Tema 5. Comunicación Serie NO-PROTOCOL. Transmisión y Recepción de datos NX1P2 de OMRON

1.- Introducción

La CPU de los **NX1P2**, puede intercambiar datos con dispositivos que dispongan de un puerto **serie** de uso general, utilizando las instrucciones para **enviar** y **recibir** datos en modo Comunicación Serie **Sin protocolo**.

Esto permitirá, comunicar el PLC, con **cualquier** dispositivo que disponga un puerto serie de comunicaciones como **otros PLC**, **variadores de frecuencia** o incluso microcontroladores como **Arduino** (aunque para comunicar con microcontroladores es necesario utilizar un conversor de niveles de tensión de RS232 a TTL).

Para usar este tipo de comunicación, deberemos utilizar alguna de las tarjetas opcionales (***Option Board***) que el fabricante nos proporciona.

1. **NX1W-CIF01**

Dispone de 9 conectores de abrazadera sin tornillo. Válido para la comunicación serie RS232 que como sabemos, es uno a uno. La distancia máxima recomendables es de 15 metros y soporta comunicaciones Host Link, Modbus RTU y No-Protocol.

2. NX1W-CIF11

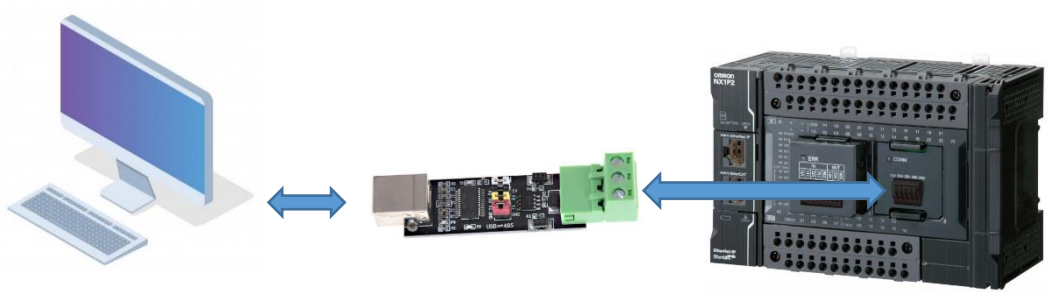
Dispone de 5 conectores de abrazadera sin tornillo. Válido para la comunicación RS422/485 que permite una comunicación de uno a varios. La distancia máxima recomendables es de 50 metros y soporta comunicaciones Host Link, Modbus RTU y No-Protocol.

3. NX1W-CIF12

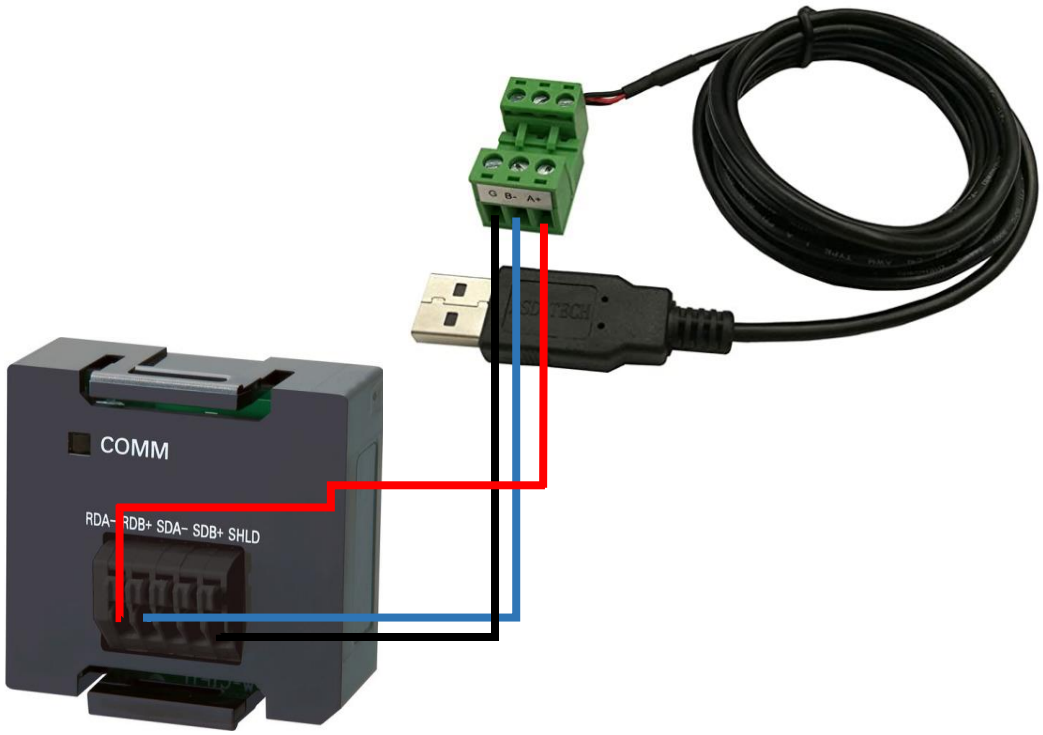
Es idéntico al anterior módulo solo que este, dispone de un aislamiento eléctrico en las líneas de comunicación, lo que le da un nivel más de seguridad.

2.- Cableado entre el PC y el NX1W-CIF11

En este tema vamos a utilizar el **NX1W-CIF11** junto con el PLC **NX1P2** para realizar comunicación serie sin protocolo, entre el **PLC** y una aplicación en **VS.NET** que se ejecutará en nuestro **PC**. Para realizar la conexión entre el ordenador y el PLC, necesitamos utilizar un **conversor** de **USB** a **RS422/485**

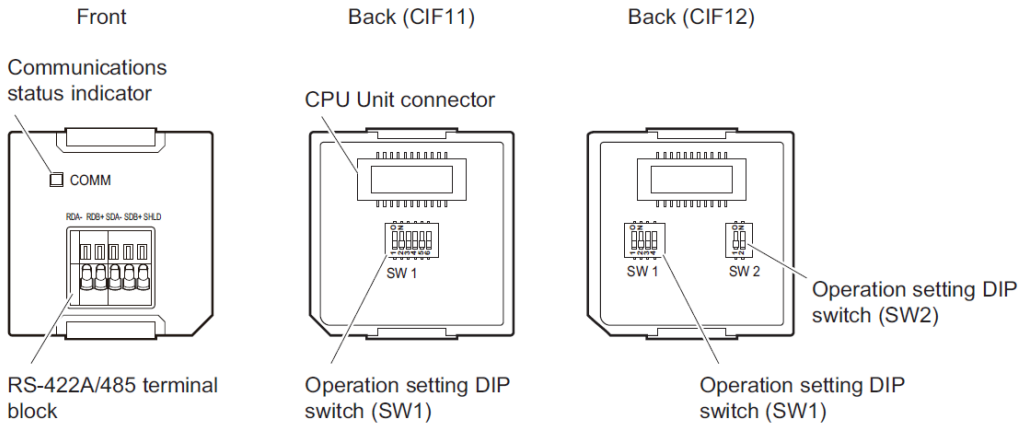


En la siguiente imagen puedes apreciar con exactitud, como son las conexiones:



En el documento ***Especificaciones técnicas y accesorios serie NX1P2.pdf***, en la página 30 podemos ver una descripción de los 5 terminales de este módulo.

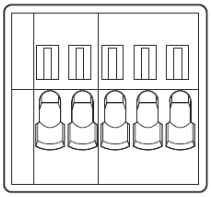
RS-422A/485 Option Board (NX1W-CIF11/NX1W-CIF12)



Note: All pins are turned OFF by default.
Use a narrow-tipped tool such as a flat-blade screwdriver to change the settings of the DIP switches.

