Slicetex Ladder Designer Studio

(StxLadder)

Componentes Ladder

Autor: Ing. Boris Estudiez

1 Descripción General

El presente documento lista los componentes gráficos Ladder disponibles en el software ***Slicetex Ladder Designer Studio (StxLadder)*** *utilizado para programar los PLC de Slicetex Electronics.*

El documento es solo una referencia para el programador y permite conocer rápidamente los recursos Ladder del software. Puede imprimir este documento para utilizarlo como referencia rápida.

Se recomienda acceder a la descripción de todos los componentes Ladder desde el entorno de programación, ya que dicha información se actualiza con mayor frecuencia.

En este documento no se describen las funciones disponibles del Lenguaje Pawn.

2 Lecturas Recomendadas

Antes de leer este documento, recomendamos que se familiarice con el software StxLadder y el PLC adquirido. Sugerimos leer los siguientes documentos:

1. Manual de Usuario del software StxLadder.
2. Hoja de datos técnicos del PLC.

Mas documentación puede encontrar en la página del producto: www.slicetex.com.

Para consultas y soporte, ponemos a disposición un foro de discusión en: www.slicetex.com/foro donde puede leer preguntas de otros usuarios y realizar también sus propias preguntas.

3 Alcance del Documento

El presente documento contiene las descripciones de todos los componentes Ladder soportados por el entorno StxLadder.

La finalidad del documento es introducirlo rápidamente al adecuado uso y selección de componentes Ladder para realizar sus proyectos. Se recomienda imprimirlo para que pueda utilizarlo como referencia

Este documento complementa al Manual de Usuario del entorno StxLadder, por lo que que deberá comprender primero los conceptos allí vertidos antes de continuar la lectura.

Note que aquí se listan todos los componentes soportados por el entorno StxLadder, por lo tanto muchos de los componentes descriptos pueden no estar soportado por su modelo de PLC.

La estructura del documento es por capítulos, donde cada capitulo agrupa componentes similares.

Puede descargar ejemplos completos desde nuestro sitio Web para comprender el uso de los componentes Ladder y consolidar la información de este texto.

Finalmente, recomendamos acceder a la descripción de los componentes Ladder desde el entorno StxLadder debido a que se actualiza dicha información con mayor frecuencia.

3.1 Estructura del Documento

Se comienza con una breve descripción acerca de cómo insertar un componente Ladder y luego se listan los componentes Ladder agrupados por categoría.

Cada componente se nombra con el nombre de tipo único definido por el sistema y un nombre mas descriptivo entre paréntesis si es aplicable.

Ejemplo: **SwitchNo** (Contacto Normal Abierto)

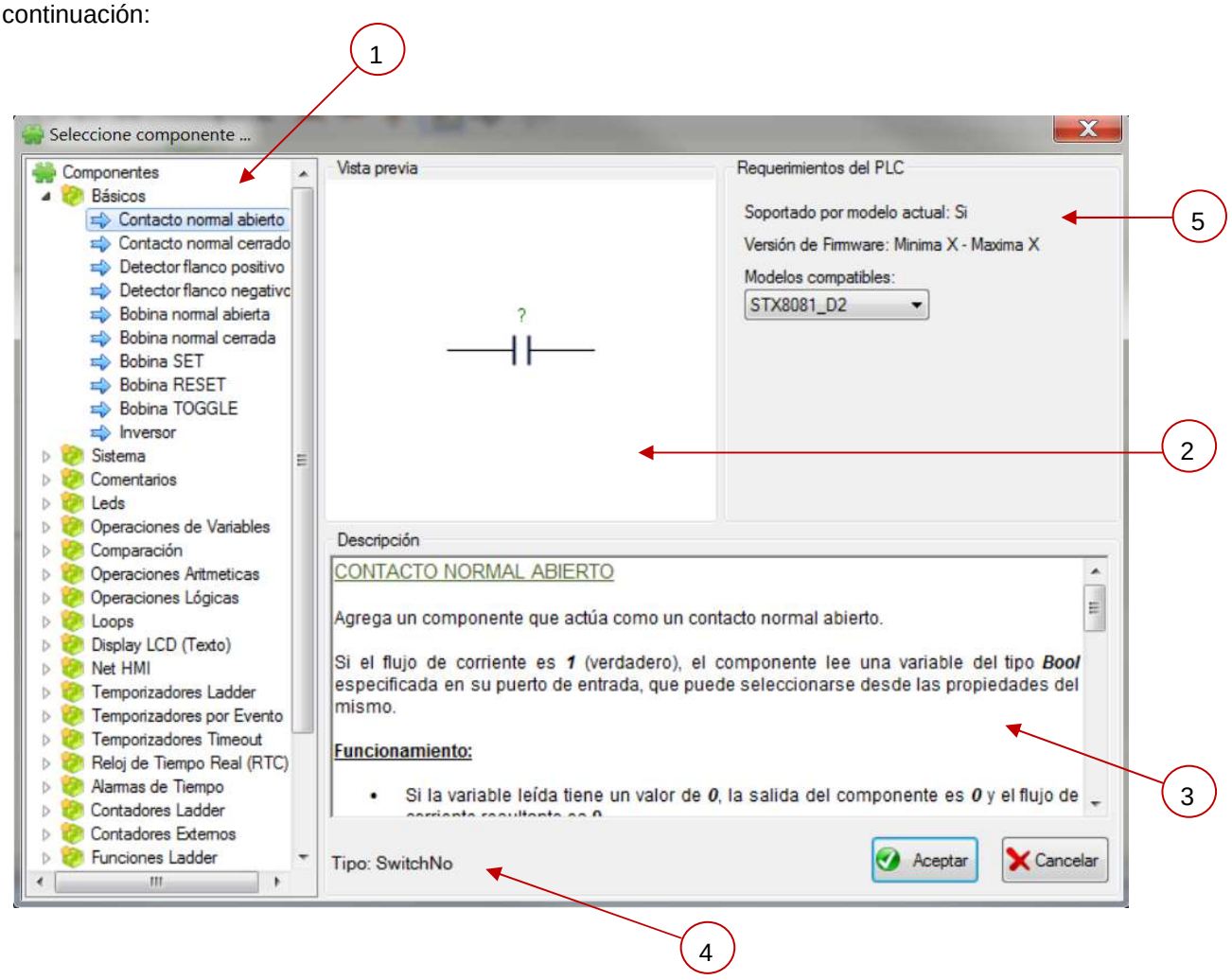
El nombre del tipo de componente es “**SwitchNo**” y “(Contacto Normal Abierto)” es un nombre descriptivo e informal para fácil identificación.

4 Selección de Componentes Ladder

Para seleccionar e insertar un componente Ladder dentro de un diagrama, primero debe seleccionar una celda libre en el diagrama para almacenar el componente.

