 metros).
- 19200 bps: Distancias medias con ruido moderado.
- 115200 bps: distancias cortas (menos de 10 metros) con ruido mínimo.

3.11. Conexión Drive PowerFlex 4M

Los variadores PowerFlex 4M son compatibles con el protocolo RS-485 (DSI), lo que permite una operación eficiente con periféricos de Rockwell Automation. Además, admiten funciones básicas de Modbus para redes simples, permitiendo su conexión en red mediante el protocolo Modbus en modo RTU.

El cableado de la red consiste en un cable blindado de dos conductores conectado en configuración en serie de nodo a nodo.

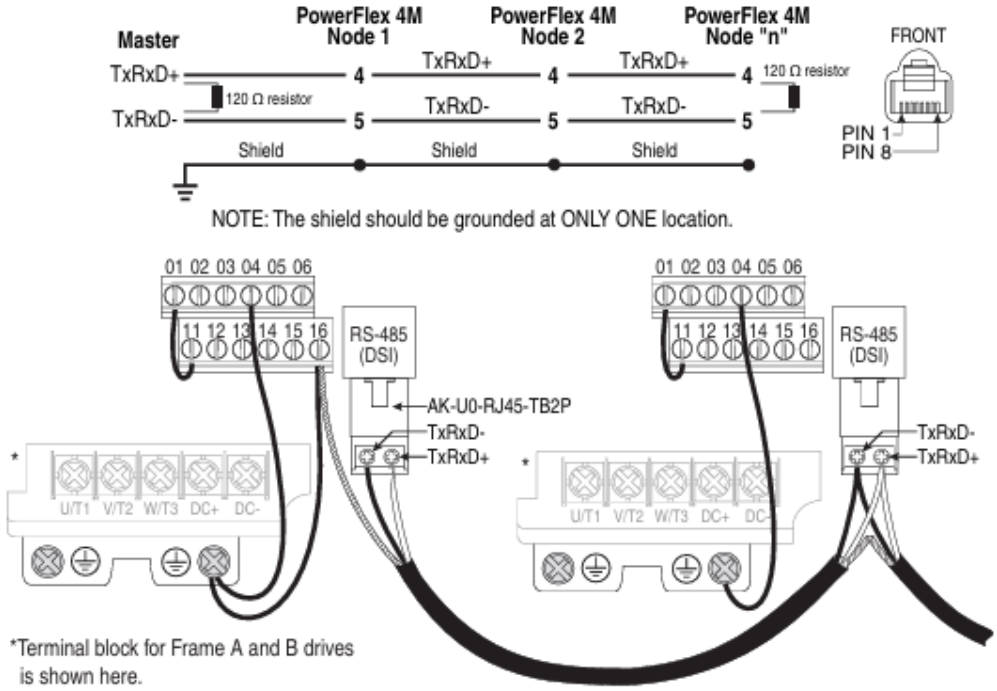


Figura 19. Conexión RS-485 PowerFlex 4M.

Solo deben cablearse los pines 4 y 5 del conector RJ45. Los demás pines del puerto RJ45 del PowerFlex 4M contienen alimentación y otros usos para dispositivos periféricos de Rockwell Automation, por lo que *no deben conectarse*.

3.12. Parametrización Drive PowerFlex 4M para Comunicación Modbus RTU

Los siguientes parámetros del PowerFlex 4M se utilizan para configurar el variador y habilitar su operación en una red:

Parameter	Details	Reference
P106 [Start Source]	Set to 5 "RS485 (DSI) Port" if Start is controlled from the network.	Page 47
P108 [Speed Reference]	Set to 5 "RS485 (DSI) Port" if the Speed Reference is controlled from the network.	Page 49
C302 [Comm Data Rate]	Sets the data rate for the RS485 (DSI) Port. All nodes on the network must be set to the same data rate.	Page 55
C303 [Comm Node Addr]	Sets the node address for the drive on the network. Each device on the network requires a unique node address.	Page 55
C304 [Comm Loss Action]	Selects the drive's response to communication problems.	Page 55
C305 [Comm Loss Time]	Sets the time that the drive will remain in communication loss before the drive implements A105 [Comm Loss Action].	Page 56
C306 [Comm Format]	Sets the transmission mode, data bits, parity and stop bits for the RS485 (DSI) Port. All nodes on the network must be set to the same setting.	Page 56

Figura 20. Parámetros necesarios para habilitar la comunicación por Modbus RTU.

Para realizar dicha parametrización, seguir los siguientes pasos (guiarse en todo caso con la Práctica 2 de la materia de Automatización de Procesos Industriales):

1. **Conexión Física:** Conectar el adaptador 1203-USB al drive PowerFlex 4M y al ordenador.
2. **Preparación en el software CCW:** Abrir el software Connected Components Workbench (CCW). Agregar el drive PowerFlex 4M al proyecto.
3. **Configuración Inicial:** Hacer doble clic en el variador PowerFlex 4M en CCW para abrirlo. Seleccionar Connect. En el Connection Browser, expandir AB_DF1 DH+ Driver, seleccionar AB DSI (PF4 Port) y hacer clic en OK.
4. **Asistente de Inicio:** Una vez conectado, abrir el Startup Wizard. Cambiar los siguientes parámetros:
   - Configurar el puerto de comunicación como referencia de velocidad: **P108 [Speed Reference] = 5 (Comm Port)**.
   - Configurar la fuente de arranque como puerto de comunicación: **P106 [Start Source] = 5 (Comm Port)**.

     Aceptar los valores predeterminados para el resto de las entradas y selecciona Finish.

5. **Ajuste de Parámetros:** En la ventana de Parameters en CCW, ajustar los valores de los parámetros relacionados con la comunicación Modbus RTU (según la tabla a continuación). Esto permite que el variador PowerFlex 4M se comunique con los controladores Micro830/850/870 mediante Modbus RTU.

Parameter	Description	Setting
C302	Comm. Data Rate (Baud Rate) 4 = 19200 bps	4
C303	Communication Node Address (address range is 1...127)	2
C304	Comm. Loss Action (Action that is taken when loss communication) 0 = Fault with coast stop	0
C305	Comm. Loss Time (Time remain in communication before taking action set in C304) 5 sec (max 60)	5
C306	Comm. Format (Data/Parity/Stop) RTU:8 Data Bit, Parity None, 1 Stop bit	0

Figura 21. Parametrización Modbus RTU.

Nota: El parámetro C303 hace referencia a la identificación del variador dentro de la red Modbus. Para el caso particular de esta práctica se dejó el valor de 2, dado que la primera identificación ya se encuentra utilizada en el tablero de prueba.

3.13. Códigos de Función Modbus Compatibles PowerFlex 4M

El PowerFlex 4M admite los siguientes códigos de función Modbus para la comunicación:

Modbus Function Code (Decimal)	Command
03	Read Holding Registers
06	Preset (Write) Single Register
16 (10 Hexadecimal)	Preset (Write) Multiple Registers

Figura 22. Códigos de Función soportados por drive PowerFlex 4M.

3.14. Registros Modbus PowerFlex 4M

Escritura de Datos de Comado Lógico (06) – Registro 8192

El variador PowerFlex 4M puede ser controlado a través de la red enviando escrituras con el Código de Función 06 al registro de dirección 8192 (Logic Command).

Nota: El parámetro **P106 [Start Source]** debe estar configurado en 5 (RS485 - Puerto DSI) para que el variador acepte los comandos enviados.

Este proceso permite enviar instrucciones como arranque, parada y cambios de dirección directamente desde el controlador maestro, como un PLC.

Logic Command		
Address (Decimal)	**Bit(s)**	**Description**
8192	0	1 = Stop, 0 = Not Stop
	1	1 = Start, 0 = Not Start
	2	1 = Jog, 0 = No Jog
	3	1 = Clear Faults, 0 = Not Clear Faults
	5,4	00 = No Command
		01 = Forward Command
		10 = Reverse Command
		11 = No Command
	6	Controls the C-form relay when the value of parameter t221 is set to 13. 1 = On, 0 = Off
	7	1 = MOP Increment, 0 = Not Increment
	9,8	00 = No Command
		01 = Accel Rate 1 Enable
		10 = Accel Rate 2 Enable
		11 = Hold Accel Rate Selected
	11,10	00 = No Command
		01 = Decel Rate 1 Enable
		10 = Decel Rate 2 Enable
		11 = Hold Decel Rate Selected
	14,13,12	000 = No Command
		001 = Freq. Source = P108 [Speed Reference]
		010 = Freq. Source = A409 [Internal Freq]
		011 = Freq. Source = Comms (Addr 8193)
		100 = A410 [Preset Freq 0]
		101 = A411 [Preset Freq 1]
		110 = A412 [Preset Freq 2]
		111 = A413 [Preset Freq 3]
	15	1 = MOP Decrement, 0 = Not Decrement

Figura 23. Registro 8192 (Logic Command).

Escritura de Referencia (06) – Registro 8193

La referencia de velocidad del variador PowerFlex 4M puede ser controlada a través de la red enviando escrituras con el Código de Función 06 al registro de dirección 8193 (Reference).

Requisito: El parámetro **P108 [Speed Reference]** debe estar configurado en 5 (RS485 - Puerto DSI) para que el variador acepte la referencia de velocidad enviada.

Este procedimiento permite ajustar la velocidad del variador directamente desde el dispositivo maestro, como un PLC.

Reference	
Address (Decimal)	**Description**
8193	A decimal value entered as xxx.x where the decimal point is fixed. For example, a decimal "100" equals 10.0 Hz and "543" equals 54.3 Hz.

Figura 24. Registro 8193 (Reference).

Lectura de códigos de error del variador (03) – Registro 8448

El estado lógico del variador PowerFlex 4M puede ser leído a través de la red enviando lecturas con el Código de Función 03 al registro de dirección 8448 (Logic Status).

Logic Status		
Address (Decimal)	**Bit(s)**	**Description**
8448	0	1 = Ready, 0 = Not Ready
	1	1 = Active (Running), 0 = Not Active
	2	1 = Cmd Forward, 0 = Cmd Reverse
	3	1 = Rotating Forward, 0 = Rotating Reverse
	4	1 = Accelerating, 0 = Not Accelerating
	5	1 = Decelerating, 0 = Not Decelerating
	6	1 = Alarm, 0 = No Alarm
	7	1 = Faulted, 0 = Not Faulted
	8	1 = At Reference, 0 = Not At Reference
	9	1 = Reference Controlled by Comm
	10	1 = Operation Cmd Controlled by Comm
	11	1 = Parameters have been locked
	12	Digital Input 1 Status
	13	Digital Input 2 Status
	14	Not Used
	15	Not Used

Figura 25. Registro 8448 (Logic Status).