RS-422A/485 Terminal Block

RDA- RDB+ SDA- SDB+ SHLD



Abbreviation	Four-wire type selected		Two-wire type selected	
	Signal name	I/O	Signal name	I/O
RDA-	Reception data -	Input	Communication data -	I/O *
RDB+	Reception data +		Communication data +	
SDA-	Transmission data -	Output	Communication data -	I/O *
SDB+	Transmission data +		Communication data +	
SHLD	Shield			

* For two-wire connection, either the RDA-/RDB+ pair or SDA-/SDB+ pair can be used.

El **NX1W-CIF11** dispone de unos conmutadores tipo **switch** para configurar una serie de parámetros referentes a la comunicación. En el documento ***Programación y Comunicaciones.pdf***, en la página 4-24, podemos encontrar una tabla, con la función de cada uno de los anteriores switch:

CIF11		CIF12		Setting	Setting description
SW	No.	SW	No.		
SW1	1	SW1	1	ON	With terminating resistance
	2		2	ON	Two-wire type
	3		3	ON	Two-wire type
	4		4	OFF	(Not used)
	5	SW2	1	ON	With RS control for receive data
	6		2	ON	With RS control for send data

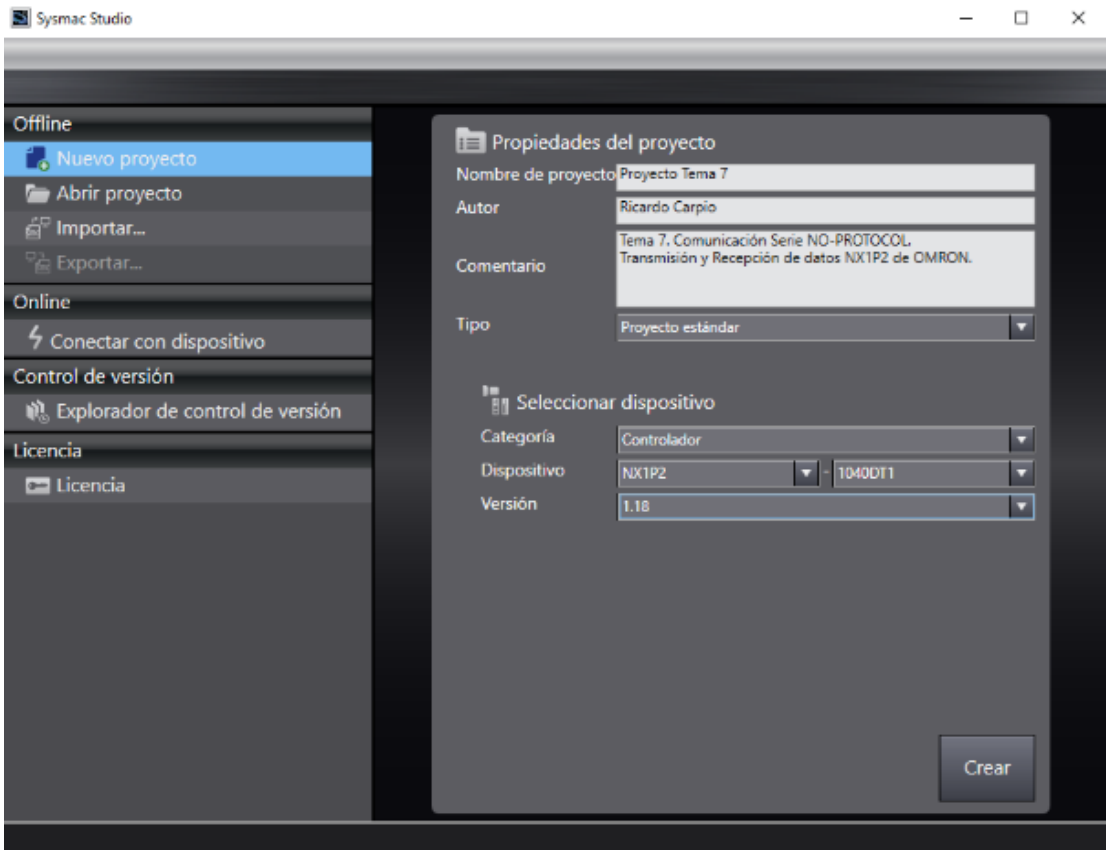
Número SW	Función	Valor utilizado
1	Activa la resistencia limitadora de final de BUS	SI el PLC es el último elemento del BUS, a ON. Si no es el último elemento, a OFF.
2	Comunicación 2 hilos	Si usamos RS485 a ON. Si usamos RS422 a OFF.
3	Comunicación 2 hilos	Si usamos RS485 a ON. Si usamos RS422 a OFF.
4	(sin usar)	(sin usar)
5	Utilización de la señal de control RS, en la recepción de la comunicación.	Si la utilizamos a ON. Si no la utilizamos a OFF.
6	Utilización de la señal de control RS, en el envío de la comunicación.	Si la utilizamos a ON. Si no la utilizamos a OFF.

En nuestro caso, el PLC será el **último** (y único) elemento del bus, usaremos comunicación de **2** hilos **RS485** y activaremos el control RS, tanto para el envío como para la lectura.

Número SW	Valor utilizado
1	**ON**
2	**ON**
3	**ON**
4	**OFF**
5	**ON**
6	**ON**

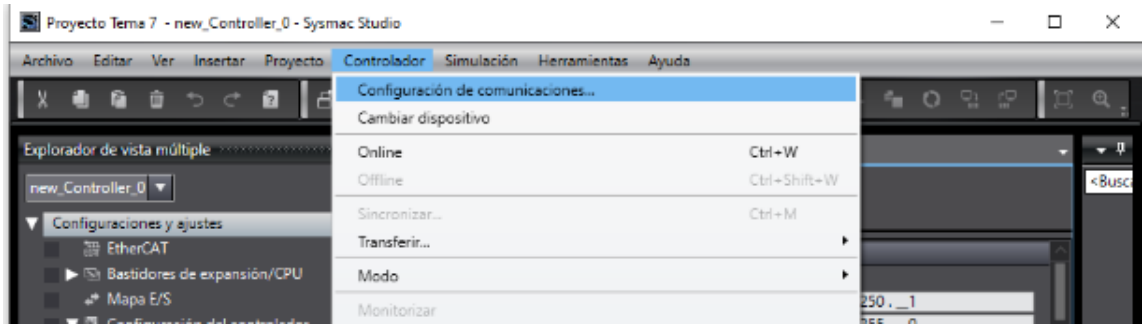
3.- Creación del proyecto en SYSMAC STUDIO

Una vez esté todo perfectamente cableado y configurados los switchs del módulo NX1W-CIF11, crearemos un nuevo proyecto en el SYSMAC STUDIO.