Luego desde la **Barra de Componentes del Diagrama**, seleccione el icono “Agregar componentes” como se muestra a continuación:

Luego haga click en la celda libre y aparecerá la ventana de selección de componentes como se muestra a continuación:



En la figura de la página anterior se han señalado con flechas y números las siguientes areas:

1. **Componentes**: Se listan todos los componentes soportados por el software y agrupados por categoría. Desde aquí puede seleccionarlos.
2. **Vista Previa**: Permite ver la forma del componente previo a insertarlo en el diagrama. La imagen es representativa.
3. **Descripcion**: Descripcion completa del componente, modo de empleo y consejos.
4. **Tipo**: Nombre único del tipo de componente.
5. **Requerimientos del PLC**: Esta sección le permite saber si el PLC soporta el componente seleccionado, los modelos alternativos de PLC que soportan el componente y la versión de firmware máximo o minimo requeridos (en caso de ser especificada).

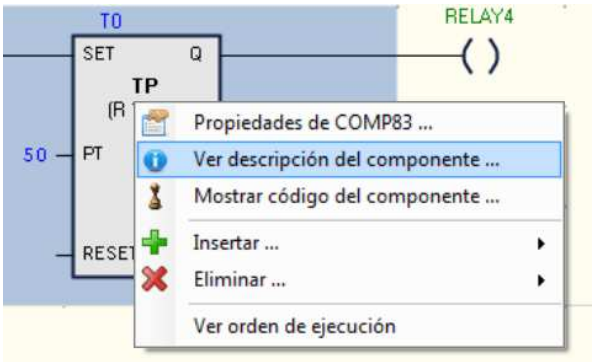
Para insertar el componente, presione el botón “Aceptar”.

En caso de no haber suficiente espacio (celdas ocupadas o posición ilegal) el entorno le mostrará un cartel indicando el tipo de error.

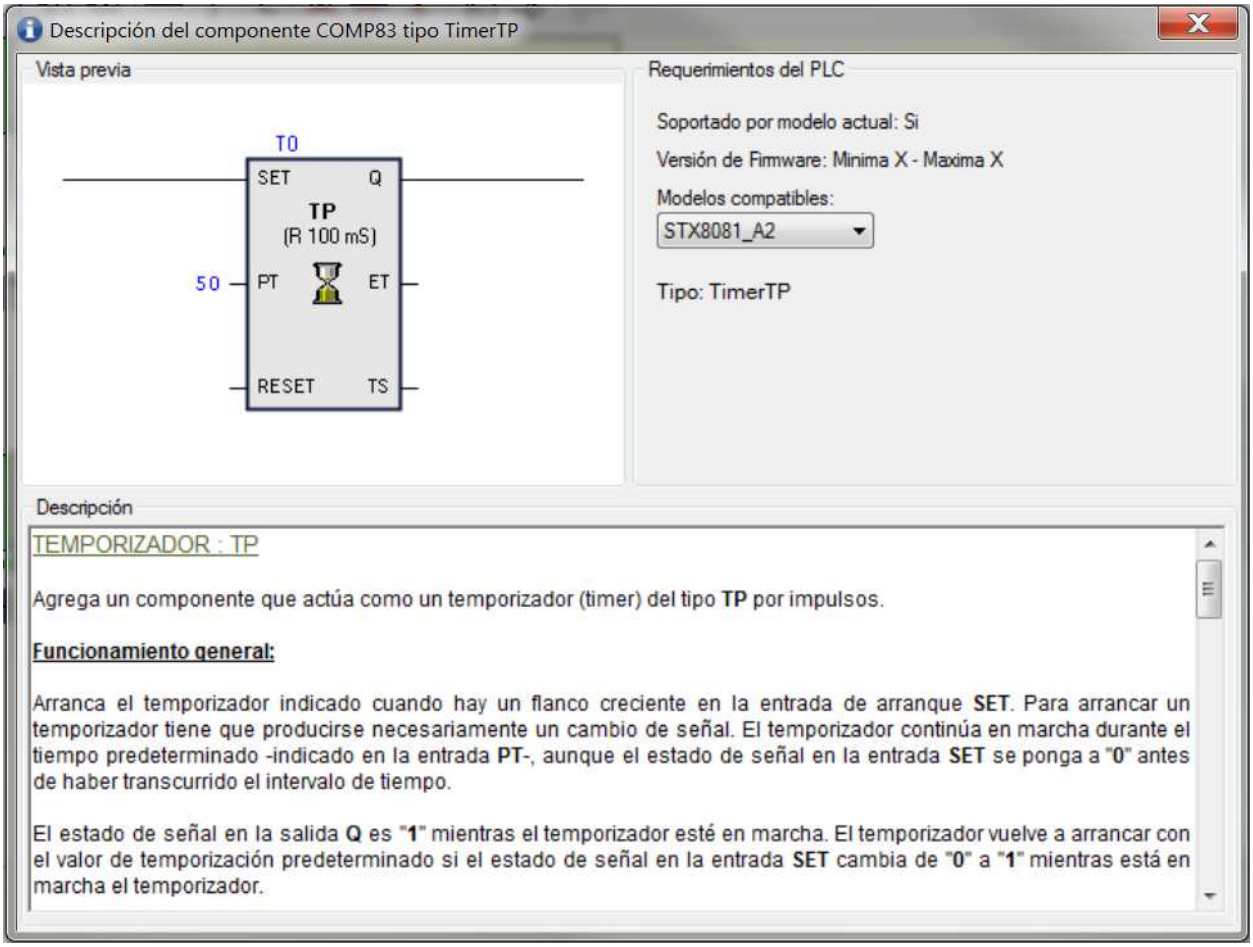
Si el componente no esta soportado por el modelo de PLC seleccionado por el proyecto, un cartel de advertencia le informará para que lo tenga en cuenta. En caso querer compilar el proyecto con un componente no soportado, el entorno le mostrará un error y no prodra compilar hasta que elimine el o los componentes incompatibles.

4.1 Ver Descripcion del Componente

Una característica muy practica de StxLadder, es que puede fácilmente obtener una descripción de los componentes ya insertados en el diagrama, haciendo click-derecho sobre el componente y seleccionado “Ver descripción del componente…” como se muestra a continuación:



La ventana de la descripción del componente se muestra a continuación:



La descripción del componente le brinda toda la información del componente (tipo, vista previa, requerimientos del PLC, etc) que ya fue tratada en la sección anterior.

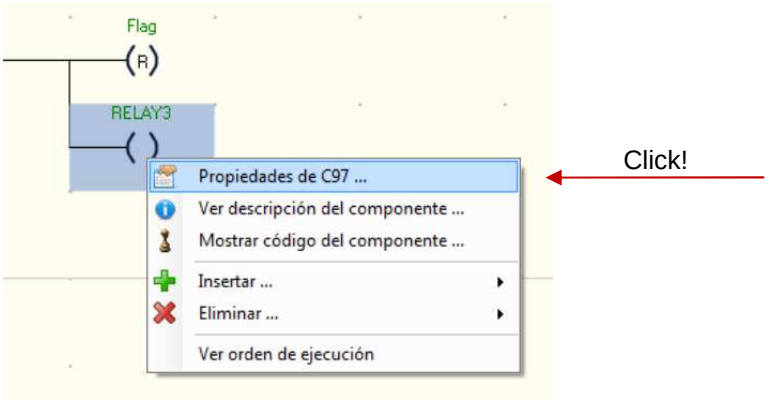
Este documento incluye las mismas descripciones que las incluidas en el software StxLadder, solo que en el software puede estar mas actualizado.