Lectura de códigos de error del variador (03) – Registro 8449

Los códigos de error del variador PowerFlex 4M pueden ser leídos a través de la red enviando lecturas con el Código de Función 03 al registro de dirección 8449 (Drive Error Codes).

Logic Status		
Address (Decimal)	**Value (Decimal)**	**Description**
8449	0	No Fault
	2	Auxiliary Input
	3	Power Loss
	4	Undervoltage
	5	Overvoltage
	6	Motor Stalled
	7	Motor Overload
	8	Heatsink Overtemperature
	12	HW Overcurrent (300%)
	13	Ground Fault
	29	Analog Input Loss
	33	Auto Restart Tries
	38	Phase U to Ground Short
	39	Phase V to Ground Short
	40	Phase W to Ground Short
	41	Phase UV Short
	42	Phase UW Short
	43	Phase VW Short
	63	Software Overcurrent
	64	Drive Overload
	70	Power Unit Fail
	80	AutoTune Fail
	81	Communication Loss
	100	Parameter Checksum Error
	122	I/O Board Fail

Figura 26. Registro 8449 (Drive Error Codes).

Lectura de Retroalimentación (03) – Registro 8451

La retroalimentación de la frecuencia de salida del variador PowerFlex 4M puede ser leída a través de la red enviando lecturas con el Código de Función 03 al registro de dirección 8451 (Feedback).

Feedback[1]	
Address (Decimal)	**Description**
8451	A xxx.x decimal value where the decimal point is fixed. For example, a decimal "123" equals 12.3 Hz and "300" equals 30.0 Hz.

Figura 27. Registro 8451 (Feedback).

Lectura (03) y Escritura (6) de Parámetros del Drive

Para acceder a los parámetros del drive, la dirección del registro Modbus es igual al número del parámetro. Por ejemplo, se utiliza un decimal "1" para direccionar el parámetro d001 [Frecuencia de salida] y un decimal "109" para direccionar el parámetro P109 [Tiempo de aceleración 1].

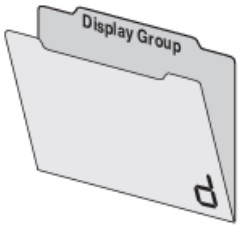
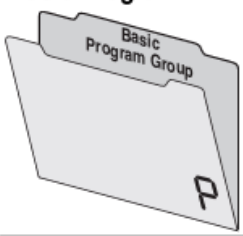
Group	Parameters			
Basic Display	Output Freq	d001	Control Source	d012
	Commanded Freq	d002	Contrl In Status	d013
	Output Current	d003	Dig In Status	d014
	Output Voltage	d004	Comm Status	d015
	DC Bus Voltage	d005	Control SW Ver	d016
	Drive Status	d006	Drive Type	d017
	Fault 1 Code	d007	Elapsed Run Time	d018
	Fault 2 Code	d008	Testpoint Data	d019
	Fault 3 Code	d009	Analog In 0-10V	d020
	Process Display	d010	Analog In 4-20mA	d021
			Drive Temp	d022
Basic Program	Motor NP Volts	P101	Stop Mode	P107
	Motor NP Hertz	P102	Speed Reference	P108
	Motor OL Current	P103	Accel Time 1	P109
	Minimum Freq	P104	Decel Time 1	P110
	Maximum Freq	P105	Motor OL Ret	P111
	Start Source	P106	Reset To Defalts	P112

Figura 28. Parámetros Básicos de Lectura y Escritura PowerFlex 4M.

3.15. Instrucción MSG_MODBUS

Envía un mensaje Modbus a través de un puerto serie.

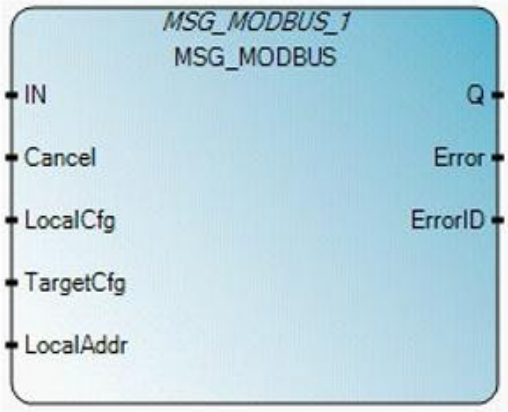


Figura 29. Instrucción MSG_MODBUS.

A continuación, los parámetros de esta instrucción:

Parámetro	Tipo de parámetro	Tipo de datos	Descripción
IN	Entrada	BOOL	Estado de entrada de línea. Cierto: se ha detectado un flanco ascendente, inicia el bloque de instrucción con la condición previa de que la última operación esté completa. Falso: no se ha detectado un flanco ascendente, no iniciado.
Cancel	Entrada	BOOL	CIERTO: cancela la ejecución del bloque de función. FALSO: si IN es Cierto. Cancelar entrada es dominante.
LocalCfg	Entrada	MODBUSLOCPARA	Define la entrada de la estructura (dispositivo local). Define la estructura de entrada para el dispositivo local utilizando el tipo de datos MODBUSLOCPARA en la página 180.
TargetCfg	Entrada	MODBUSTARPARA	Define la entrada de la estructura (dispositivo de destino). Define la estructura de entrada para el dispositivo de destino mediante el tipo de datos MODBUSTARPARA en la página 183.
LocalAddr	Entrada	MODBUSLOCADDR	MODBUSLOCADDR es una matriz de 125 palabras que utilizan los comandos Read para almacenar los datos (1-125 palabras) que devuelve el esclavo Modbus y los comandos Write para almacenar en búfer los datos (1-125 palabras) que se enviarán al dispositivo esclavo Modbus.
Q	Salida	BOOL	Las salidas de esta instrucción se actualizan de forma asíncrona desde la exploración del programa. La salida Q no se puede utilizar para volver a activar la instrucción ya que IN tiene el flanco activado. CIERTO: la instrucción MSG ha finalizado correctamente. FALSO: la instrucción MSG no se finaliza.
Error	Salida	BOOL	Indica que se ha producido un error. CIERTO: se ha detectado un error. FALSO: no hay ningún error.
ErrorID	Salida	UINT	Un número único que identifica el error. Los errores de esta instrucción se definen en códigos de error MSG_MODBUS.

Figura 30. Parámetros instrucción MSG_MODBUS.

Tipo de datos MODBUSLOCPARA

Utilice esta tabla para determinar los valores de los parámetros para el tipo de datos MODBUSLOCPARA.

Parámetro	Tipo de datos	Descripción
Channel	UINT	Número de puerto de serie de Micro800 PLC: • 2 para el puerto serie integrado, o bien • 5-9 para módulos enchufables de puerto serie instalados en las ranuras 1 a • 5 para la ranura 1 • 6 para la ranura 2 • 7 para la ranura 3 • 8 para la ranura 4 • 9 para la ranura 5
TriggerType	USINT	Representa una de las siguientes acciones: • 0: mensaje activado una vez (si IN cambia de Falso a Cierto) • 1: mensaje activado continuamente si IN es Cierto • Otro valor: Reservado

Parámetro	Tipo de datos	Descripción
Cmd	USINT	Representa una de las siguientes acciones: • 01: Lectura de estado de bobina (0xxxx) • 02: Lectura de estado de entrada (1xxxx) • 03: Lectura de Holding Registers (4xxxx) • 04: Lectura de registros de entrada (3xxxx) • 05: Escritura de bobina simple (0xxxx) • 06: Escritura de registro único (4xxxx) • 15: Escritura de bobinas múltiples (0xxxx) • 16: Escritura de registros múltiples (4xxxx) • Otros: Compatibilidad con comandos personalizados. Compatibilidad con comandos personalizados MODBUSLOCPARA: También se admiten los comandos personalizados en el intervalo de 0 a 255 que aún no están asignados a un comando de Modbus. Si se utiliza un comando personalizado, LocalCfg:ElementCnt contiene el número de bytes recibidos. La respuesta se recibe en los datos de dirección local y sobrescribe los datos de la solicitud. • Ejemplo de CMD=0x2B • Datos de dirección local 1:0x0E, READ_DEVICE_ID_MEI • Datos de dirección local 2:0x01, READ_DEV_ID_BASIC • Datos de dirección local 3:0x00, Lectura de objeto de proveedor
ElementCnt	UINT	Límites • Para entradas de lectura de bobina/discretas: 2000 bits • Para lectura de registros: 125 palabras • Para escritura de bobinas: 1968 bits • Para escritura de registros: 123 palabras

Figura 31. Parámetro LocalCfg.

Tipo de datos MODBUSTARPARA

En la tabla siguiente se describe el tipo de datos MODBUSTARPARA.

Nombre	Tipo de datos	Descripción
Addr	UDINT	Dirección de datos de destino (1 - 65536). Se reduce en uno cuando se envía.
Node	USINT	La dirección de nodo esclavo predeterminada es 1. El intervalo es de 1 a 247. La dirección de difusión de Modbus es 0 y solo es válida para los comandos de escritura de Modbus (por ejemplo: 5, 6, 15 y 16).

Figura 32. Parámetro TargetCfg.

4. Arquitectura de Comunicación

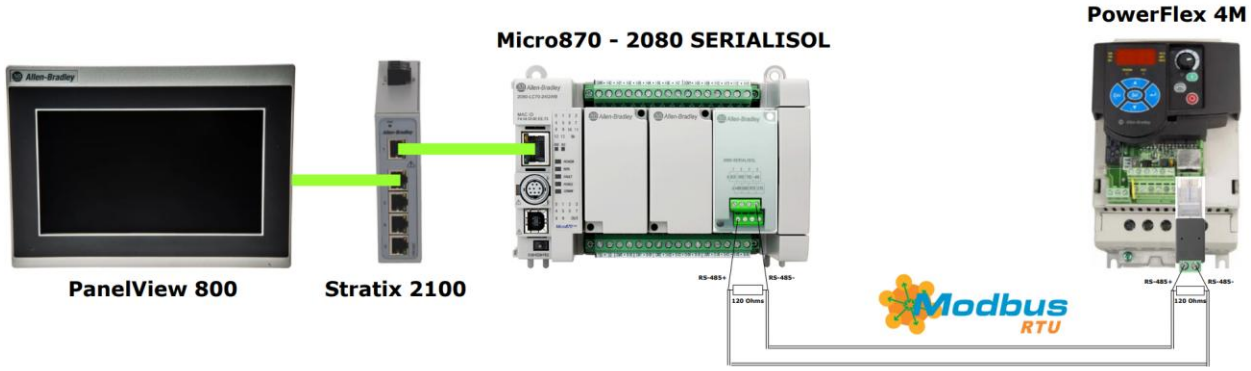


Figura 33. Arquitectura de Comunicación.