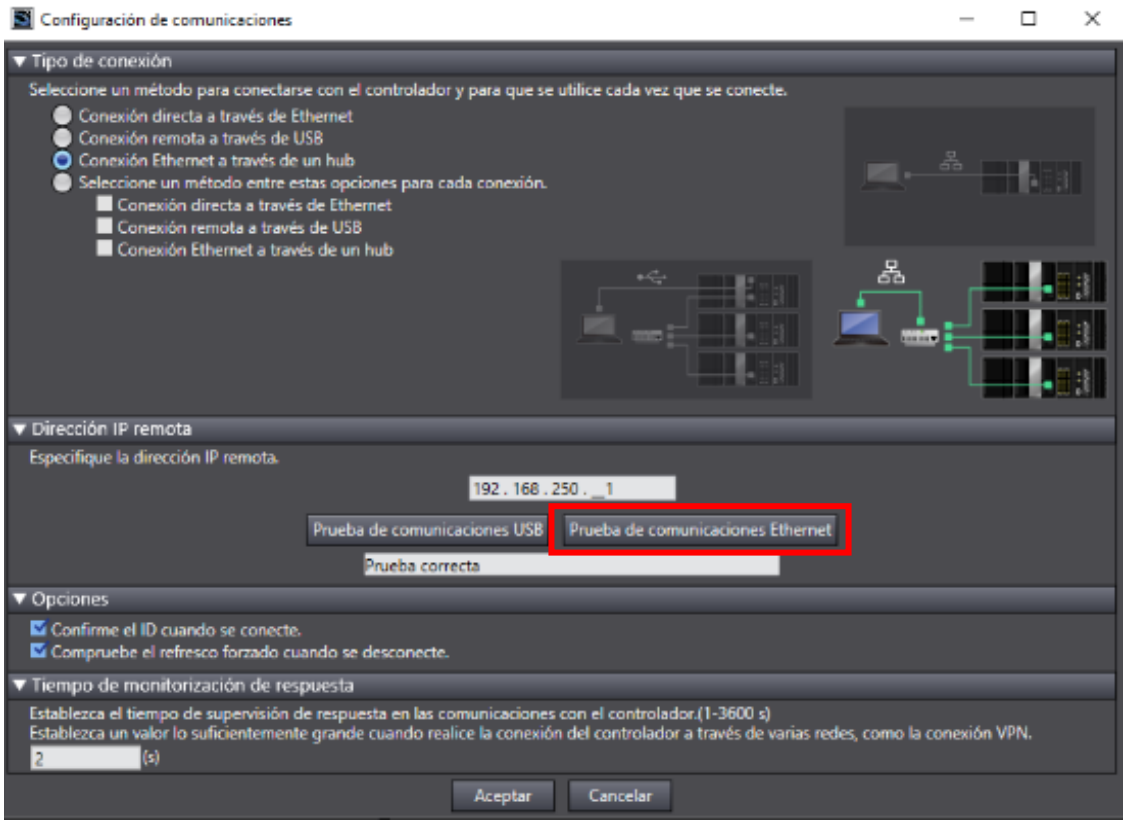


Como por defecto, el PLC tiene la dirección ip 192.168.250.001, pondremos en nuestro ordenador, una ip del mismo rango. Por ejemplo pondremos la 192.168.250.002.

Accederemos al menú ***Controlador*** > ***Configuración de comunicaciones***:

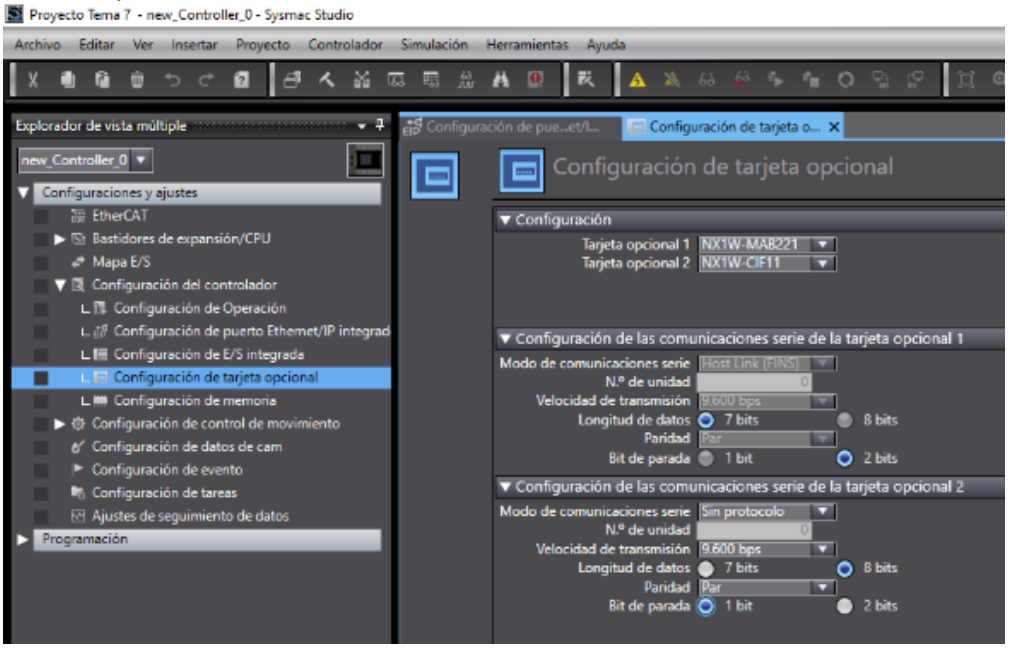


Estableceremos el modo de comunicación a través de un HUB y pondremos la ip del PLC que por defecto es la 192.168.250.001:



Si todo ha ido bien, haremos click en el botón Prueba de comunicaciones Ethernet y nos debe aparecer un mensaje de **prueba correcta**.

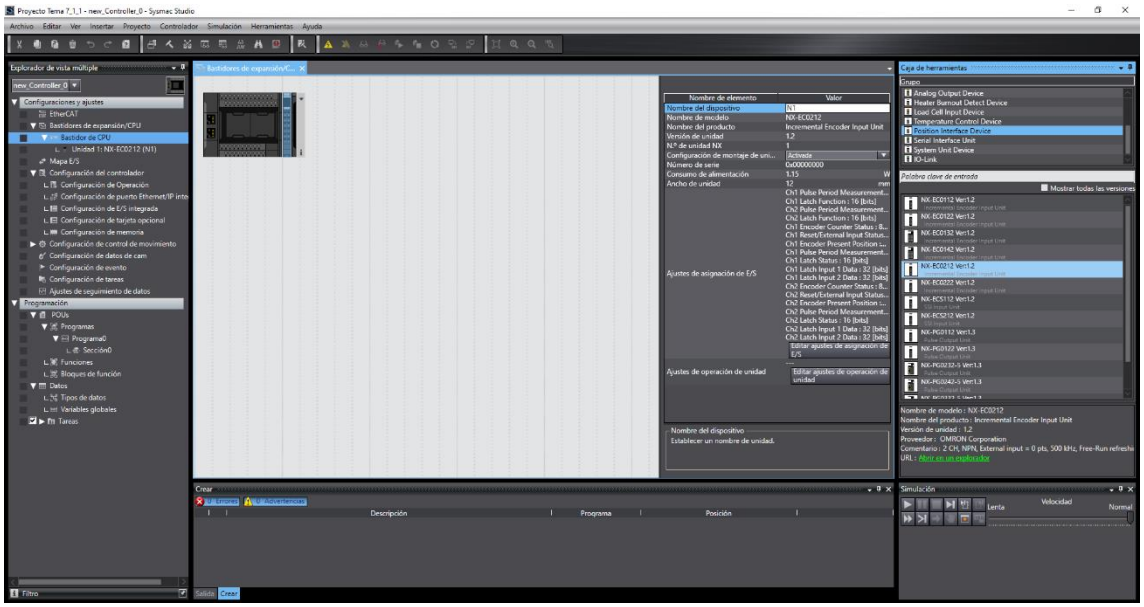
Ahora vamos a configurar las tarjetas opcionales que tengamos en el PLC. En nuestro caso tendremos en el slot 1 la NX1W-MAB221 y en el slot 2, la NX1W-CIF11:



Como vemos en la anterior imagen, configuraremos el NX1W-CIF11 como sigue:

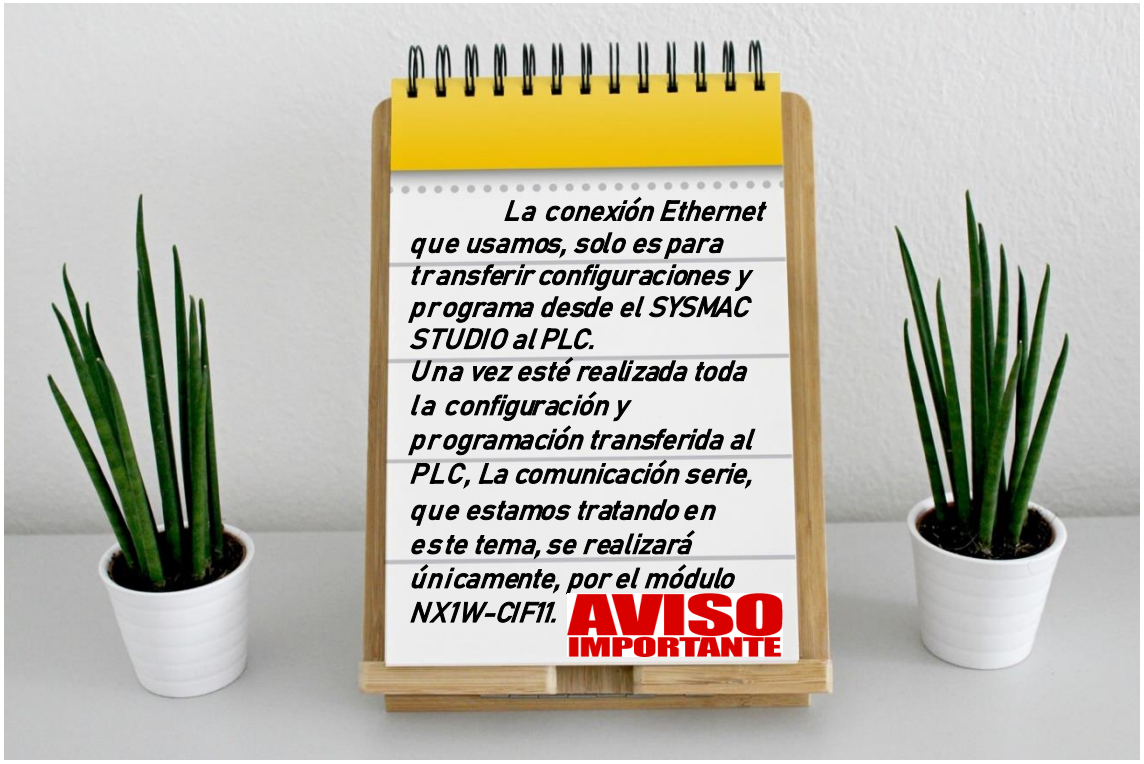
Nombre del Campo	Valor seleccionado
Modo de Comunicación Serie	Sin Protocolo
Número de Unidad	No requerido
Velocidad de transmisión	9600 bps
Longitud de datos	8
Paridad	Par
Bit de parada	1 bit

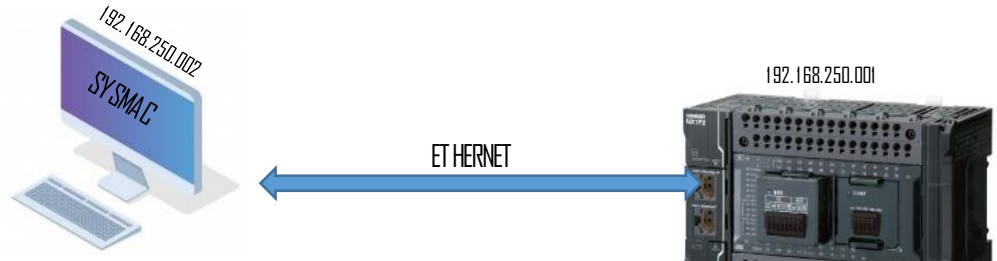
También tenemos en nuestro PLC, en el bastidor de la CPU, una unidad de entrada para encoder incremental. Es la ***NX-EC0212***. Solo tendremos que acceder al menú ***Bastidor de CPU***, seleccionar el grupo ***Position Interface Device***, seleccionar el dispositivo ***NX-EC0212*** y arrastrarlo hasta la CPU. Se colocará entre la CPU y el terminal de finalización que llevan todos los PLC de esta gama.



Con esto ya tenemos configurado en SYSMAC STUDIO, nuestro PLC, tal y como lo tenemos en la realidad.

Antes de empezar con la programación en el NX1P2, tenemos que tener claro una cuestión referida a las conexiones.





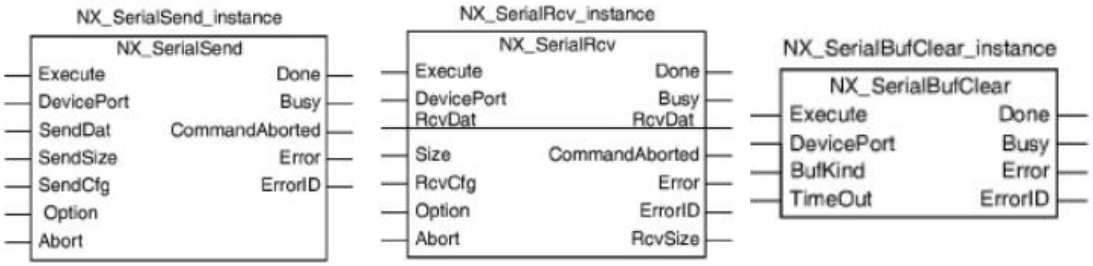
4.- Instrucciones utilizadas en la comunicación Serie sin protocolo.

En el documento ***Programación y Comunicaciones.pdf***, en la página 4-29, podemos encontrar un listado de funciones para la comunicación serie sin protocolo:

Instruction	Name	Outline of function
NX_SerialSend	Send No-protocol Data	Sends data in No-Protocol mode from a serial port on a CIF Unit or Option Board.
NX_SerialRcv	Receive No-protocol Data	Reads data in No-Protocol Mode from a serial port on a CIF Unit or Option Board.
NX_SerialSigCtl	Serial Control Signal ON/OFF Switching	Turns ON or OFF the ER or RS signal of a serial port on a CIF Unit or Option Board.
NX_SerialSigRead	Read Serial Control Signal	Reads the CS or DR signal of a serial port on an Option Board.
NX_SerialStatus-Read	Read Serial Port Status	Reads the status of a serial port on an Option Board.
NX_SerialBufClear	Clear Buffer	Clears the send or receive buffer.

- ***NX_SerialSend***: Para enviar datos por el puerto serie utilizado.
- ***NX_SerialRcv***: Para recibir datos por el puerto serie utilizado.
- ***NX_SerialSigCtl***: Para activar o desactivar la línea de control ER o RS del puerto serie, en caso de utilizarlas.
- ***NX_SerialSigRead***: Para leer las líneas de control del puerto serie CS o DR, en caso de utilizarlas.
- **NX_SerialStatusRead**: Lee el estado del puerto serie que estemos utilizando.
- ***NX_SerialBufClear:*** Borra el buffer de memoria de entrada o salida del puerto serie que estemos utilizando

En este tema, veremos cómo se utilizan las 2 primeras y la última, es decir, estudiaremos las instrucciones ***NX_SerialSend***, ***NX_SerialRcv*** y ***NX_SerialBufClear***.