4.2 Acceso a las Propiedades del Componente

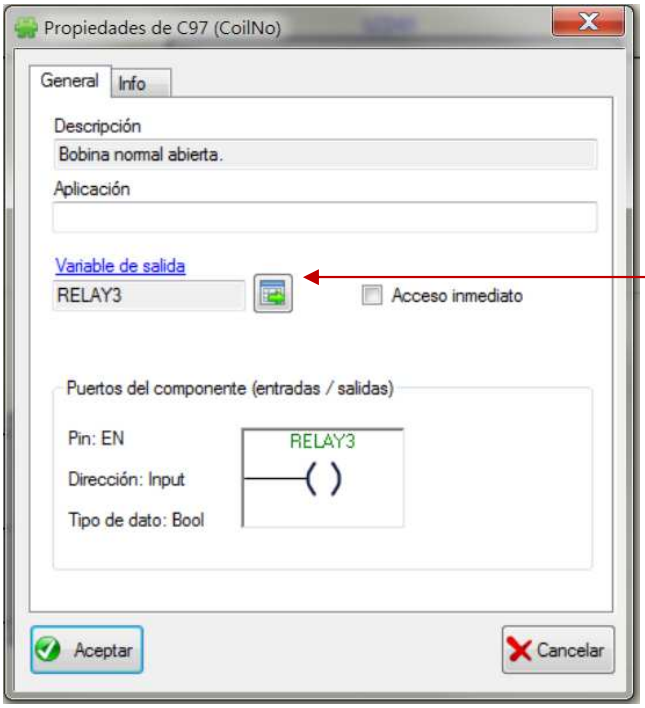
La mayoria de los componentes son configurables a través de sus propiedades, ya sea para seleccionar variables en sus puertos de entrada/salida o para especificar datos generales. El mecanismo de configuración y selección de variables se explicó en el Manual de Usuario con ejemplos.

En esta sección haremos un breve repaso.

Para acceder a las propiedades del componente diríjase a su ubicación dentro del diagrama Ladder y haga doble-click sobre el componente. Tambien puede hacer click-derecho sobre el componente y seleccionar "**Propiedades …"** como se muestra a continuación en un componente bobina de prueba:

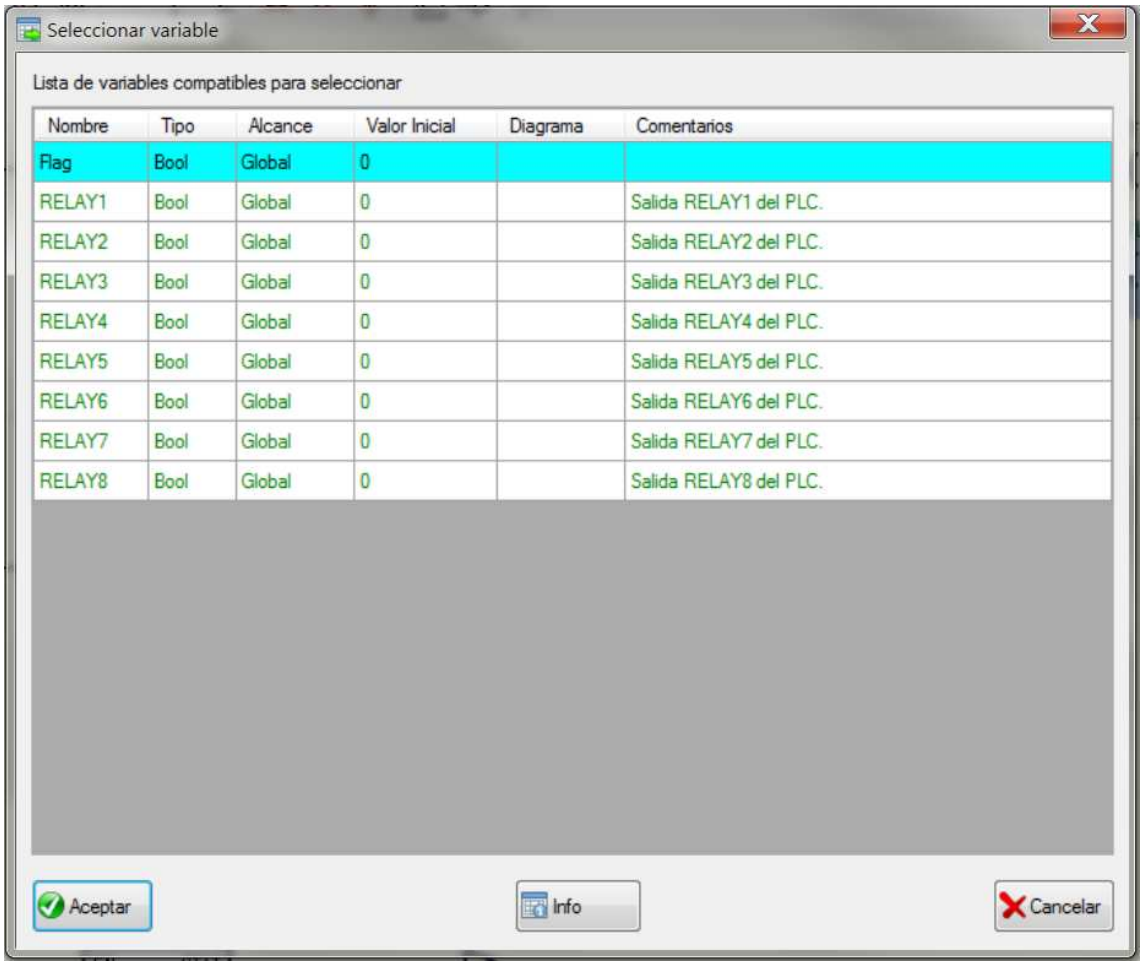


Aparecerá la siguiente ventana:



Si hace click en "**Variable de salida**" puede acceder al seleccionador de variables que le mostrará todas las variables compatibles para el puerto del componente.

A continuación mostramos la ventana de selección:



Como se aprecia en la figura superior, puede elegir otras variables para el componente (por ejemplo RELAY1, RELAY2 o Flag).

La mayoría de los componentes que requieran o puedan seleccionar variables utilizan este mecanismo.

Remitase al Manual de Usuario del StxLadder para mayor Información.

5 Componentes Básicos

Los componentes básicos son aquellos de uso frecuentes: contactos, bobinas, etc. Se listan a continuación.

5.1 SwitchNo (Contacto Normal Abierto)

Agrega un componente que actúa como un contacto normal abierto.

Si el flujo de corriente es ***1*** (verdadero), el componente lee una variable del tipo ***Bool*** especificada en su puerto de entrada, que puede seleccionarse desde las propiedades del mismo.

5.1.1 Funcionamiento:

- Si la variable leída tiene un valor de ***0***, la salida del componente es ***0*** y el flujo de corriente resultante es **0**.
- Si la variable leída tiene un valor de ***1***, la salida del componente es ***1*** y el flujo de corriente resultante es **1**.

Típicamente el componente se utiliza para realizar condicionales leyendo un valor del tipo ***Bool***, si el valor resultante es "**1**", el resto de los componentes conectados en serie a este componente se ejecutarán.