5. Procedimiento

Previo a realizar la configuración del módulo 2080-SERIALISOL y la programación en CCW, debe haber parametrizado correctamente al drive PowerFlex 4M (Grupos Basic Program y Communications).

Para efectos de esta práctica se trabajará con un Motor Siemens 1LA7 070-4YA60, el cual tiene las siguientes especificaciones de placa:

Tipo	Frame IEC	Potencia		Factor de Servicio	Corriente Nominal [A]		Eficiencia n	Factor de Potencia	Velocidad Nominal	Torque Nominal	Torque de Arranque	Corriente de Arranque	Peso Neto IM B3	Rodamientos
		HP	Kw		440V	220V	(%)	Cos φ	(rpm)	[Nm]	(Ta/Tn)	(Ia/In)	[kg]	AS/BS
MOTORES DE 4 POLOS, 1800 RPM														
1LA7 070-4YC60	71	0.4	0.3	1.05	1.6	0.8	63.3	0.77	1640	1.7	1.8	2.8	4.7	6202 2Z C3 / 6202 2Z C3
1LA7 070-4YA60	71	0.5	0.4	1.15	1.9	0.95	63.6	0.81	1590	2.24	1.3	2.7	4.7	6202 2Z C3 / 6202 2Z C3
1LA7 071-4YA60	71	0.6	0.4	1.15	2.2	1.1	70.2	0.76	1680	2.54	1.8	3.4	6.0	6202 2Z C3 / 6202 2Z C3

Figura 34. Datos de Placa Motor Siemens 1LA7 070-4YA60.

Luego, confirmar los siguientes parámetros para efectuar correctamente la comunicación Modbus RTU a través de RS-485 en el Drive PowerFlex 4M:

Grupo Basic Program

Figura 35. Parámetro P106 (Start Source).

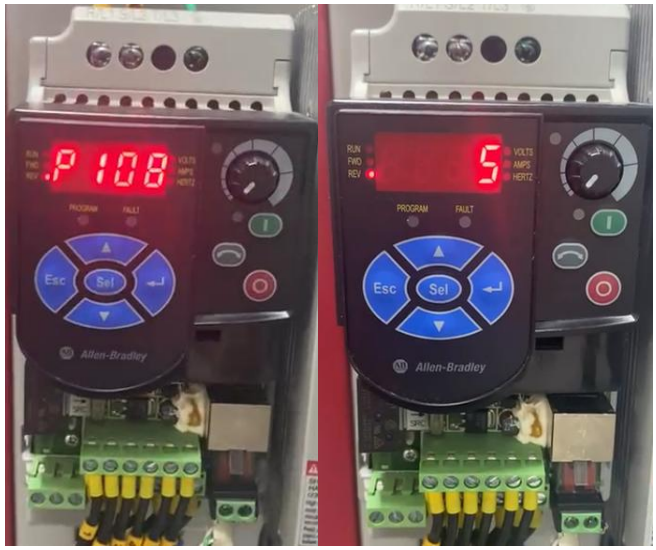


Figura 36. Parámetro P108 (Reference Source).

Grupo Communications

Figura 37. Parámetro C302 (Comm Data Rate).

Figura 38. Parámetro C303 (Comm Node Addr).

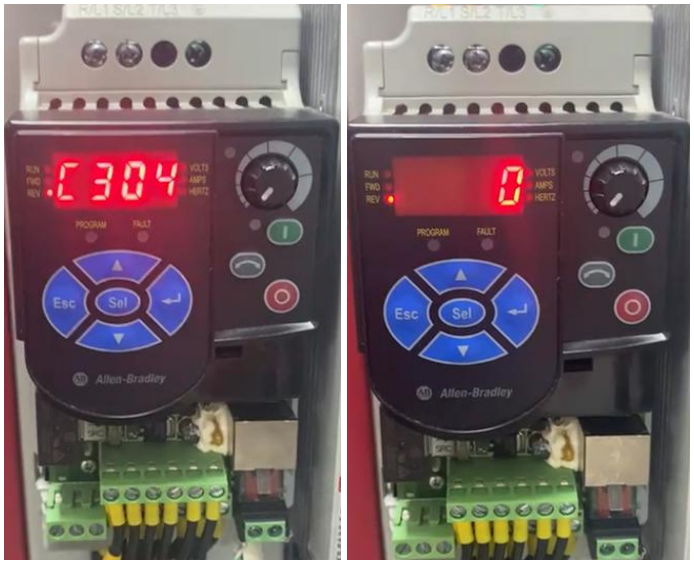


Figura 39. Parámetro C304 (Comm Loss Action).

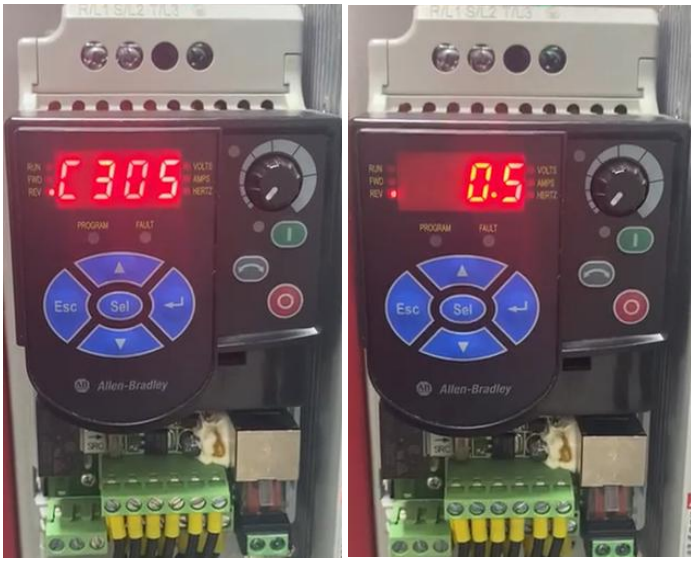


Figura 40. Parámetro C305 (Comm Loss Time).

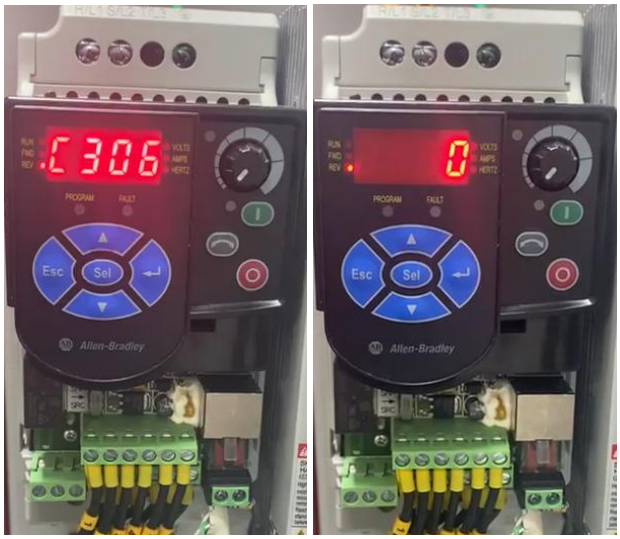


Figura 41. Parámetro C305 (Comm Write Mode).

5.1. Configuración Módulo 2080-SERIALISOL en CCW

- Para agregar el módulo de comunicación dar clic derecho en el slot correspondiente del Micro870, seleccionar *Communication*, y escoger el módulo 2080-SERIALISOL**.**

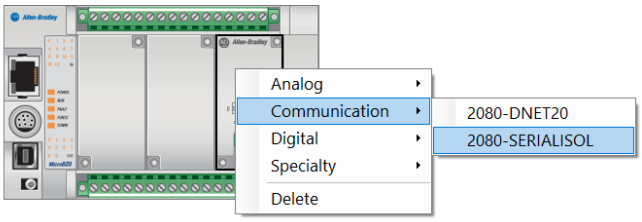


Figura 42. Agregar Módulo 2080-SERIALISOL.

- Buscar el módulo serial en *Plug-in Modules* y dar click en *Configuration*. En *Driver* escoger la opción de *Modbus RTU*. Escoger los mismos baudios con los que se parametrizó el Powerflex 4M (19200). Escoger en *Modbus Role* la opción de Master. Cerciorarse que en *Media* se encuentre seleccionada la interfaz RS-485.

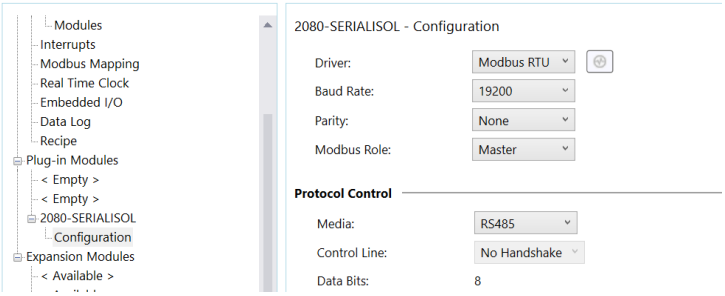


Figura 43. Configuración Módulo 2080-SERIALISOL.

5.2. Configuración Instrucción MSG_MODBUS

5.2.1. Escritura Registro 8192 (Logic Command Data)

- En el Diagrama Ladder, arrastrar un bloque de instrucción, y buscar MSG_MODBUS.

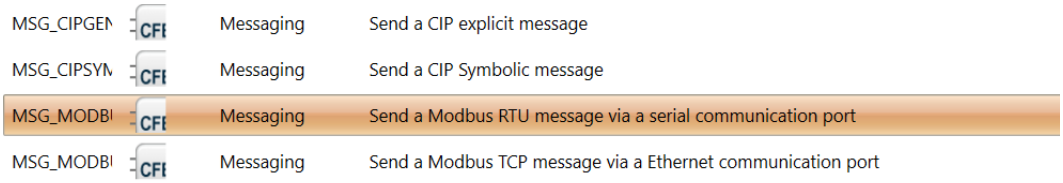


Figura 44. Buscar Instrucción MSG_MODBUS.

- Realizar la siguiente declaración de variables en el bloque de la instrucción.