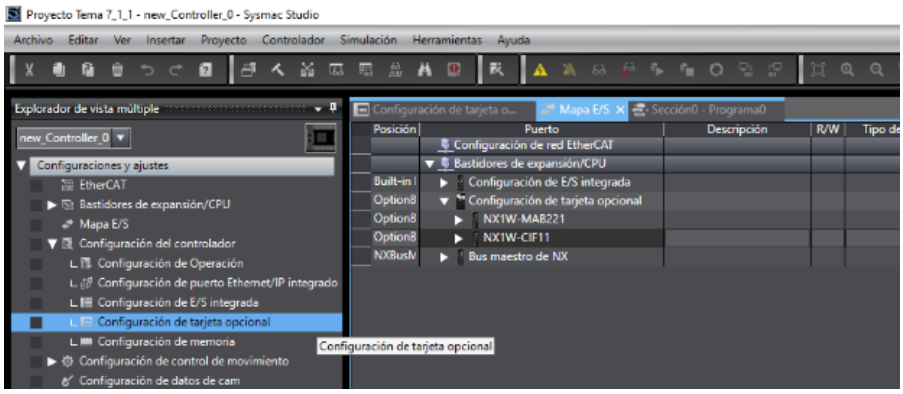


Observamos que las dos primeras entradas, ***Execute*** y ***DevicePort***, son idénticas en las tres instrucciones. Execute es simplemente la condición de ejecución de la instrucción y ***DevicePort***, merece una explicación más detallada que veremos en el siguiente punto.

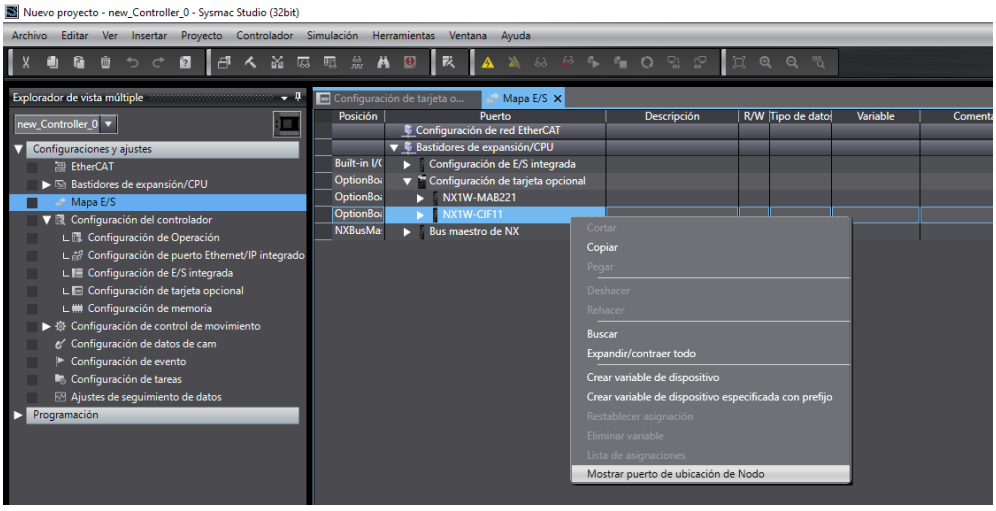
5.- Creación del parámetro DevicePort.

Para utilizar nuestro módulo opcional como un puerto de comunicaciones, debemos crear una variable que represente al módulo en nuestro código. El procedimiento para crear dicha variable es el siguiente:

- Hacemos click en la sección de ***Mapa de E/S***



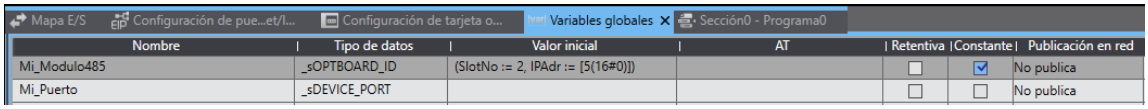
- Marcamos el módulo NX1W-CIF11 con un click y cuando esté marcado, botón derecho y ***Mostrar puerto de ubicación de Nodo.***



- Se ha creado una variable de lectura, del tipo ***_sOPTBOARD_ID***, al que le debemos dar un nombre como por ejemplo ***Mi_Modulo485***.

Podemos observar que se ha creado una variable global de tipo ***_sOPTBOARD_ID***, que representa la ubicación del módulo que estamos utilizando.

Debemos crear una variable más para poder utilizar el módulo como un puerto de comunicaciones. Se trata de una variable de tipo **_sDevicePort**, a la que llamaremos ***Mi_puerto***.

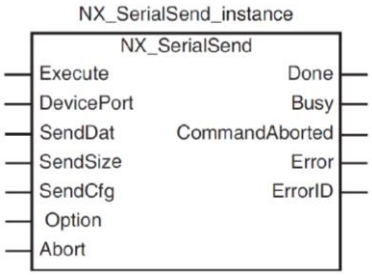


Nombre	Tipo de datos	Valor inicial	AT	Retentiva	Constante	Publicación en red
Mi_Modulo485	_sOPTBOARD_ID	(SlotNo := 2, IPAdr := [5(16#0)])		☐	☑	No publica
Mi_Puerto	_sDEVICE_PORT			☐	☐	No publica

Es decir, debemos tener una variable (***Mi_Modulo485***) asociada al ***módulo*** opcional NX1W_CIF11 insertada en el Slot 2, y otra variable (***Mi_puerto***) asociada al puerto que vamos a crear con la ayuda de la primera variable.

Estas dos variables, las deberemos crear una sola vez aunque usemos las tres instrucciones (la de envío, la de recepción y la de borrar los buffers de memoria)

6.- Instrucción para el envío de datos NX_SerialSend.



Execute: *BOOL*
Parámetro de entrada de ejecución. Cuando recibe el flanco ascendente ejecuta la operación del FB.

DevicePort: *_sDEVICE_PORT*
Parámetro de entrada para la localización y definición del puerto serie que vamos a utilizar.
Para generarlo, es necesario crear antes, la variable _sOPTBOARD_ID.

SendDat: ARRAY[0..X] OF BYTE
Parámetro de entrada en el que se deben introducir los datos de envío des del buffer del módulo. Se debe declarar como una estructura de máximo 256 bytes. La dimensión de la trama dependerá de la declaración de la siguiente entrada, que indicará cuantos elementos del array se concatenaran.

SendSize: UINT
Variable de entrada (entero sin signo) que indica la longitud de la trama en bytes.
Su valor máximo es 4096. Una trama con mayor número de bytes no puede ser ni enviada ni recibida.

: _sSERIAL_CFG (Se puede omitir)
Parámetro de entrada que nos permite configurar la gestión de la trama a escribir; código de inicio, código de fin, tamaño de trama…
Es una estructura que contiene los siguientes miembros:

Nombre	Definición	Formato	Valores
StartTrig	Config. código de inicio	*_eSERIAL_START*	*_SERIAL_START_NONE* (ninguno)
			_SERIAL_START_STARTCODE1 (1byte)
			_SERIAL_START_STARTCODE2 (2bytes)
StartCode		*BYTE[2]*	---
EndTrig	Config. código de fin	*_eSERIAL_END*	*_SERIAL_START_NONE* (ninguno)
			_SERIAL_END_ENDCODE1 (1byte)
			_SERIAL_END_ENDCODE2 (2bytes)
			_SERIAL_END_TERMINATION_CAHR (condición de terminación) [1]
			_SERIAL_END_RCV_SIZE (por tamaño) [1]
EndCode		*BYTE[2]*	---

Aunque este parámetro no acostumbra a modificarse para envíos estándar, puede ser necesario para la recepción de tramas específicas.

Option: *_sSERIAL_SEND_OPTION* (Se puede omitir)

Variable de entrada que permite ajustar algunos parámetros concretos del envío o la recepción. También es una estructura formada únicamente por el siguiente miembro:

Nombre	Definición	Formato	Valores
SendDelay	Retraso al envío (s)	*UINT*	---

Abort: *BOOL* (Se puede omitir)

Parámetro de entrada para la interrupción de ejecución del FB. Solo será útil si la salida *Busy* está activada.

Done: *BOOL* (Se puede omitir)

Parámetro de salida que indica que el proceso de envío se ha completado o se ha abortado con la entrada *Abort*.