5.1.2 Acceso Inmediato:

- Es posible acceder directamente al valor de un periférico de entrada (por ejemplo una entrada discreta/digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza en memoria el valor de los periféricos de entrada al comienzo de cada SCAN cycle. Si se utiliza el acceso inmediato, el componente accederá al valor actual del periférico. Esto es útil para obtener el valor más reciente sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a un valor obtenido de un periférico de entrada, el acceso inmediato es redundante y no tiene efecto, porque el valor ya se encuentra actualizado en memoria.
- El acceso inmediato lee una entrada de un periférico sin filtrar.
- Para un periférico de salida, el acceso inmediato obtiene el valor actual del periférico.

5.1.3 TIPS:

- En programación escrita o en el lenguaje Pawn, este componente es conocido como una operación "***if***".

5.2 SwitchNc (Contacto Normal Cerrado)

Agrega un componente que actúa como un contacto normal cerrado.

Si el flujo de corriente es ***1*** (verdadero), el componente lee una variable del tipo ***Bool*** especificada en su puerto de entrada, que puede seleccionarse desde las propiedades del mismo.

5.2.1 Funcionamiento:

- Si la variable leída tiene un valor de ***0***, la salida del componente es ***1*** y el flujo de corriente resultante es **1**.
- Si la variable leída tiene un valor de ***1***, la salida del componente es ***0*** y el flujo de corriente resultante es **0**.

Típicamente el componente se utiliza para realizar condicionales leyendo un valor del tipo ***Bool***, si el valor resultante es "**1**", el resto de los componentes conectados en serie a este componente se ejecutarán.

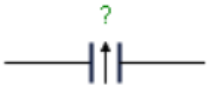
5.2.2 Acceso Inmediato:

- Es posible acceder directamente al valor de un periférico de entrada (por ejemplo una entrada discreta/digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza en memoria el valor de los periféricos de entrada al comienzo de cada SCAN cycle. Si se utiliza el acceso inmediato, el componente accederá al valor actual del periférico. Esto es útil para obtener el valor más reciente sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a un valor obtenido de un periférico de entrada, el acceso inmediato es redundante y no tiene efecto, porque el valor ya se encuentra actualizado en memoria.
- El acceso inmediato lee una entrada de un periférico sin filtrar.
- Para un periférico de salida, el acceso inmediato obtiene el valor actual del periférico.

5.2.3 TIPS:

- En programación escrita o en el lenguaje Pawn, este componente es conocido como una operación "***if***".

5.3 SwitchPE (Detector de Flanco Positivo)



Agrega un componente que detecta un flanco positivo en el cambio de estado de una variable del tipo **Bool** especificada como parámetro en su puerto de entrada. La variable se especifica en las propiedades del componente.

El componente es ejecutado si el flujo de corriente es **1** (verdadero).

5.3.1 Funcionamiento:

- Si la variable leída pasa de un valor **0** a **1** (flanco positivo), la salida del componente es **1** y el flujo de corriente resultante es **1**.
- Si la variable leída tiene un valor **0** o **1**, la salida del componente es **0** y el flujo de corriente resultante es **0**.

Opcionalmente, puede detectar el flanco positivo del flujo de corriente (**CurrentFlow**), especificando "**Usar CurrentFlow**" desde las propiedades del componente. Esto le permite detectar flancos sin utilizar variables intermedias, que puede ser muy útil en ciertas lógicas cableadas.

5.3.2 Acceso Inmediato:

- Es posible acceder directamente al valor de un periférico de entrada (por ejemplo una entrada discreta/digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza en memoria el valor de los periféricos de entrada al comienzo de cada SCAN cycle. Si se utiliza el acceso inmediato, el componente accederá al valor actual del periférico. Esto es útil para obtener el valor más reciente sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a un valor obtenido de un periférico de entrada, el acceso inmediato es redundante y no tiene efecto, porque el valor ya se encuentra actualizado en memoria.
- El acceso inmediato lee una entrada de un periférico sin filtrar.
- Para un periférico de salida, el acceso inmediato obtiene el valor actual del periférico.

5.3.3 Notas Importantes:

- Tenga especial cuidado al utilizar este componente dentro de funciones **Ladder** o en lógicas donde no se llame frecuentemente (conectado en serie luego de otros componentes). Porque internamente el componente guarda el estado previo de la variable leída en una variable interna. Si la variable interna no es refrescada periódicamente con el último valor de la variable leída, debido a que el componente no fue ejecutado, es posible perder el flanco o transición de la variable leída y el componente retornara un flujo de corriente con valor **0**.
- Habitualmente este componente se utiliza como primer componente conectado a la barra de alimentación en cualquier **Network** del diagrama principal.

5.4 SwitchNE (Detector de Flanco Negativo)

Agrega un componente que detecta un flanco negativo en el cambio de estado de una variable del tipo **Bool** especificada como parámetro en su puerto de entrada. La variable se especifica en las propiedades del componente.

El componente es ejecutado si el flujo de corriente es **1** (verdadero).

5.4.1 Funcionamiento:

- Si la variable leída pasa de un valor **1** a **0** (flanco negativo), la salida del componente es **1** y el flujo de corriente resultante es **1**.
- Si la variable leída tiene un valor **0** o **1**, la salida del componente es **0** y el flujo de corriente resultante es **0**.

Opcionalmente, puede detectar el flanco negativo del flujo de corriente (**CurrentFlow**), especificando "**Usar CurrentFlow**" desde las propiedades del componente. Esto le permite detectar flancos sin utilizar variables intermedias, que puede ser muy útil en ciertas lógicas cableadas.

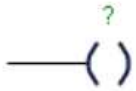
5.4.2 Acceso Inmediato:

- Es posible acceder directamente al valor de un periférico de entrada (por ejemplo una entrada discreta/digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza en memoria el valor de los periféricos de entrada al comienzo de cada SCAN cycle. Si se utiliza el acceso inmediato, el componente accederá al valor actual del periférico. Esto es útil para obtener el valor más reciente sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a un valor obtenido de un periférico de entrada, el acceso inmediato es redundante y no tiene efecto, porque el valor ya se encuentra actualizado en memoria.
- El acceso inmediato lee una entrada de un periférico sin filtrar.
- Para un periférico de salida, el acceso inmediato obtiene el valor actual del periférico.

5.4.3 Notas Importantes:

- Tenga especial cuidado al utilizar este componente dentro de funciones **Ladder** o en lógicas donde no se llame frecuentemente (conectado en serie luego de otros componentes). Porque internamente el componente guarda el estado previo de la variable leída en una variable interna. Si la variable interna no es refrescada periódicamente con el último valor de la variable leída, debido a que el componente no fue ejecutado, es posible perder el flanco o transición de la variable leída y el componente retornara un flujo de corriente con valor **0**. Es posible que retorne **1**, si el último valor leído fue "**1**" y el valor actual es "**0**" (por ejemplo, si el componente se ejecuta dentro de una función).
- Habitualmente este componente se utiliza como primer componente conectado a la barra de alimentación en cualquier **Network** del diagrama principal.

5.5 CoilNo (Bobina Normal Abierta)



Agrega un componente que actúa como un relé o bobina normal abierta.

El componente, altera una variable del tipo **Bool** especificada en su puerto de salida, que puede definirse desde las propiedades del mismo.