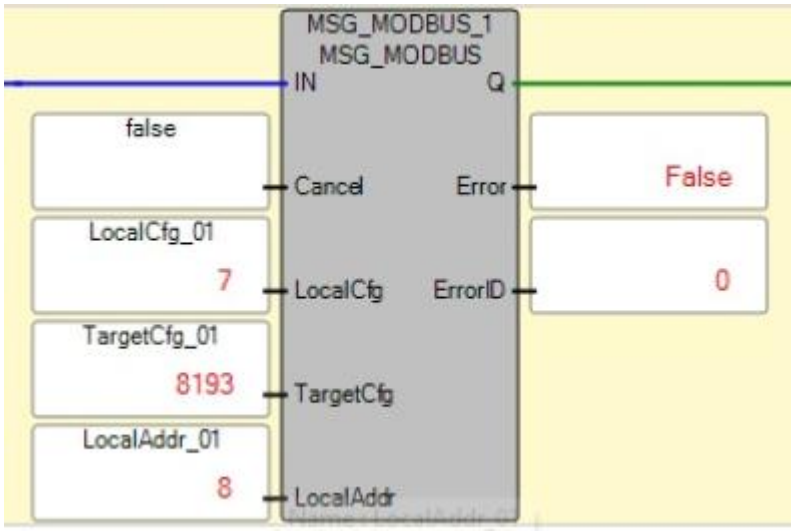


Figura 45. Declaración de Variables.

Nota: El scope de los parámetros de la instrucción MSG_MODBUS (LocalCfg_0X, TargetCfg_0X, LocalAddr_0X) puede ser global o local sin ningún problema. Preferiblemente trabaje con un scope global.

- A continuación, ir a Global Variables, y desplegar la variable LocalCfg_01**.** Escribir los siguientes valores en *Initial Values*. En *Channel* colocar el valor de 7, debido a que este indica el identificador de puerto serie (en este caso el 7 se refiere al tercer slot). En *Trigger* colocar el valor de 0, lo cual activa el mensaje sólo una vez (si IN recibe un flanco positivo). En *Cmd* colocar el valor de 6, debido a que se utilizará la función 0x06 de Modbus (Write Single Register). En Element colocar el valor de 1, debido a que se escribirá un solo registro.

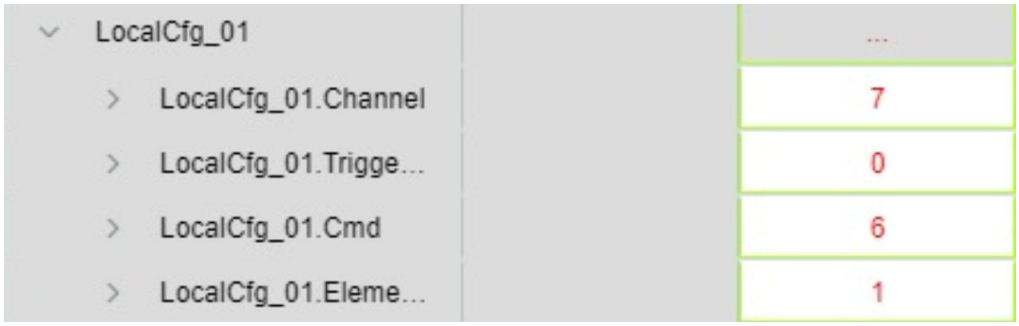


Figura 46. Configuración Parámetro LocalCfg.

- A continuación, desplegar la variable TargetCfg_01**.** Escribir los siguientes valores en *Initial Values*. En *Addr* colocar el valor de 8193, debido a que se pretende apuntar al registro 8192 (durante la comunicación se le resta 1). En *Node* colocar el valor de 2, debido a que este es el Slave-ID que se le asignó al drive.

Figura 47. Configuración Parámetro TargetCfg.

- Implementar la siguiente lógica de programación.

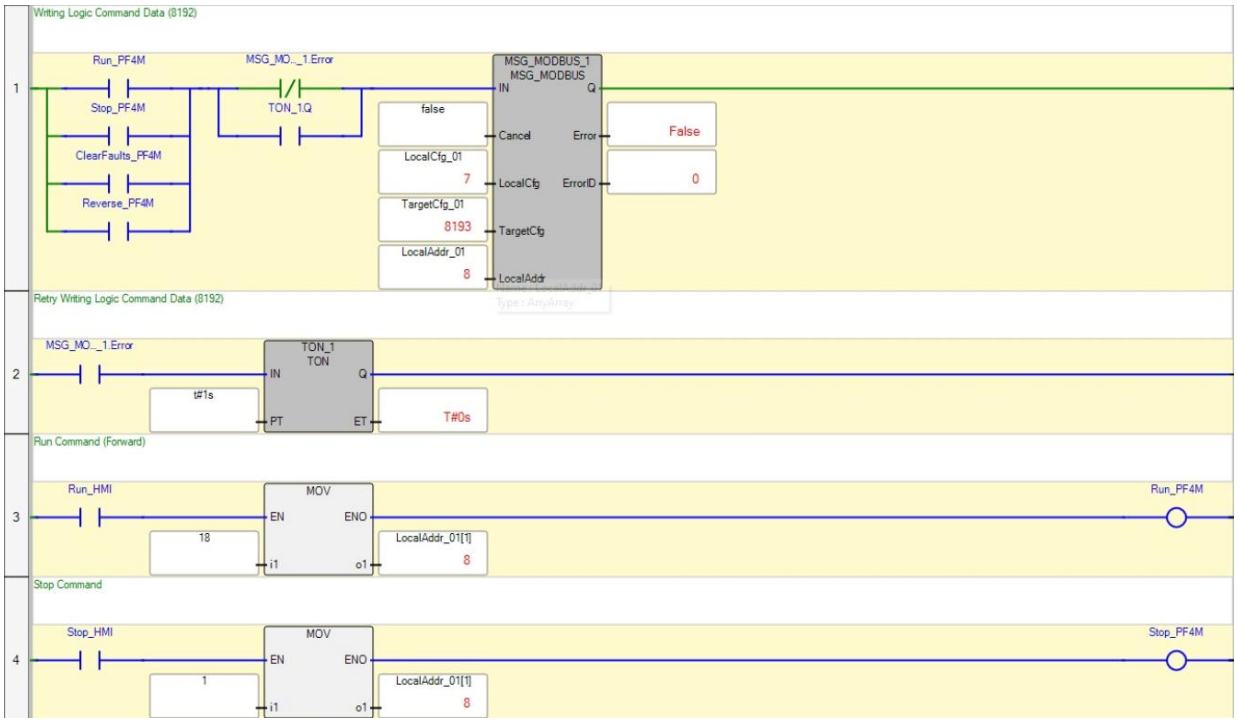


Figura 48. Lógica de Programación Registro 8192.

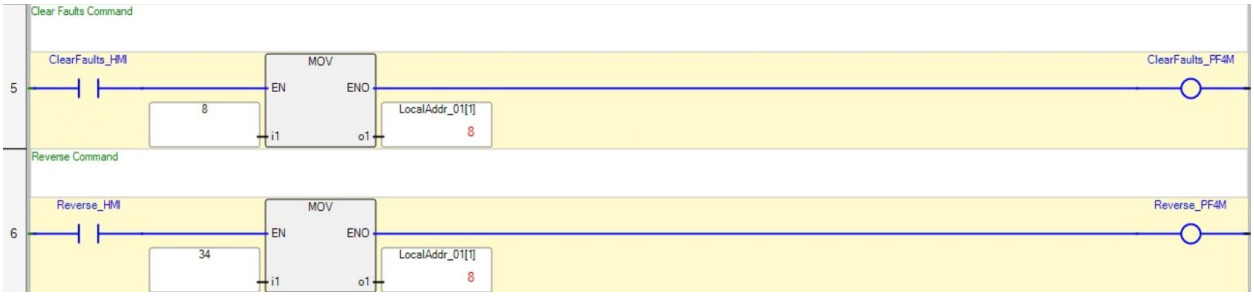


Figura 49. Lógica de Programación Registro 8192.

Nota: En esta lógica de programación lo que se añadió inicialmente fue un tratamiento de reintento de comunicación en el caso de error mediante una instrucción TON. Luego, los siguientes renglones hacen referencia a los comandos a enviar mediante comunicación al drive. Estos comandos están vinculados a botones momentáneos en la interfaz gráfica. Cuando alguno de estos se active, se ejecutará una instrucción MOV, la cual traspasará el contenido binario en formato entero al parámetro LocalAddr_01[1] (primer registro de la estructura LocalAddr_01), el cual es de tipo WORD.

Por ejemplo, para mandar el comando de Run con dirección Forward, se asigna el valor 18 en entero, o en formato binario 00010010, donde el bit .1 activa la acción Start, y la combinación 01 para los bits .5 y .4 activa la acción Forward.

Para mandar el comando de Stop, se asigna el valor de 1 o en formato binario 00000001, donde solo se activa la acción de Stop en el bit .0.

Para mandar el comando de ClearFaults, se asigna el valor de 8 o en formato binario 00001000, donde solo se activa la acción de Clear Faults en el bit .3.

Para mandar el comando de Run con dirección Reverse, se asigna el valor de 34 o en formato binario 00100010, donde el bit .1 activa la acción Start, y la combinación 10 para los bits .5 y .4 activa la acción Reverse.

Cuando se complete la acción de MOV, se activarán ciertas marcas las cuales se colocan en paralelo en el renglón de la instrucción MSG_MODBUS, logrando así un flanco positivo en IN por cada comando.

- Implementar la siguiente lógica de corrección de comunicación. La configuración de parámetros de esta instrucción MSG_MODBUS_3 es la misma que la de MSG_MODBUS_1, salvo en el valor de *Trigger*, al cual se le coloca el valor de 1 para que se ejecute continuamente cada que IN esté activado.