Busy: *BOOL* (Se puede omitir)

Parámetro de salida que indica que el FB está ejecutándose.

CommandAborted: *BOOL* (Se puede omitir)

Parámetro de salida que indica que el FB se ha interrumpido (normalmente por activación de la entrada *Abort*).

Error: *BOOL* (Se puede omitir)

Parámetro de salida que indica que la instrucción no se ha podido ejecutar con éxito.

ErrorID: *WORD* (Se puede omitir)

Variable de salida que indica la causa del error mediante un código en hexadecimal. Para descifrar el significado de cada código, se debe consultar la siguiente tabla:

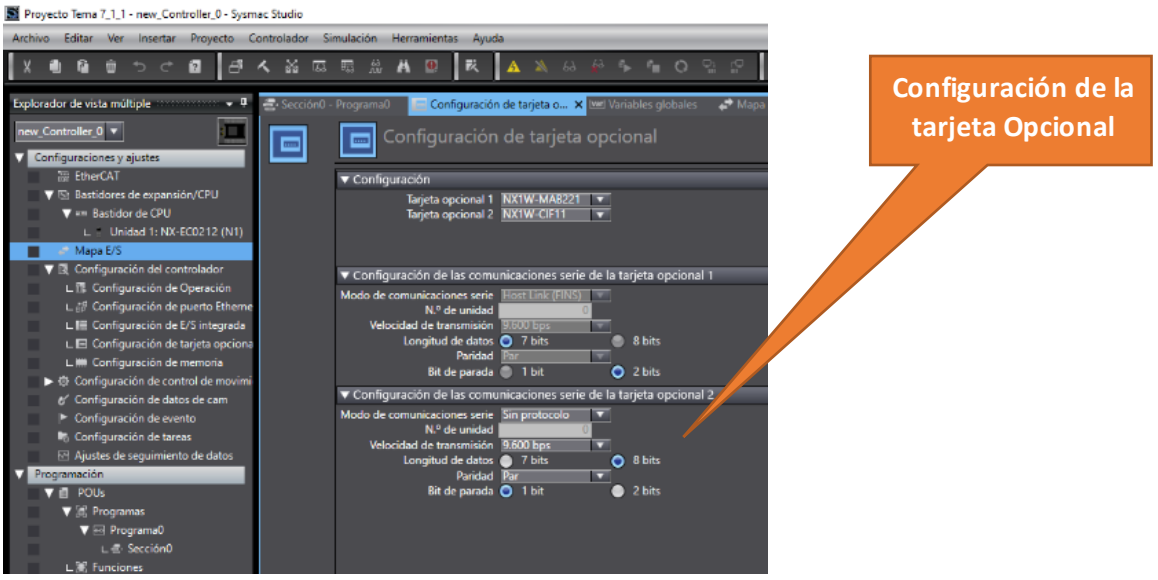
Errores *NX_SerialSend*	
Código	**Definición**
16#0400	Valor de entrada fuera de rango
16#0406	Posición ilegal de datos
16#040D	Unidad ilegal especificada
16#0419	Tipo de dato incorrecto
16#041D	Demasiadas instrucciones ejecutadas simultáneamente
16#0C04	Multi-ejecución en el puerto
16#0C0C	Instrucción ejecutada en puerto no disponible
16#0C0D	Unidad inicializándose

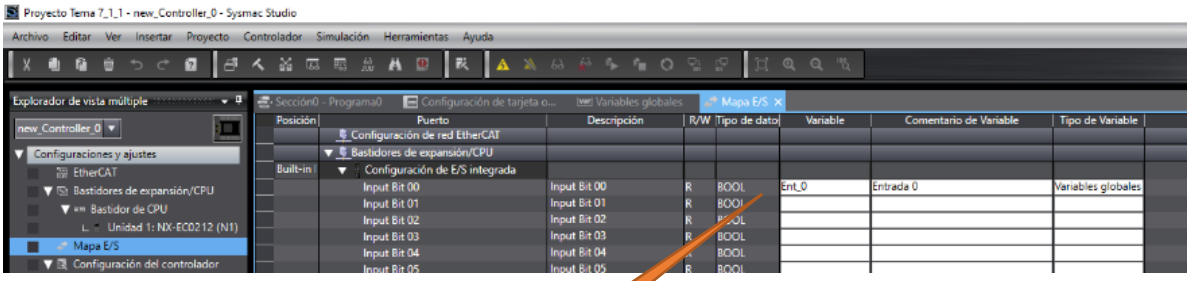
Errores *NX_SerialRcv*	
Código	**Definición**
16#0400	Valor de entrada fuera de rango
16#0406	Posición ilegal de datos
16#0407	Rango de datos superado
16#040D	Unidad ilegal especificada
16#0419	Tipo de dato incorrecto
16#041D	Demasiadas instrucciones ejecutadas simultáneamente
16#0C03	Buffer de recepción lleno
16#0C04	Multi-ejecución en el puerto
16#0C05	Error de paridad
16#0C06	Error de configuración puerto serie
16#0C07	Error de desbordamiento
16#0C0B	Timeout
16#0C0C	Instrucción ejecutada en puerto no disponible
16#0C0D	Unidad inicializándose

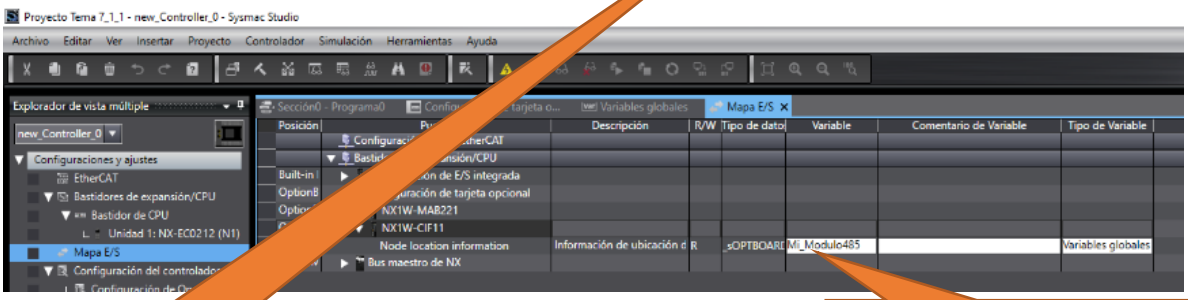
Veamos un pequeño ejemplo con el uso de esta instrucción.

Enunciado

*Conectar el PLC, a través de un módulo conversor de USB a RS485, al PLC NX1P2. El PLC, deberá enviar dos bytes con los valores **16#22** y **16#33**, cuando se produzca un flanco de subida en la **entrada 0** de nuestro PLC. Usaremos una Aplicación en VS.NET para visualizar los dos datos anteriores.*







Variable Entrada 0 del PLC

Variable De localización del módulo

Variables Globales del Proyecto

Sección0 - Programa0 | Configuración de... | Variables globales | Mapa E/S

Nombre	Tipo de datos	Valor inicial	AT	Retentiva	Constante	Publicación en red
Mi_Modulo485	_sOPTBOARD_ID	(SlotNo := 2, IPAdr := [5(16#0)])		☐	☑	No publica
Mi_Puerto	_sDEVICE_PORT			☐	☐	No publica
Ent_0	BOOL		BuiltInIO://cpu/#0/Input_Bit_00	☐	☐	No publica
Dato	ARRAY[0..1] OF BYTE	[16#22, 16#33]		☐	☑	No publica
Numero_bytes	UINT	2		☐	☐	No publica