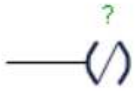
5.5.1 Funcionamiento:

- Si el flujo de corriente a la entrada al componente tiene un valor de ***1***, la variable de salida es puesta en ***1.***
- Si el flujo de corriente a la entrada al componente tiene un valor de ***0***, la variable de salida es puesta en ***0.***

5.5.2 Acceso Inmediato:

- Es posible operar directamente sobre el valor de un periférico de salida (por ejemplo un relé o salida digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza el valor de los periféricos de salida al comienzo de cada SCAN cycle con los valores almacenados en memoria del ciclo anterior. Si se utiliza el acceso inmediato, el componente modificará directamente el valor actual del periférico y la memoria imagen. Esto es útil para modificar el valor del periférico sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a periférico de salida, el acceso inmediato es redundante y no tiene efecto, porque el valor se modifica en memoria instantáneamente.
- El acceso inmediato opera directamente sobre el periférico de salida.

5.6 CoilNc (Bobina Normal Cerrada)



Agrega un componente que actúa como un relé o bobina normal cerrada.

El componente, altera una variable del tipo **Bool** especificada en su puerto de salida, que puede definirse desde las propiedades del mismo.

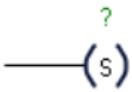
5.6.1 Funcionamiento:

- Si el flujo de corriente a la entrada al componente tiene un valor de ***0***, la variable de salida es puesta en ***1.***
- Si el flujo de corriente a la entrada al componente tiene un valor de ***1***, la variable de salida es puesta en ***0.***

5.6.2 Acceso Inmediato:

- Es posible operar directamente sobre el valor de un periférico de salida (por ejemplo un relé o salida digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza el valor de los periféricos de salida al comienzo de cada SCAN cycle con los valores almacenados en memoria del ciclo anterior. Si se utiliza el acceso inmediato, el componente modificará directamente el valor actual del periférico y la memoria imagen. Esto es útil para modificar el valor del periférico sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a periférico de salida, el acceso inmediato es redundante y no tiene efecto, porque el valor se modifica en memoria instantáneamente.
- El acceso inmediato opera directamente sobre el periférico de salida.

5.7 CoilSet (Bobina Set)



Agrega un componente que actúa como una bobina "***SET***".

Solo si el flujo de corriente es 1 (verdadero) en su puerto de entrada cableado, se procede a ejecutar el componente.

Una ***bobina set*** establece a "1" el estado lógico de la variable del tipo **Bool** especificada en su puerto de salida, que puede definirse desde las propiedades del componente.

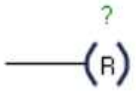
5.7.1 Acceso Inmediato:

- Es posible operar directamente sobre el valor de un periférico de salida (por ejemplo un relé o salida digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza el valor de los periféricos de salida al comienzo de cada SCAN cycle con los valores almacenados en memoria del ciclo anterior. Si se utiliza el acceso inmediato, el componente modificará directamente el valor actual del periférico y la memoria imagen. Esto es útil para modificar el valor del periférico sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a periférico de salida, el acceso inmediato es redundante y no tiene efecto, porque el valor se modifica en memoria instantáneamente.
- El acceso inmediato opera directamente sobre el periférico de salida.

5.7.2 Notas:

- Una vez activa, no se puede desactivar si no es por su correspondiente bobina "**RESET**".

5.8 CoilReset (Bobina Reset)



Agrega un componente que actúa como una bobina "***RESET***".

Solo si el flujo de corriente es 1 (verdadero) en su puerto de entrada cableado, se procede a ejecutar el componente.

Una ***bobina reset*** establece a "0" el estado lógico de la variable del tipo **Bool** especificada en su puerto de salida, que puede definirse desde las propiedades del componente.

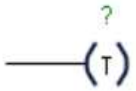
5.8.1 Notas:

- Permite desactivar una bobina "**SET**" previamente activada.

5.8.2 Acceso Inmediato:

- Es posible operar directamente sobre el valor de un periférico de salida (por ejemplo un relé o salida digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza el valor de los periféricos de salida al comienzo de cada SCAN cycle con los valores almacenados en memoria del ciclo anterior. Si se utiliza el acceso inmediato, el componente modificará directamente el valor actual del periférico y la memoria imagen. Esto es útil para modificar el valor del periférico sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a periférico de salida, el acceso inmediato es redundante y no tiene efecto, porque el valor se modifica en memoria instantáneamente.
- El acceso inmediato opera directamente sobre el periférico de salida.

5.9 CoilToggle (Bobina Toggle)



Agrega un componente que actúa como una bobina "***TOGGLE***".

Solo si el flujo de corriente es 1 (verdadero) en su puerto de entrada cableado, se procede a ejecutar el componente.

Una ***bobina toggle*** invierte el estado lógico de la variable del tipo **Bool** especificada en su puerto de salida, que puede definirse desde las propiedades del componente.

5.9.1 Funcionamiento:

- Si la variable seleccionada tiene un valor "0", la ***bobina toggle*** la convierte en "1".
- Si la variable seleccionada tiene un valor "1", la ***bobina toggle*** la convierte en "0".

5.9.2 Acceso Inmediato:

- Es posible operar directamente sobre el valor de un periférico de salida (por ejemplo un relé o salida digital) si se tilda la opción "**Acceso Inmediato**" en las propiedades del componente.
- El PLC actualiza el valor de los periféricos de salida al comienzo de cada SCAN cycle con los valores almacenados en memoria del ciclo anterior. Si se utiliza el acceso inmediato, el componente modificará directamente el valor actual del periférico y la memoria imagen. Esto es útil para modificar el valor del periférico sin esperar un nuevo SCAN cycle.
- Un icono de un relámpago en el dibujo del componente se mostrará si el acceso es inmediato.
- Si la variable a leer no corresponde a periférico de salida, el acceso inmediato es redundante y no tiene efecto, porque el valor se modifica en memoria instantáneamente.
- El acceso inmediato opera directamente sobre el periférico de salida.

5.10 Not (Inversor)

Agrega un componente que invierte el estado lógico del flujo de corriente en el cableado.

5.10.1 Funcionamiento:

- Si el flujo corriente es ***1***, el componente lo transformará en ***0***.
- Si el flujo corriente es ***0***, el componente lo transformará en ***1***.

5.10.2 Notas:

- Habitualmente se denomina al flujo de corriente como "**CurrentFlow**" en los PLC de Slicetex.
- Puede utilizar este componente para lograr resultados opuestos a la salida de otros componentes.

6 Componentes del Sistema

Los componentes del sistema permiten configurar o modificar el funcionamiento del PLC u obtener información del mismo.

6.1 DinDelay (Retardo de Lectura de Entradas Discretas)

Agrega un componente que permite especificar un retardo de lectura para las entradas digitales (**DIN**).

El PLC actualiza en una memoria de periféricos o imagen, el estado de las entradas digitales o discretas.

La actualización de memoria se produce en cada SCAN cycle del PLC.

Este componente permite especificar un tiempo mínimo de actualización de las entradas digitales en memoria.

Si hay un retardo especificado por este componente, el PLC actualizará el estado de las entradas digitales al comienzo del SCAN cycle, solo si ha pasado un tiempo igual o mayor al especificado por el componente.