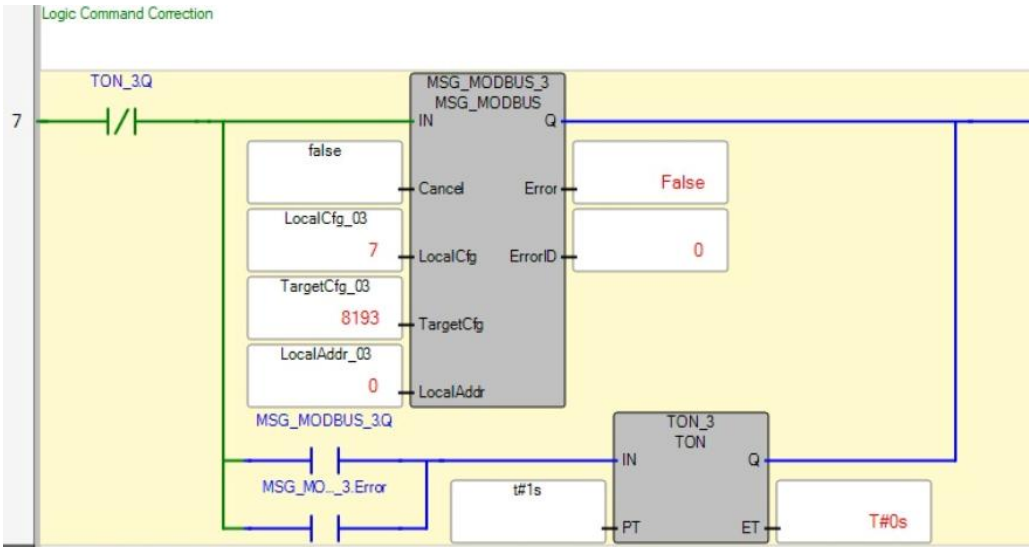


Figura 50. Lógica de Corrección Registro 8192.

Nota: El propósito de esta lógica es encerar continuamente el registro 8192. Sólo cuando se accionen alguno de los comandos antes mencionados de forma asíncrona, el mensaje tendrá un parecido a un flanco positivo seguido de un flanco negativo de forma casi instantánea.

La razón del por qué se necesita esta lógica con este registro en particular se debe a resultados experimentales, donde a pesar de que el Trigger de MSG_MODBUS_1 sea 0 o 1, al ejecutarse la comunicación se lleva inevitablemente a un fallo de comunicación (código 81). Al implementar esta lógica, se resuelve dicho problema.

5.2.2. Escritura Registro 8193 (Reference)

- Realizar la siguiente declaración de variables en el bloque de la instrucción.

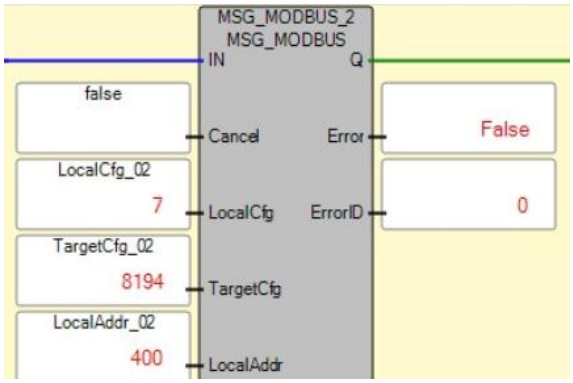


Figura 51. Declaración de Variables.

- A continuación, ir a Global Variables, y desplegar la variable LocalCfg_02**.** Escribir los siguientes valores en *Initial Values*. En *Channel* colocar el valor de 7, debido a que este indica el identificador de puerto serie (en este caso el 7 se refiere al tercer slot). En *Trigger* colocar el valor de 0, lo cual activa el mensaje sólo una vez (si IN recibe un flanco positivo). En *Cmd* colocar el valor de 6, debido a que se utilizará la función 0x06 de Modbus (Write Single Register). En Element colocar el valor de 1, debido a que se escribirá un solo registro.

Figura 52. Configuración Parámetro LocalCfg.

- A continuación, desplegar la variable TargetCfg_02**.** Escribir los siguientes valores en *Initial Values*. En *Addr* colocar el valor de 8194, debido a que se pretende apuntar al registro 8193 (durante la comunicación se le resta 1). En *Node* colocar el valor de 2, debido a que este es el Slave-ID que se le asignó al drive.

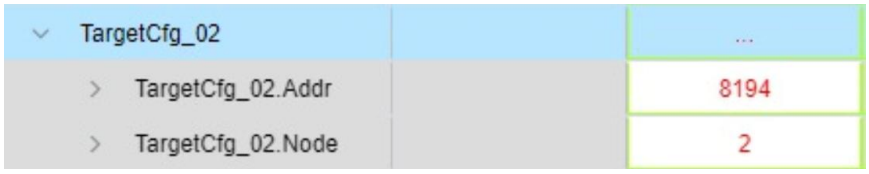


Figura 53. Configuración Parámetro TargetCfg.

- Implementar la siguiente lógica de programación.

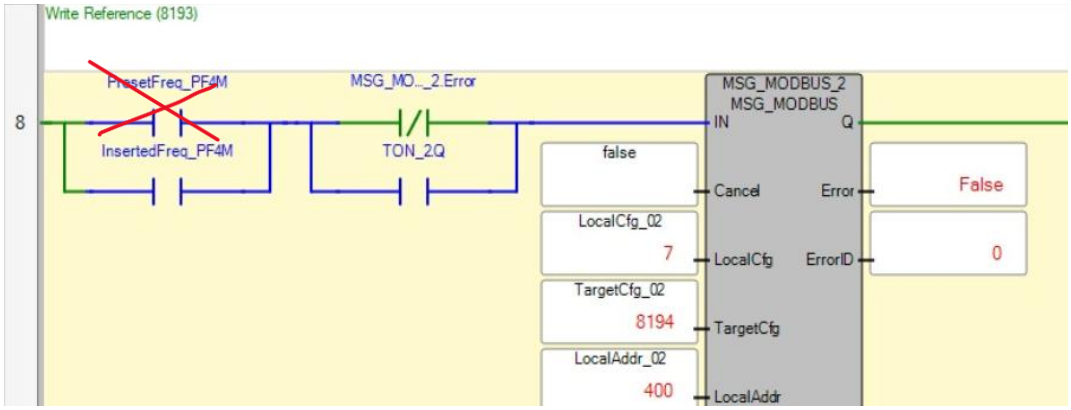


Figura 54. Lógica de Programación Registro 8193.

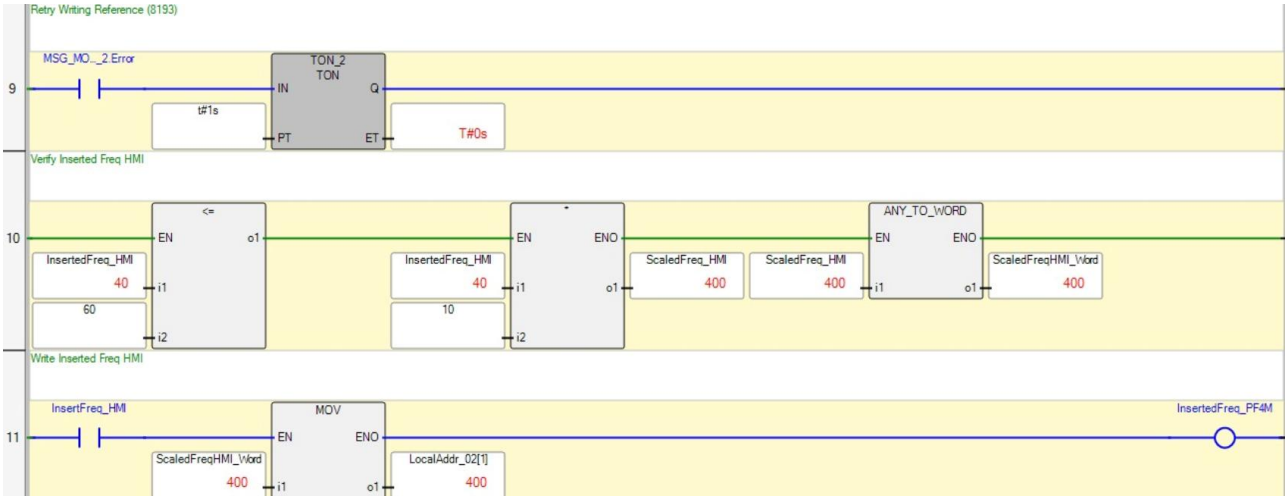


Figura 55. Lógica de Programación Registro 8193.

Nota: En esta lógica de programación lo que se añadió inicialmente fue un tratamiento de reintento de comunicación en el caso de error mediante una instrucción TON. Luego, en el siguiente renglón se realiza una verificación del valor de frecuencia ingresado desde la interfaz gráfica (si dicho valor es menor igual a 60). Luego se escala dicho valor por un factor de 10 (debido al formato interno del drive, donde por ejemplo un valor de 400 equivale a 40.0 Hz). Luego se convierte dicho valor escalado de tipo INT a tipo WORD para poder ejecutar sin ningún error la instrucción MOV. Cuando se active el botón momentáneo de ingresar frecuencia, se ejecutará una instrucción MOV, la cual traspasará el contenido binario en formato entero al parámetro LocalAddr_02[1] (primer registro de la estructura LocalAddr_02), el cual es de tipo WORD.

Cuando se complete la acción de MOV, se activará una marca la cual se coloca al inicio del renglón de la instrucción MSG_MODBUS, logrando así un flanco positivo en IN por cada ingreso de frecuencia.

5.2.3. Lectura Registro 8448 (Logic Status Data)

- Realizar la siguiente declaración de variables en el bloque de la instrucción.

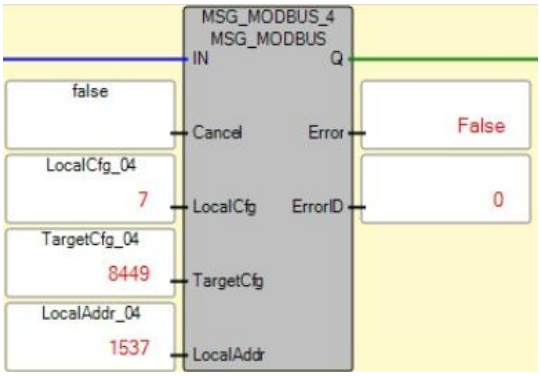


Figura 56. Declaración de Variables.

- A continuación, ir a Global Variables, y desplegar la variable LocalCfg_04**.** Escribir los siguientes valores en *Initial Values*. En *Channel* colocar el valor de 7, debido a que este indica el identificador de puerto serie (en este caso el 7 se refiere al tercer slot). En *Trigger* colocar el valor de 0, lo cual activa el mensaje sólo una vez (si IN recibe un flanco positivo). En *Cmd* colocar el valor de 3, debido a que se utilizará la función 0x03 de Modbus (Read Single Register). En Element colocar el valor de 1, debido a que se leerá un solo registro.

Figura 57. Configuración Parámetro LocalCfg.

- A continuación, desplegar la variable TargetCfg_04**.** Escribir los siguientes valores en *Initial Values*. En *Addr* colocar el valor de 8449, debido a que se pretende apuntar al registro 8448 (durante la comunicación se le resta 1). En *Node* colocar el valor de 2, debido a que este es el Slave-ID que se le asignó al drive.