Código del Proyecto

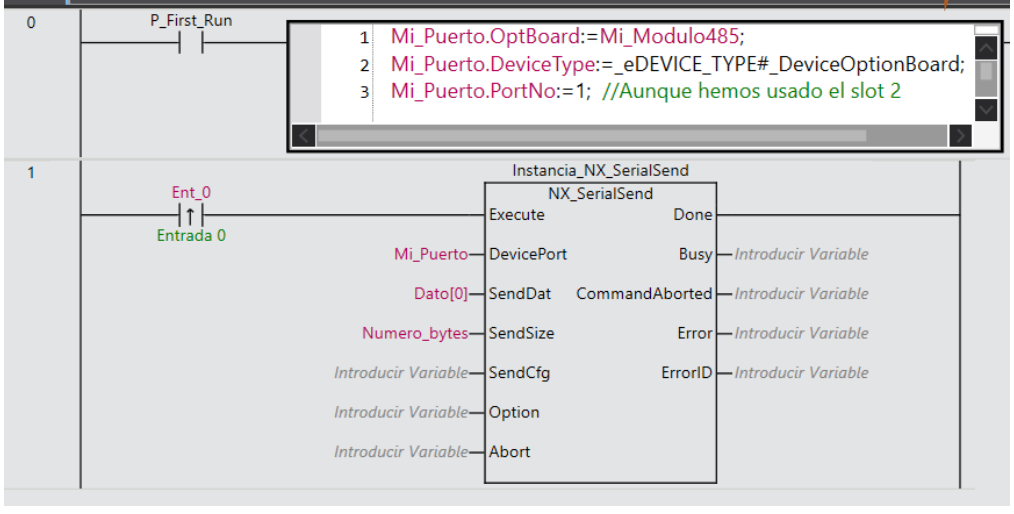
Sección0 - Programa0 | Configuración de tarjeta o... | Variables globales | Mapa E/S

Variables

Namespace - Uso de

Internas / Externas

Nombre	Tipo de datos	Constante	Comentario
Ent_0	BOOL	☐	
Mi_Modulo485	_sOPTBOARD_ID	☑	
Mi_Puerto	_sDEVICE_PORT	☐	
Dato	ARRAY[0..1] OF BYTE	☑	
Numero_bytes	UINT	☐	

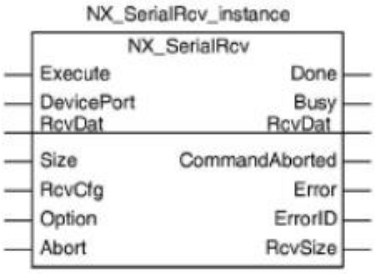


En cuanto a la aplicación de VS.NET, sólo veremos Los procedimientos relacionados con la lectura:

```
0 referencias
Private Sub SerialPort_DataReceived(sender As Object, e As SerialDataReceivedEventArgs) Handles SerialPort.DataReceived
    'Defino el tamaño de la matriz de lectura, con el número de bytes
    'presentes en el buffer de entrada, en el momento de la lectura
    Trama_Entrante = New Byte(SerialPort.BytesToRead - 1) {}
    'leemos la trama recibida y la guardamos en la matriz Trama_Entrante.
    SerialPort.Read(Trama_Entrante, 0, 2)
    Me.Invoke(New EventHandler(AddressOf modificar))
    'limpieza de buffers de entrada y salida
    SerialPort.DiscardInBuffer()
    SerialPort.DiscardOutBuffer()
End Sub
```

```
1 referencia
Public Sub modificar()
    Beep()
    'Trama_Saliente(2) = Convert.ToByte(byte_alt_string, 16)
    TextBox_Trama_Recibida0.Text = Convert.ToString(Trama_Entrante(0), 16)
    TextBox_Trama_Recibida1.Text = Convert.ToString(Trama_Entrante(1), 16)
End Sub
```

7.- Instrucción para la lectura de datos NX_SerialRcv.



Execute y DevicePort: *Idénticos al bloque anterior NX_SerialSend*

RcvDat: Idéntico al parámetro ***SendDat*** del bloque anterior. En esta ocasión, este array de BYTES, se utilizará para almacenar los bytes que se lean. En este bloque es un parámetro presente en la entrada y en la salida.

Size: UINT
Variable de entrada (entero sin signo) que indica la longitud de la matriz en bytes, donde se va a almacenar la lectura. Su valor máximo es 4096. Una trama con mayor número de bytes no puede ser ni enviada ni recibida.

RcvCfg: Idéntico al parámetro ***SendCfg*** del bloque anterior.

Option: *_sSERIAL_RCV_OPTION* (Se puede omitir)
Variable de entrada que permite ajustar algunos parámetros concretos del envío o la recepción. También es una estructura formada por los siguientes miembros:

NX_SerialRcv: _sSERIAL_RCV_OPTION			
Nombre	**Definición**	**Formato**	**Valores**
TimeOut	Tiempo para lanzar error (s)	*UINT*	--- 0 = indefinido
ClearBuf	Condición para limpiar buffer	*BOOL*	---

Abort, Done y Busy: Idénticos al bloque anterior *NX_SerialSend.*

CommandAborted, Error y ErrorID: Idénticos al bloque anterior *NX_SerialSend.*

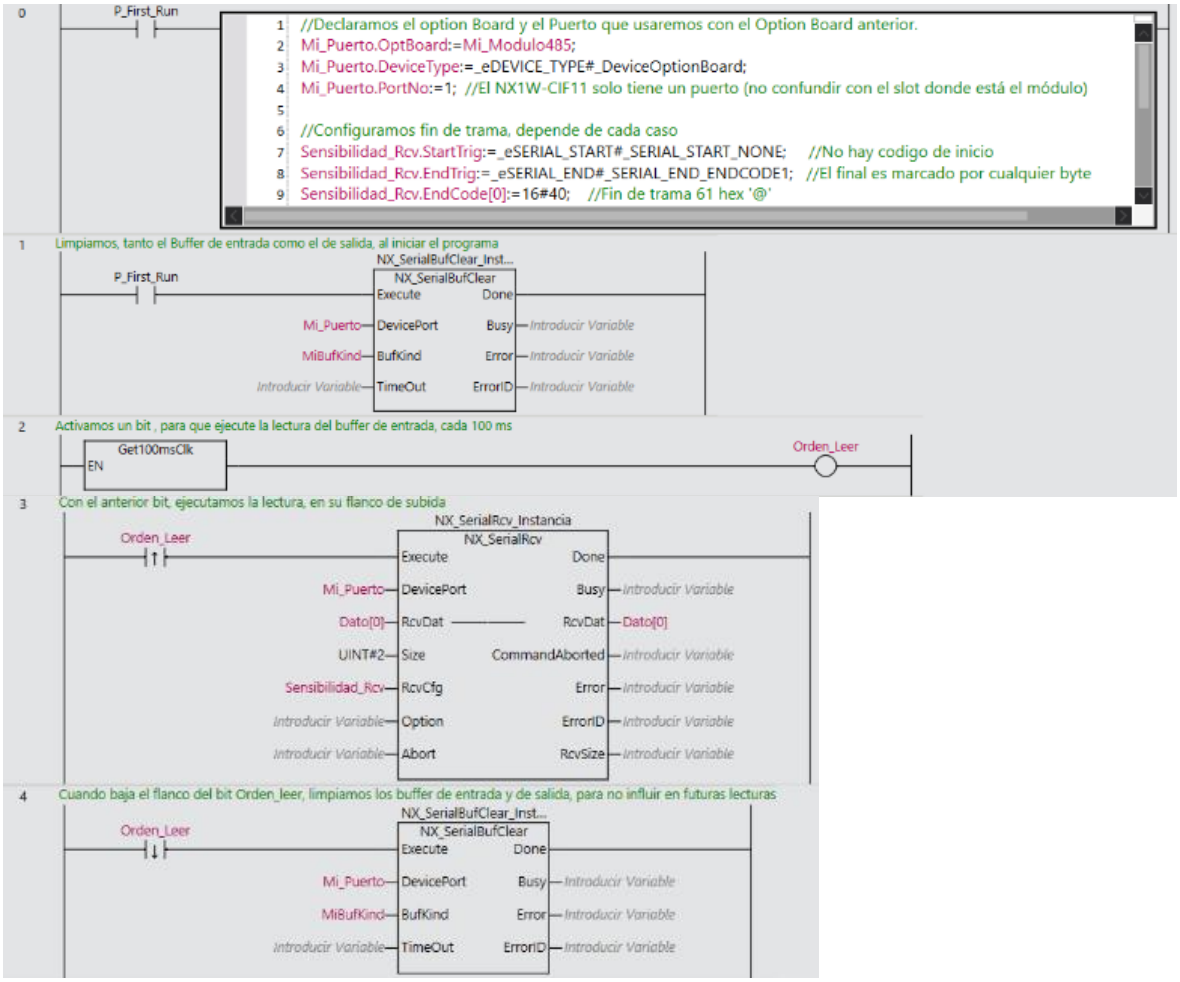
RcvSize: UINT
Variable de salida (entero sin signo) que indica la longitud en bytes, que se ha leído.