El retardo actúa como un filtro en señales digitales que contienen ruidos, como rebotes de contactos o cables con perturbaciones.

6.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**MS**" se especifica el retardo en Mili-Segundos. El puerto puede leer un valor de una constante o variable tipo **Int32**. Valor máximo admitido es de 2550. La resolución es de 10 mS.

6.1.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

6.1.3 Atención:

- Un valor de **MS** = 0, implica retardo "nulo", limitado solo por el tiempo entre cada SCAN cycle del PLC.
- Es posible acceder de forma inmediata a una entrada, esto es, ignorando el retardo configurado. Consulte la documentación de los componentes que leen entradas digitales del PLC.
- El valor por defecto que utiliza el PLC es de **MS** = 10. Valores superiores a **MS** = 200 son poco recomendables, ya que obtendrá una actualización lenta de las entradas digitales.
- Un valor de **MS** = 20, filtra bastante el ruido. Un ejemplo real, puede ser cuando utilizamos contadores ladder ascendentes con una entrada discreta común, si hay ruidos en el contacto, la cuenta puede incrementarse más de lo esperado con un valor de bajo de **MS**, es decir **MS** = 0 o **MS** = 10.

6.1.4 Notas:

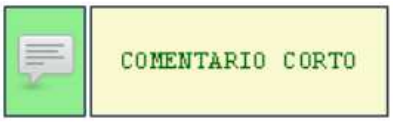
- Habitualmente este componente se llama una sola vez, en el diagrama "Inicio" del proyecto.

7 Componentes para Comentarios

Los componentes para comentarios le permiten documentar y agregar comentarios descriptivos en el diagrama, de tal forma que luego pueda interpretarlos fácilmente.

No influyen en el funcionamiento de la lógica del diagrama y no producen código para el PLC.

7.1 CommentShort (Comentario Corto)



Agrega un componente que muestra un comentario en el diagrama.

No tiene efecto sobre el código generado y permite mostrar texto que ayude a comprender el diagrama.

Es posible seleccionar un icono o dibujo representativo para el comentario y de este modo facilitar la comprensión del tipo de comentario con solo un vistazo rápido en el diagrama.

Se proponen los siguientes tipos de comentarios para uso general:

- **Normal:** Comentario normal.
- **Warning:** Comentario para señalar advertencias o llamar la atención. Muy útil en ocasiones donde necesitamos que se lea el comentario antes de proceder, ya que el contenido es importante.
- **Info:** Comentario en el cual mostramos información. Por ejemplo una formula, un dato o una referencia.
- **Note:** Comentario para señalar una "Nota". Muy útil para indicar particularidades que nos permitan comprender alguna idea o nos ayude a implementarla. Similar a una nota al pie de página de un libro.
- **Text:** Comentario general o normal, pero que es extenso. Al ver el icono de "Text", podremos inferir que el comentario incluido es largo.
- **ToDo:** El término "TO-DO", proviene del inglés y se traduce como "PARA HACER". Un comentario de este tipo puede indicar que hay una tarea pendiente o cierta parte del diagrama no está totalmente terminada (en construcción). Por ejemplo, si en una parte del diagrama, nuestro proyecto debe manejar una emergencia en caso que un motor se caliente, pero la misma no fue implementada, podemos agregar un comentario "**ToDo**" con la leyenda: "Implementar emergencia para motor". En el futuro, al ver el icono del comentario, sabremos que algo falta y esta "PARA HACER".
- **UnderConstruction:** Un comentario de este tipo indica que el diagrama (o una parte del mismo) se encuentra "Bajo Construcción" o "En desarrollo". Por lo tanto la lógica no está completamente probada (o terminada) y debe utilizarse con precaución. En programación suele decirse que la lógica o el programa están en un estado "Beta".

7.1.1 Nota:

El contenido de los tipos de comentarios es sugerido y puede utilizarlos según sus gustos. Pero tenga en cuenta que luego es posible buscarlos por "tipo" y será más fácil filtrarlos si el contenido de los comentarios es coherente.

7.2 ComentLong (Comentario Largo)

Agrega un componente que muestra un comentario en el diagrama.

No tiene efecto sobre el código generado y permite mostrar texto que ayude a comprender el diagrama.

Es posible seleccionar un icono o dibujo representativo para el comentario y de este modo facilitar la comprensión del tipo de comentario con solo un vistazo rápido en el diagrama.

Se proponen los siguientes tipos de comentarios para uso general:

- **Normal:** Comentario normal.
- **Warning:** Comentario para señalar advertencias o llamar la atención. Muy útil en ocasiones donde necesitamos que se lea el comentario antes de proceder, ya que el contenido es importante.
- **Info:** Comentario en el cual mostramos información. Por ejemplo una formula, un dato o una referencia.
- **Note:** Comentario para señalar una “Nota”. Muy útil para indicar particularidades que nos permitan comprender alguna idea o nos ayude a implementarla. Similar a una nota al pie de página de un libro.
- **Text:** Comentario general o normal, pero que es extenso. Al ver el icono de “Text”, podremos inferir que el comentario incluido es largo.
- **ToDo:** El término “TO-DO”, proviene del inglés y se traduce como “PARA HACER”. Un comentario de este tipo puede indicar que hay una tarea pendiente o cierta parte del diagrama no está totalmente terminada (en construcción). Por ejemplo, si en una parte del diagrama, nuestro proyecto debe manejar una emergencia en caso que un motor se caliente, pero la misma no fue implementada, podemos agregar un comentario “**ToDo**” con la leyenda: “Implementar emergencia para motor”. En el futuro, al ver el icono del comentario, sabremos que algo falta y esta “PARA HACER”.
- **UnderConstruction:** Un comentario de este tipo indica que el diagrama (o una parte del mismo) se encuentra “Bajo Construcción” o “En desarrollo”. Por lo tanto la lógica no está completamente probada (o terminada) y debe utilizarse con precaución. En programación suele decirse que la lógica o el programa están en un estado “Beta”.

7.2.1 Nota:

El contenido de los tipos de comentarios es sugerido y puede utilizarlos según sus gustos. Pero tenga en cuenta que luego es posible buscarlos por “tipo” y será más fácil filtrarlos si el contenido de los comentarios es coherente.

8 Componentes Leds

Componentes para controlar Leds (Light Emiting Diode) de propósito general incluidos en el PLC. Pueden ser utilizados por el usuario para indicar o señalizar alguna condición dentro del PLC. Muy útil para el proceso de desarrollo de programas.

8.1 LedDebug (Led para Depuración)

Agrega un componente que permite controlar el LED utilizado para debug o pruebas del PLC.

Se pueden definir en las propiedades del componente 3 tipos de operaciones para el LED:

1. **ON** : Activa el LED si es ejecutado.
2. **OFF** : Desactiva el LED si es ejecutado.
3. **TOGGLE** : Invierte el estado del LED si es ejecutado.

8.1.1 Funcionamiento:

- Si el flujo corriente es ***1***, el componente ejecutará la operación de control para el LED.
- Si el flujo corriente es ***0***, el componente no ejecutará la operación de control para el LED.