Figura 58. Configuración Parámetro TargetCfg.

- Implementar la siguiente lógica de programación.

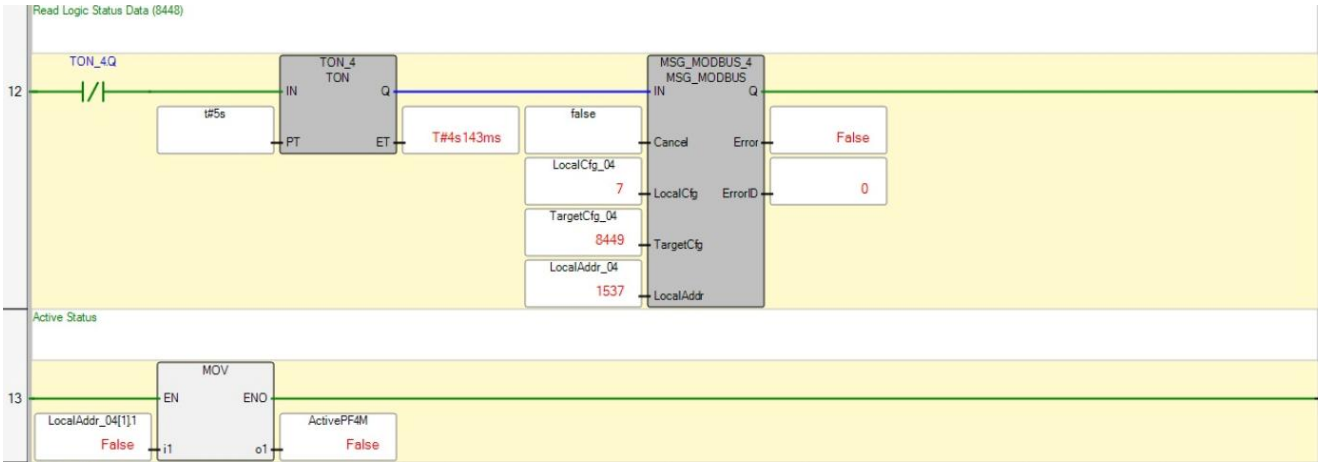


Figura 59. Lógica de Programación Registro 8448.

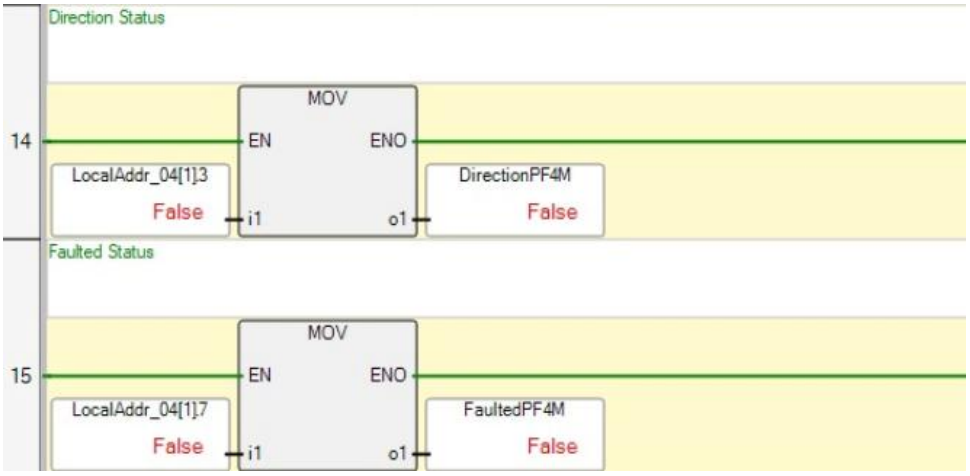


Figura 60. Lógica de Programación Registro 8448.

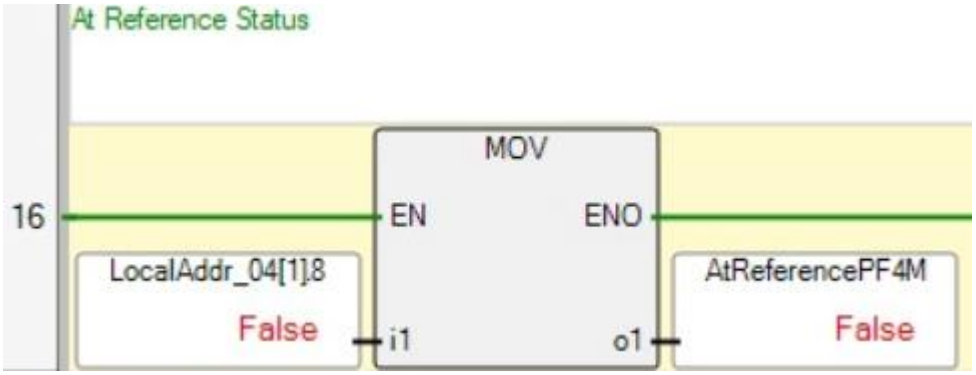


Figura 61. Lógica de Programación Registro 8448.

Nota: En esta lógica de programación lo que se añadió inicialmente fue una especie de Clock Memory mediante un TON con autoreinicio, de tal manera que cada 5 segundos se realiza la operación de lectura. Luego, en los siguientes renglones se implementaron instrucciones MOV donde se obtienen las variables de estado del drive accediendo a los bits correspondientes del registro LocalAddr_04[1].

Por ejemplo, el bit .1 del registro 8448 hace referencia a la variable de estado Active.

Por ejemplo, el bit .3 del registro 8448 hace referencia a la variable de estado Direction (Forward o Reverse).

Por ejemplo, el bit .7 del registro 8448 hace referencia a la variable de estado Faulted.

Por ejemplo, el bit .8 del registro 8448 hace referencia a la variable de estado At Reference.

5.2.4. Lectura Registro 8451 (Feedback)

- Realizar la siguiente declaración de variables en el bloque de la instrucción.

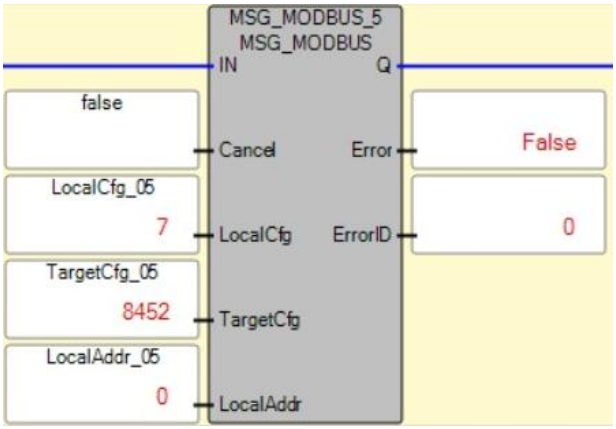


Figura 62. Declaración de Variables.

- A continuación, ir a Global Variables, y desplegar la variable LocalCfg_05**.** Escribir los siguientes valores en *Initial Values*. En *Channel* colocar el valor de 7, debido a que este indica el identificador de puerto serie (en este caso el 7 se refiere al tercer slot). En *Trigger* colocar el valor de 0, lo cual activa el mensaje sólo una vez (si IN recibe un flanco positivo). En *Cmd* colocar el valor de 3, debido a que se utilizará la función 0x03 de Modbus (Read Single Register). En Element colocar el valor de 1, debido a que se leerá un solo registro.

Figura 63. Configuración Parámetro LocalCfg.

- A continuación, desplegar la variable TargetCfg_05**.** Escribir los siguientes valores en *Initial Values*. En *Addr* colocar el valor de 8452, debido a que se pretende apuntar al registro 8451 (durante la comunicación se le resta 1). En *Node* colocar el valor de 2, debido a que este es el Slave-ID que se le asignó al drive.

Figura 64. Configuración Parámetro TargetCfg.

- Implementar la siguiente lógica de programación.

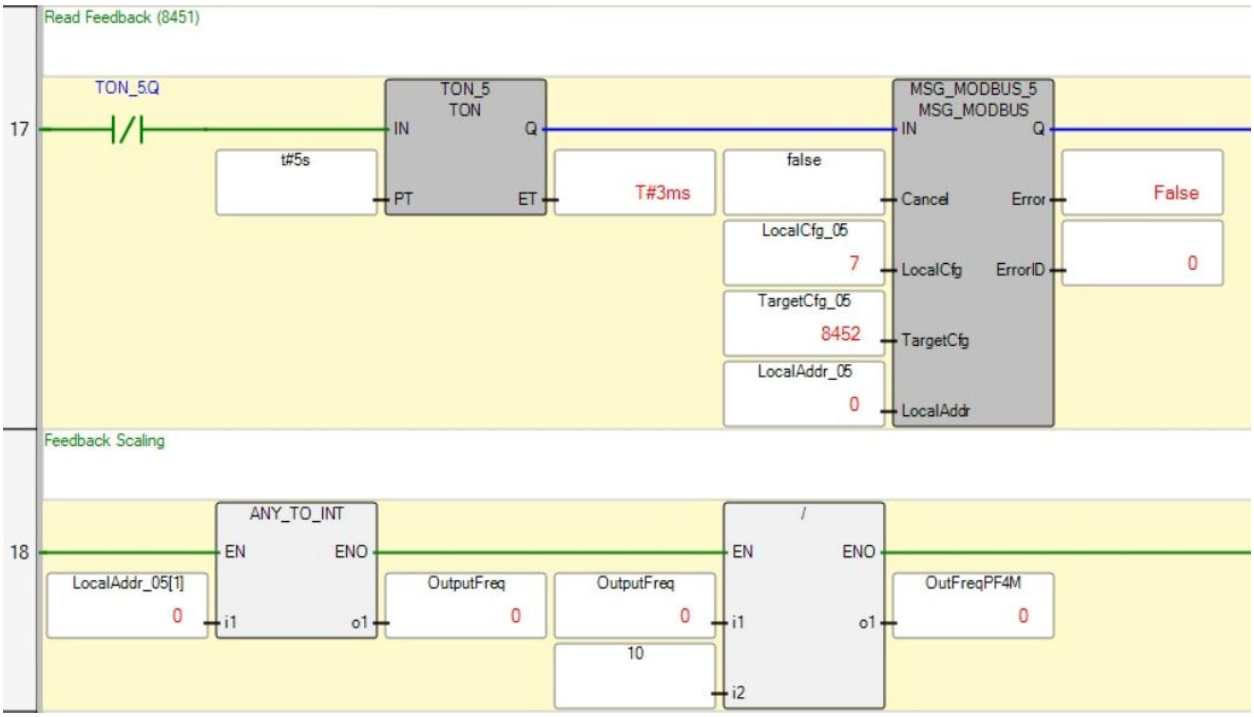


Figura 65. Lógica de Programación Registro 8451.