Veamos un pequeño ejemplo con el uso de esta instrucción.

Enunciado

Conectar el PLC, a través de un módulo conversor de USB a RS485, al PLC NX1P2. El PLC, deberá activar la salida 0 cuando desde una aplicación VS .NET, se envíen los bytes 34 y F2. Cuando desde la anterior aplicación, se envíen los bytes, 25 y 31, el PLC deberá desactivar la misma salida. Enviaremos un byte '@' (16#40) a modo de finalizador de trama.

En el programa del PLC, tendremos 2 secciones, una para la lectura y otra para la comparación de los bytes leídos y control de la salida. A continuación, podemos observar la sección de lectura:



La sección salida, la puedes observar a continuación, así como las variables globales utilizadas en todo el programa. Esta sección, estará hecha solo con un bloque de texto estructurado.

```
0
1  //En función de los bytes guardados en el array Dato, despues de compararlos
2  //con los bytes guardados en los array Valor_Sal_0_ON y Valor_Sal_0_OFF
3  //activamos o no la salida 0 del PLC
4  IF (Dato[0]=Valor_Sal_0_ON[0]) & (Dato[1]=Valor_Sal_0_ON[1]) then
5      Sal_0 :=TRUE;
6  END_IF;
7  IF (Dato[0]=Valor_Sal_0_OFF[0]) & (Dato[1]=Valor_Sal_0_OFF[1]) then
8      Sal_0 :=FALSE;
9  END_IF;
```

Salida - Programa0 | Recepcion - Programa0 | Variables globales

Nombre	Tipo de datos	Valor inicial	AT	Retentiva	Constante	Publicación en red
Mi_Modulo485	_sOPTBOARD_ID	(SlotNo := 2, IPAdr := [5(16#0)])		☐	☑	No publica
Mi_Puerto	_sDEVICE_PORT			☐	☐	No publica
MiBufKind	_eSERIAL_BUF_KIND	_BUF_SENDRCV		☐	☐	No publica
Sensibilidad_Rcv	_sSERIAL_CFG			☐	☐	No publica
Orden_Leer	BOOL			☐	☐	No publica
Dato	Array[0..1] of BYTE			☐	☐	No publica
Valor_Sal_0_ON	ARRAY[0..1] OF BYTE	[16#34, 16#F2]		☐	☐	No publica
Valor_Sal_0_OFF	ARRAY[0..1] OF BYTE	[16#25, 16#31]		☐	☐	No publica
Sal_0	BOOL		BuiltInIO://cpu/#0/Output_Bit_00	☐	☐	Solo publicar

Fijémonos que el array Dato, está formado por 2 bytes, igual que los arrays Valor_Sal_0_ON y Valor_Sal_0_OFF. En estos dos últimos bytes, guardaremos los valores especificados en el enunciado:

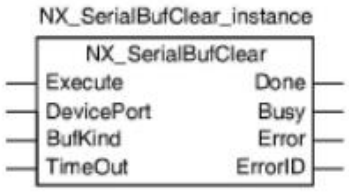
Array	Valor (Hexadecimal)
Valor_Sal_0_ON[0]	34
Valor_Sal_0_ON[1]	F2
Valor_Sal_0_OFF[0]	25
Valor_Sal_0_OFF[1]	31

En cuanto a la aplicación de VS.NET, sólo veremos Los procedimientos relacionados con los envíos para activar o desactivar la salida del PLC:

```
Private Sub Button_salida0_ON_Click(sender As Object, e As EventArgs) Handles Button_salida0_ON.Click
    'Si el puerto está abierto
    If SerialPort.IsOpen Then
        'Voy a intentar enviar la trama de activar la salida 0 del PLC.
        Try
            Dim Trama_Saliente As Byte() 'Declaro la matriz de bytes.
            Trama_Saliente = New Byte(2) {} 'Asigno a la matriz 3 posiciones
            'Asigno un valor a cada byte:
            'La trama es:  34 F2 '@'  >>> 34 y F2 son los dos bytes que vamos a trasnmitir y 40 es la terminación
            Trama_Saliente(0) = &H34
            Trama_Saliente(1) = &HF2
            Trama_Saliente(2) = &H40 ' el &H40 representa el caracter '@'
            'Enviamos la trama de bytes por el puerto. Desde el componente 0 de la matriz
            'hasta el último componente de la matriz.
            SerialPort.Write(Trama_Saliente, 0, Trama_Saliente.Length) 'Enviar trama
        Catch ex As Exception
            MessageBox.Show(Me, ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End If
End Sub
```

```
Private Sub Button_salida0_OFF_Click(sender As Object, e As EventArgs) Handles Button_salida0_OFF.Click
    'Si el puerto está abierto
    If SerialPort.IsOpen Then
        'Voy a intentar enviar la trama de desactivar la salida 0 del PLC.
        Try
            Dim Trama_Saliente As Byte() 'Declaro la matriz de bytes.
            Trama_Saliente = New Byte(2) {} 'Asigno a la matriz 3 posiciones
            'Asigno un valor a cada byte:
            'La trama es:  25 31 '@'  >>> 25 y 31 son los dos bytes que vamos a trasnmitir y 40 es la terminación
            Trama_Saliente(0) = &H25
            Trama_Saliente(1) = &H31
            Trama_Saliente(2) = &H40 ' el &H40 representa el caracter '@'
            'Enviamos la trama de bytes por el puerto. Desde el componente 0 de la matriz
            'hasta el último componente de la matriz.
            SerialPort.Write(Trama_Saliente, 0, Trama_Saliente.Length) 'Enviar trama
        Catch ex As Exception
            MessageBox.Show(Me, ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End If
End Sub
```

8.- Instrucción para el borrado de los buffers del puerto NX_SerialBufClear.



Como vimos en el tema de la comunicación serie, después de leer y después de enviar, es conveniente hacer una limpieza de los buffers del puerto, para eliminar posibles bytes, que puedan entorpecer, operaciones futuras tanto de lectura como de escritura.

Los dos primeros parámetros, ya los hemos visto anteriormente. El tercer parámetro ***BufKind***, es una variable, de tipo ***_eSERIAL_BUF_KIND***, que determina, que buffer se debe borrar: el de entrada, el de salida o los dos.

Es un parámetro opcional, es decir, podemos no poner nada y por defecto, borrará los dos buffers. Pero si en alguna ocasión, queremos borrar un buffer específico, los valores de la variable admitidos son:

_BUF_SENDRCV

_BUF_SEND

_BUF_RCV