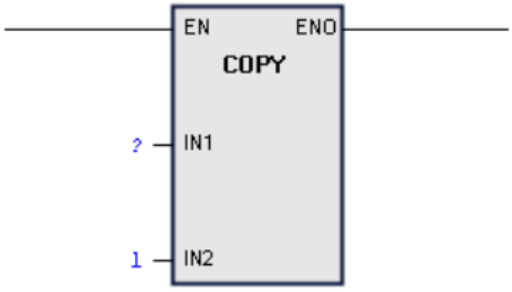
8.1.2 Notas:

- Este componente es muy útil para realizar indicaciones visuales y probar ciertas partes del código o diagrama Ladder. Muy usado en el proceso de desarrollo de un programa.
- Algunos modelos de PLC pueden no disponer de este LED.

9 Componentes para Operaciones de Variables

Son componentes que realizan operaciones generales sobre variables: copiar, asignar valor, incrementar, etc.

9.1 Int32Copy (Copia/Asignación de variable Int32)



Agrega un componente que permite copiar o asignar un valor a una variable del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

9.1.1 Funcionamiento:

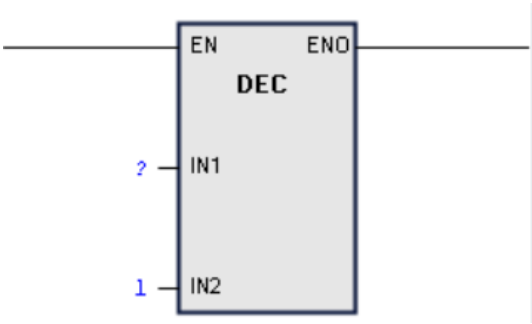
En el puerto de entrada "**IN1**" se especifica una variable tipo **Int32**, a la cual se le copiara el valor de otra variable o constante tipo **Int32** definida en el puerto "**IN2**".

La operación resultante es:

IN1 = IN2

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

9.2 Int32Dec (Decrementar Variable Int32)



Agrega un componente que permite decrementar el valor de una variable tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

9.2.1 Funcionamiento:

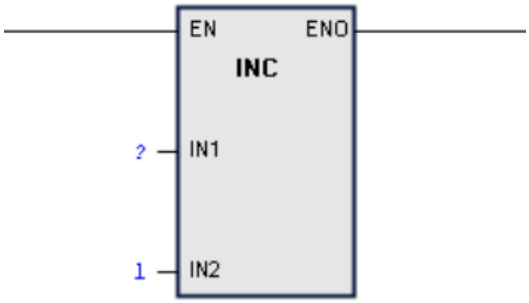
En el puerto de entrada "**IN1**" se especifica una variable tipo **Int32**, a la cual se le restará el valor de otra variable o constante tipo **Int32** definida en el puerto "**IN2**".

La operación resultante es:

IN1 = IN1 - IN2

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

9.3 Int32Inc (Incrementar Variable Int32)



Agrega un componente que permite incrementar el valor de una variable tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

9.3.1 Funcionamiento:

En el puerto de entrada "**IN1**" se especifica una variable tipo **Int32**, a la cual se le sumará el valor de otra variable o constante tipo **Int32** definida en el puerto "**IN2**".

La operación resultante es:

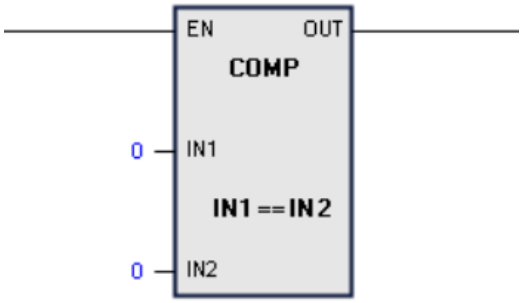
IN1 = IN1 + IN2

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

10 Componentes de Comparación

Los componentes de comparación le permiten comparar valores entre variables, constantes o ambas. Por ejemplo, puede determinar si una variable es mayor o igual a un valor y realizar una acción.

10.1 Int32Comp (Comparación de Variables Int32)



Agrega un componente que permite comparar dos valores del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

10.1.1 Funcionamiento:

- En el puerto de entrada "**IN1**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de entrada "**IN2**" se especifica una variable o constante del tipo **Int32**.

Luego desde las propiedades del componente, es posible elegir una de las siguientes operaciones de comparación:

- **EQ**: Compara si **IN1** es igual a **IN2**.
- **GE**: Compara si **IN1** es mayor o igual a **IN2**.
- **GT**: Compara si **IN1** es mayor a **IN2**.
- **LE**: Compara si **IN1** es menor o igual a **IN2**.
- **LT**: Compara si **IN1** es menor a **IN2**.
- **NE**: Compara si **IN1** es distinto a **IN2**.

Si la operación de comparación se cumple, el resultado es "**1**" lógico. De lo contrario es "**0**" lógico.

El resultado de la operación de comparación, se copia al puerto de salida "**ENO**".

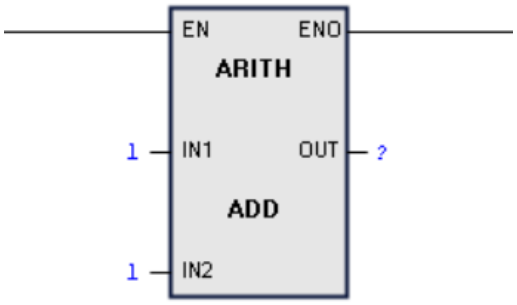
10.1.2 TIPS:

- El tipo de operación se muestra para rápida visualización en el dibujo del componente con un símbolo.
- En las propiedades del componente se puede ver la operación y descripción de forma simple.

11 Componentes para Operaciones Aritméticas

Componentes que le permiten realizar operaciones matemáticas con valores numéricos sobre variables.

11.1 Int32Arithmetic (Operaciones Aritméticas para Variables Int32)



Agrega un componente que permite realizar operaciones aritméticas entre dos valores del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

11.1.1 Funcionamiento:

- En el puerto de entrada "**IN1**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de entrada "**IN2**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de salida "**OUT**" debe especificarse una variable del tipo **Int32** que almacenará el resultado de la operación aritmética.

Luego desde las propiedades del componente, es posible elegir una de las siguientes operaciones aritméticas:

- **ADD**: Suma **IN1** e **IN2**. Resultado en **OUT** = **IN1** + **IN2**.
- **SUB**: Resta **IN2** a **IN1**. Resultado en **OUT** = **IN1** - **IN2**.
- **MUL**: Multiplica **IN1** por **IN2**. Resultado en **OUT** = **IN1** * **IN2**.
- **DIV**: Divide **IN1** por **IN2**. Resultado en **OUT** = **IN1** / **IN2**.
- **INC**: Incrementa en 1 el valor de **IN1**. Resultado en **OUT** = **IN1** + 1.
- **DEC**: Decrementa en 1 el valor de **IN1**. Resultado en **OUT** = **IN1** - 1.
- **CPY**: Copia el valor de **IN1** en **OUT**. Resultado **OUT** = **IN1**.
- **REM**: Obtiene el remanente o resto de la división de **IN1** por **IN2**. El signo del resultado es el mismo del operando **IN2**. Resultado en **OUT** = **IN1** % **IN2**.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

11.1.2 TIPS:

- El tipo de operación se muestra para rápida visualización en el dibujo del componente con un símbolo.
- En las propiedades del componente se puede ver la operación y descripción de forma simple.