Nota: En esta lógica de programación lo que se añadió inicialmente fue una especie de Clock Memory mediante un TON con autoreinicio, de tal manera que cada 5 segundos se realiza la operación de lectura. Luego, en el siguiente renglón se realiza la conversión del valor obtenido en LocalAddr_05[1] a tipo INT, luego dicho valor se lo escala por un factor de 0.1 (esto debido al formato interno del drive).

5.2.5. Lectura Registro 8449 (Error Code)

- Realizar la siguiente declaración de variables en el bloque de la instrucción.

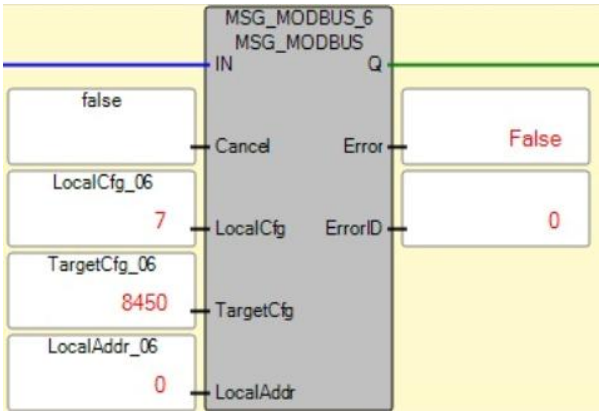


Figura 66. Declaración de Variables.

- A continuación, ir a Global Variables, y desplegar la variable LocalCfg_06**.** Escribir los siguientes valores en *Initial Values*. En *Channel* colocar el valor de 7, debido a que este indica el identificador de puerto serie (en este caso el 7 se refiere al tercer slot). En *Trigger* colocar el valor de 0, lo cual activa el mensaje sólo una vez (si IN recibe un flanco positivo). En *Cmd* colocar el valor de 3, debido a que se utilizará la función 0x03 de Modbus (Read Single Register). En Element colocar el valor de 1, debido a que se leerá un solo registro.

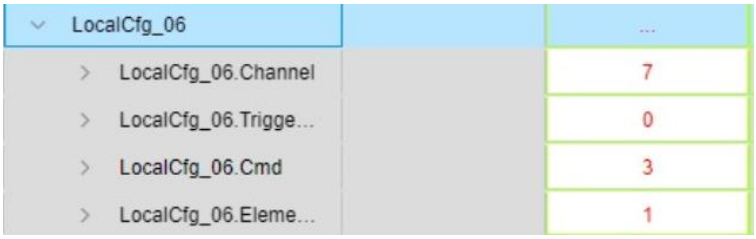


Figura 67. Configuración Parámetro LocalCfg.

- A continuación, desplegar la variable TargetCfg_06**.** Escribir los siguientes valores en *Initial Values*. En *Addr* colocar el valor de 8450, debido a que se pretende apuntar al registro 8449 (durante la comunicación se le resta 1). En *Node* colocar el valor de 2, debido a que este es el Slave-ID que se le asignó al drive.

Figura 68. Configuración Parámetro TargetCfg.

- Implementar la siguiente lógica de programación.

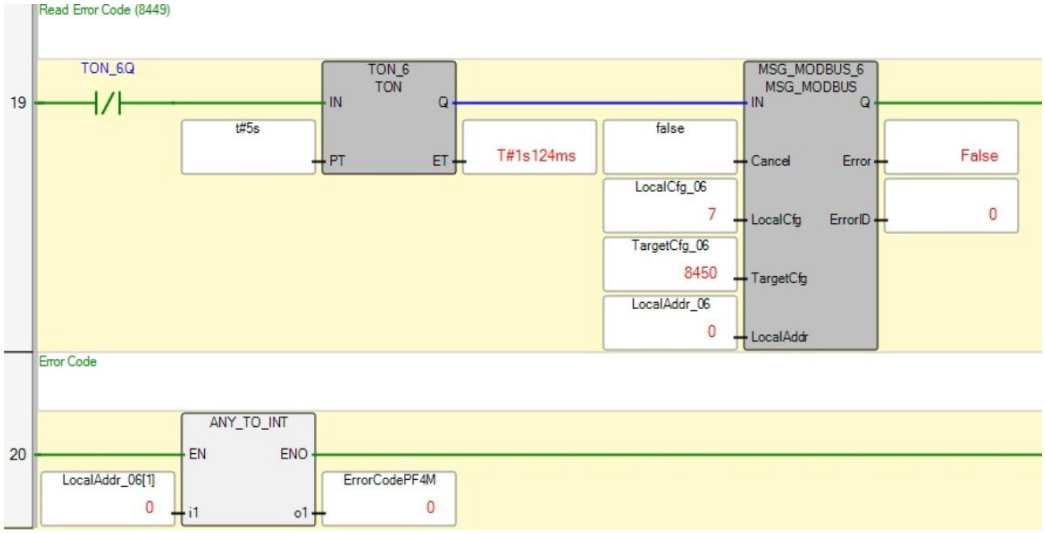


Figura 69. Lógica de Programación Registro 8449.

Nota: En esta lógica de programación lo que se añadió inicialmente fue una especie de Clock Memory mediante un TON con autoreinicio, de tal manera que cada 5 segundos se realiza la operación de lectura. Luego, en el siguiente renglón se realiza la conversión del valor obtenido en LocalAddr_06[1] a tipo INT.

5.2.6. Lectura y Escritura Parámetros PowerFlex 4M

- Mediante la instrucción MSG_MODBUS también se puede tanto leer como escribir parámetros del drive. Para efectos de esta práctica se realizará la lectura de la Corriente de Salida (Parámetro d003).
- Realizar la siguiente declaración de variables en el bloque de la instrucción.

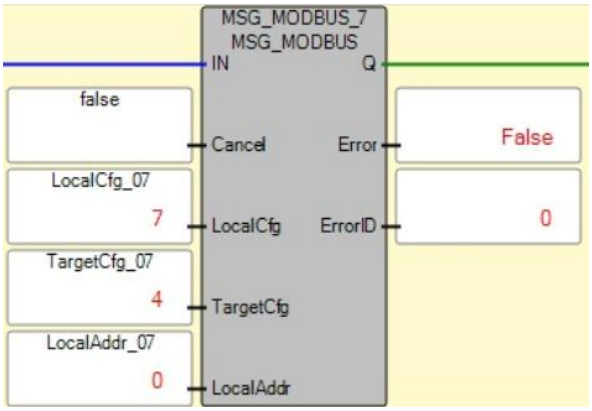


Figura 70. Declaración de Variables.

- A continuación, ir a Global Variables, y desplegar la variable LocalCfg_07**.** Escribir los siguientes valores en *Initial Values*. En *Channel* colocar el valor de 7, debido a que este indica el identificador de puerto serie (en este caso el 7 se refiere al tercer slot). En *Trigger* colocar el valor de 0, lo cual activa el mensaje sólo una vez (si IN recibe un flanco positivo). En *Cmd* colocar el valor de 3, debido a que se utilizará la función 0x03 de Modbus (Read Single Register). En Element colocar el valor de 1, debido a que se leerá un solo registro.

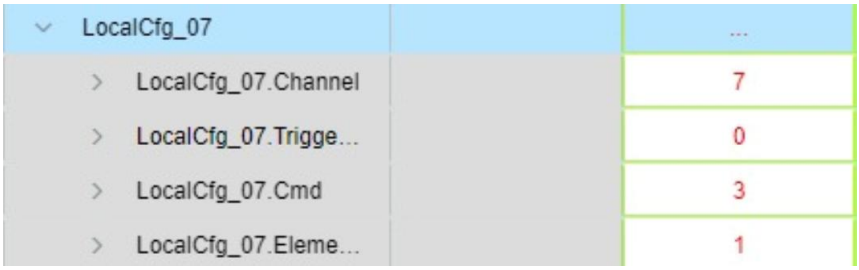


Figura 71. Configuración Parámetro LocalCfg.

- A continuación, desplegar la variable TargetCfg_07**.** Escribir los siguientes valores en *Initial Values*. En *Addr* colocar el valor de 4, debido a que se pretende apuntar al parámetro d003 (durante la comunicación se le resta 1). En *Node* colocar el valor de 2, debido a que este es el Slave-ID que se le asignó al drive.

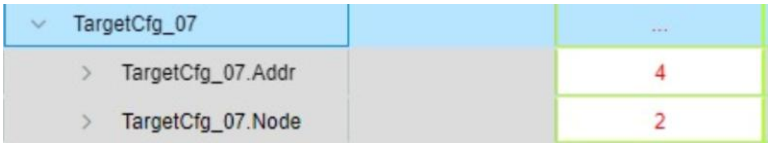


Figura 72. Configuración Parámetro TargetCfg.

- Implementar la siguiente lógica de programación.

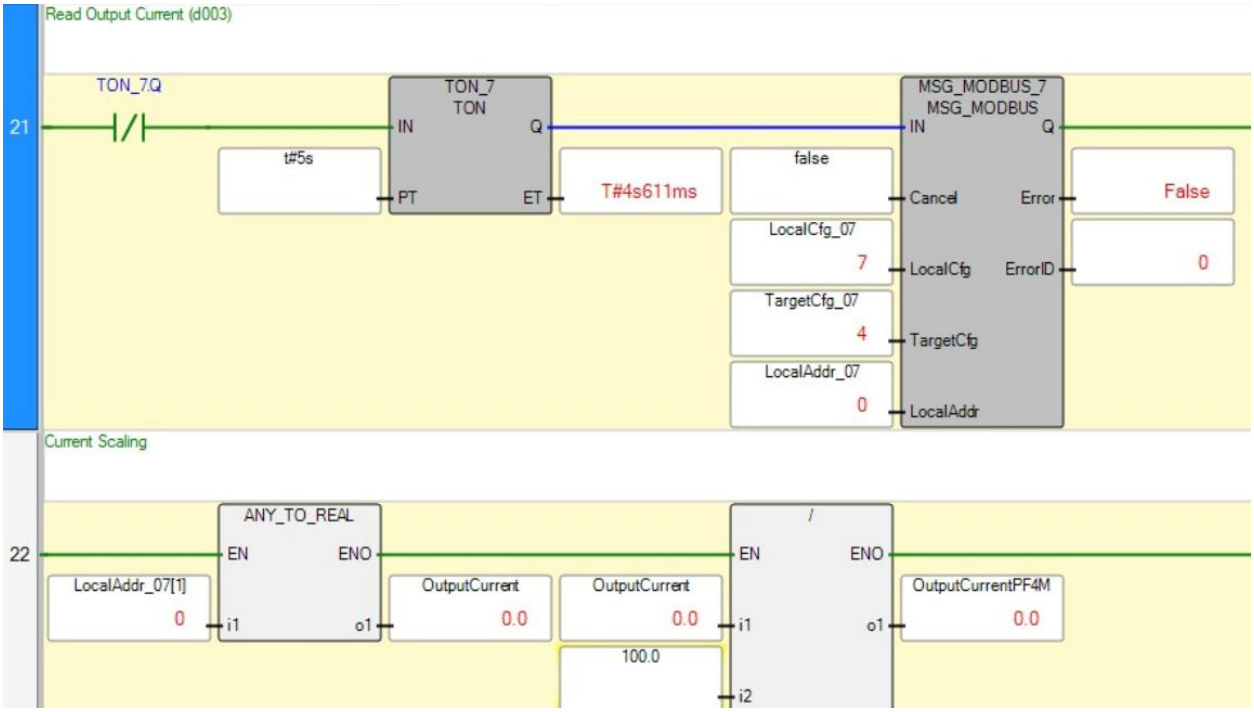


Figura 73. Lógica de Programación Registro 3.

Nota: En esta lógica de programación lo que se añadió inicialmente fue una especie de Clock Memory mediante un TON con autoreinicio, de tal manera que cada 5 segundos se realiza la operación de lectura. Luego, en el siguiente renglón se realiza la conversión del valor obtenido en LocalAddr_07[1] a tipo REAL, luego dicho valor se lo escala por un factor de 0.01 (esto debido al formato interno del drive, donde por ejemplo un valor de 110 equivale a 1.10 A).

5.3. Interfaz Gráfica

- Añadir al Proyecto una interfaz gráfica. Seleccionar PanelView 800 modelo 2711R-T7T. Luego, en *Communication*, configurar *Port* en Ethernet, y verificar que *Protocol* sea Allen-Bradley Micro800 CIP. En *Controller Settings*, verificar que en *Controller Type* sea Micro800, y en *Address* ingresar la dirección IP del controlador.

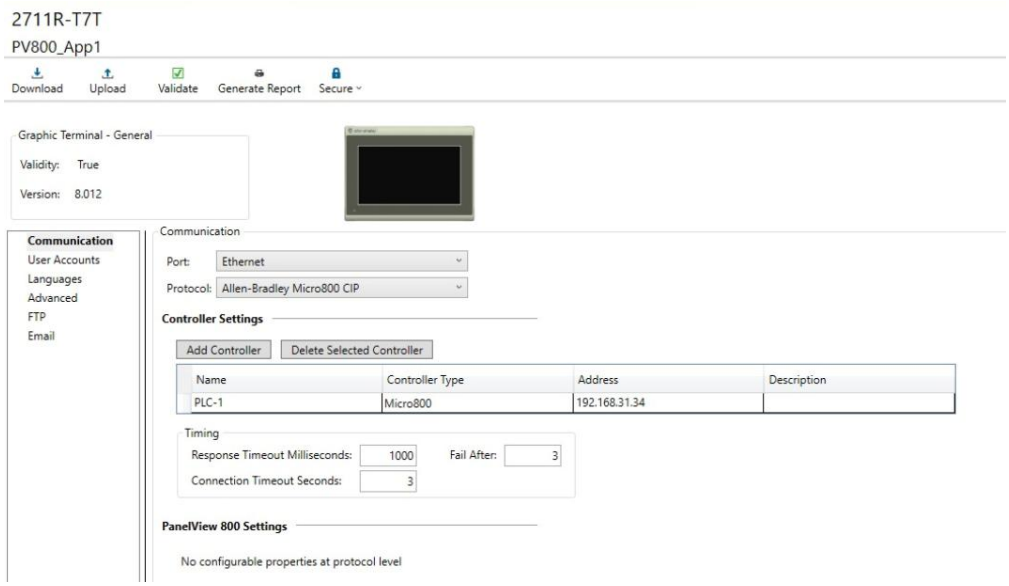


Figura 74. Configuración Ethernet PanelView 800.

- En *Tags*, añadir las correspondientes variables de la interfaz gráfica y vincularlas con sus respectivos alias en el controlador mediante la columna *Address*.

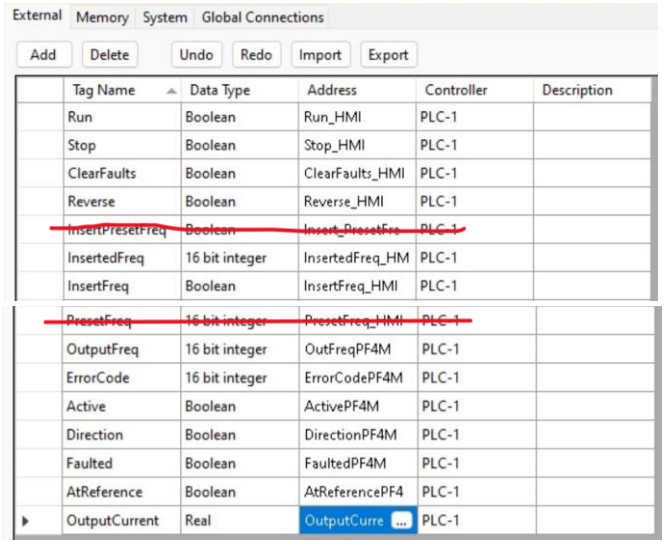


Figura 75. Tags Mapping.

- En Screens, realizar su respectiva interfaz gráfica, vinculando a cada objeto su respectiva variable.

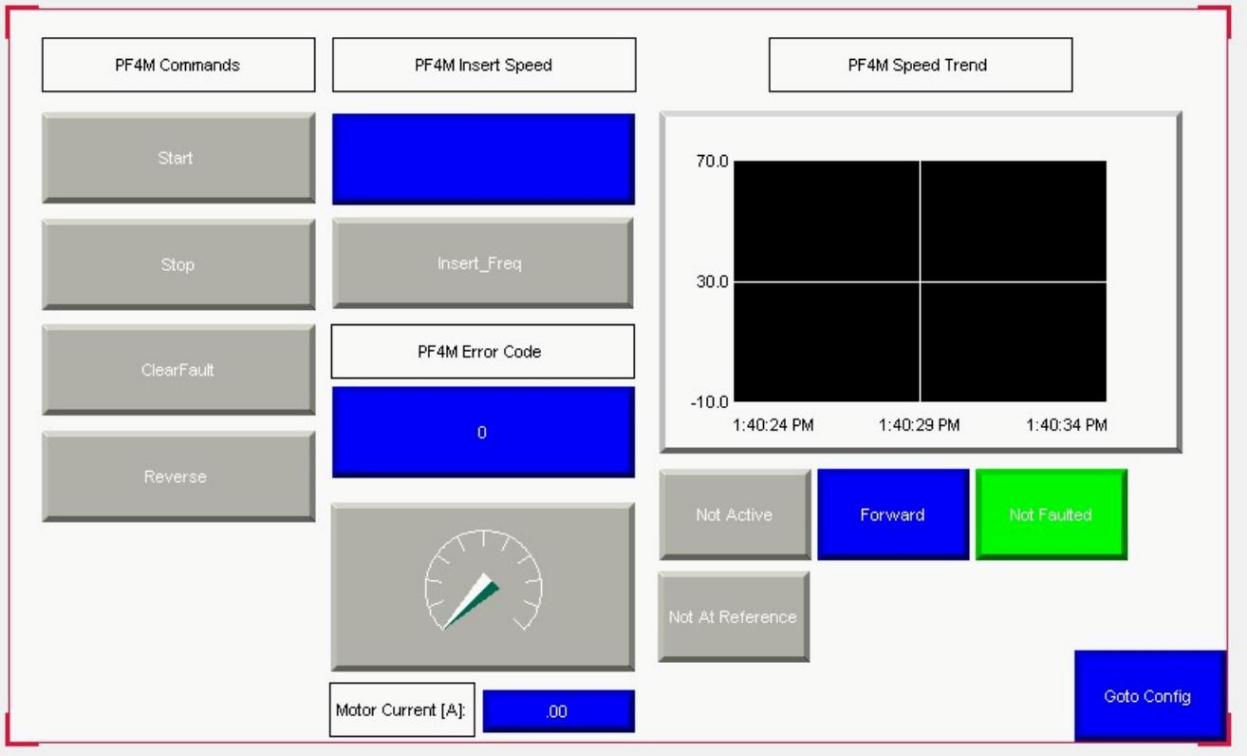


Figura 76. Diseño Interfaz Gráfica.

- Descargar la aplicación a la HMI.
- Finalmente… usted ha logrado implementar la arquitectura propuesta.

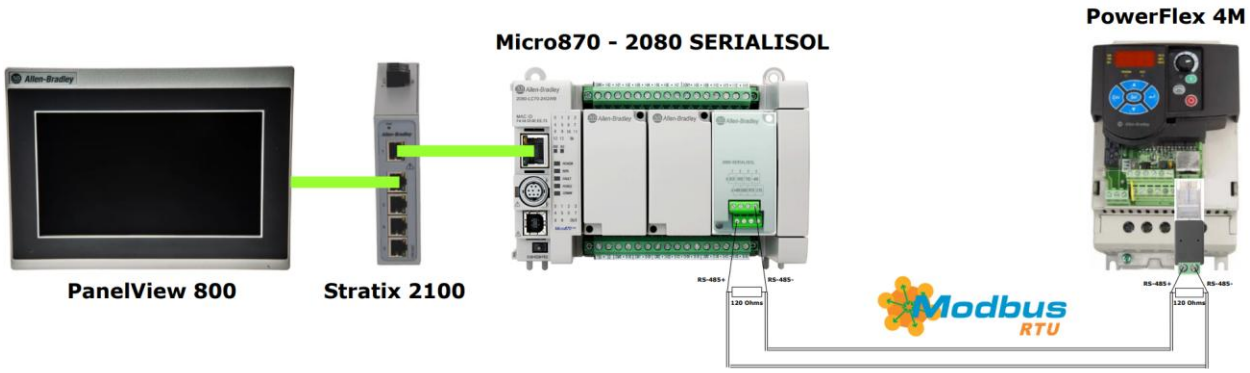


Figura 77. Arquitectura de Comunicación.