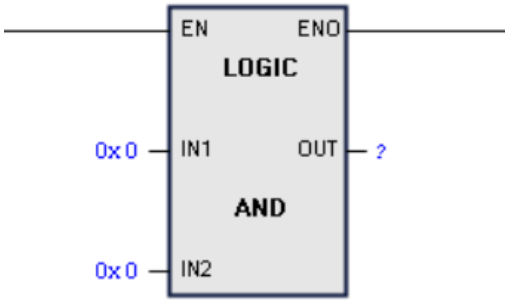
11.1.3 Notas:

- Una operación aritmética se entiende por una operación matemática con valores numéricos. Notar la diferencia con una operación lógica que utiliza valores booleanos o bits.

12 Componentes para Operaciones Lógicas

Componentes para realizar operaciones lógicas tipo AND, NOT, OR, XOR, SHIFT, etc. Una operación lógica se entiende por una operación bit-a-bit, utilizando algebra de **bool**. No confundir con una operación aritmética que comprende valores numéricos en base 10 o 16.

12.1 Int32Logic (Operaciones Lógicas para Variables Int32)



Agrega un componente que permite realizar operaciones lógicas entre dos valores del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

12.1.1 Funcionamiento:

- En el puerto de entrada "**IN1**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de entrada "**IN2**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de salida "**OUT**" debe especificarse una variable del tipo **Int32** que almacenará el resultado de la operación lógica.

Luego desde las propiedades del componente, es posible elegir una de las siguientes operaciones lógicas:

- **And**: AND lógico entre **IN1** e **IN2**. Resultado en **OUT** = **IN1** AND **IN2**.
- **AndNot**: AND lógico y luego negar (NOT). Resultado en **OUT** = NOT(**IN1** AND **IN2)**.
- **Not**: Negar **IN1**. Complemento a 1. Resultado en **OUT** = NOT(**IN1**).
- **Or**: OR lógico entre **IN1** e **IN2**. Resultado en **OUT** = **IN1** OR **IN2**.
- **OrNot**: OR lógico y luego negar (NOT). Resultado en **OUT** = NOT(**IN1** OR **IN2)**.
- **Xor**: OR exclusivo. Resultado en **OUT** = **IN1** XOR **IN2**.
- **Comp2**: Complemento a dos. Negar IN1 y luego sumar 1. Resultado **OUT** = NOT(**IN1**) + 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

12.1.2 TIPS:

- El tipo de operación se muestra para rápida visualización en el dibujo del componente con un símbolo.
- En las propiedades del componente se puede ver la operación y descripción de forma simple.

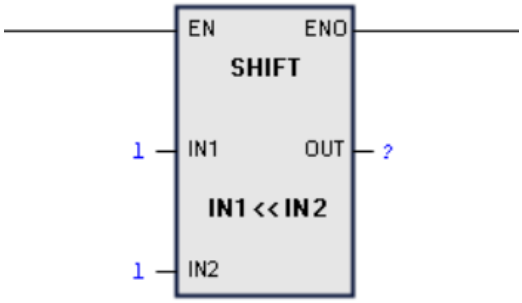
12.1.3 Notas:

- Una operación lógica se entiende por una operación bit-a-bit, utilizando algebra de **bool**. No confundir con una operación aritmética que comprende valores numéricos en base 10 o 16.
- En las propiedades del componente se especifican los valores de constantes en formato hexadecimal.
- Se aconseja repasar algebra de bool. En internet hay vasta bibliografía.

12.1.4 Ejemplos:

- Si **IN1** = 3 y **IN2** = 1, una operación AND resulta en: **OUT** = **IN1** AND **IN2** = 3 AND 1 = 0011b AND 0001b, resultando en **OUT** el valor 0001b. Notar como la operación AND se realiza bit a bit. La letra “b” minúscula indica notación binaria.
- Si **IN1** = 5, una operación NOT resulta en: **OUT** = NOT(**IN1**) = NOT(0101b), resultando en **OUT** el valor 1010b. Notar cómo se invirtieron todos los bits de **IN1**.

12.2 Int32Shift (Operaciones de Desplazamiento para Variables Int32)



Agrega un componente que permite realizar operaciones de desplazamiento (shift) a valores del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

12.2.1 Funcionamiento:

- En el puerto de entrada "**IN1**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de entrada "**IN2**" se especifica una variable o constante del tipo **Int32**.
- En el puerto de salida "**OUT**" debe especificarse una variable del tipo **Int32** que almacenará el resultado de la operación de desplazamiento.

Luego desde las propiedades del componente, es posible elegir una de las siguientes operaciones de desplazamiento:

- **Left**: Desplazamiento lógico hacia la izquierda de **IN1** por **IN2** bits. Los bits vacantes de la derecha se llenan con cero. No importa el signo. Resultado en **OUT** = **IN1** << **IN2**.
- **Right**: Desplazamiento lógico hacia la derecha (sin signo) de **IN1** por **IN2** bits. Cada bit vacante se llena con cero. No importa el signo. Resultado en **OUT** = **IN1** >>> **IN2**.
- **RightArithmetic**: Desplazamiento lógico hacia la derecha (con signo) de **IN1** por **IN2** bits. El bit más significativo de **IN1** se copia a cada bit vacante. Resultado en **OUT** = **IN1** >> **IN2**.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

12.2.2 TIPS:

- El tipo de operación se muestra para rápida visualización en el dibujo del componente con un símbolo.
- En las propiedades del componente se puede ver la operación y descripción de forma simple.

12.2.3 Notas:

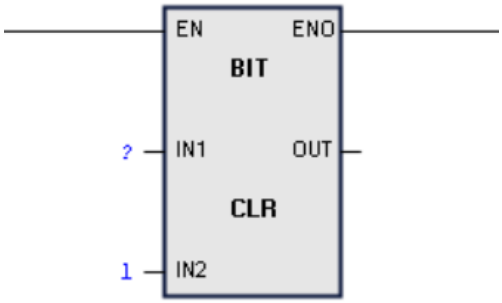
- Los desplazamientos habitualmente se utilizan para multiplicar o dividir en base 2 de forma rápida. Es posible elevar a potencias en base a 2, desplazando hacia la izquierda.
- Se utilizan para los desplazamientos junto a las operaciones lógicas para activar o desactivar determinados bits en variables.

12.2.4 Ejemplos:

- Si **IN1** = 3 y **IN2** = 1, una operación **Left** resulta en: **OUT** = **IN1** << **IN2** = 3 << 1 = 0011b << 0001b, resultando en **OUT** el valor 0110b o 6 en decimal. Notar como la operación **Left** desplaza "**IN2** bits" a **IN1**, lo que equivale a multiplicar por 2 en decimal. La letra "b" minúscula indica notación binaria. Para multiplicar por 4, se debe utilizar el valor 2 en **IN2**.

- Si tenemos el valor de **IN1** = 4 y queremos dividirlo por 2, podemos hacer un desplazamiento **Right**. Entonces: **OUT** = **IN1** >>> **IN2** = 4 >>> 1 = 0100b >>> 0001b = 0010b = 2 decimal. Notar cómo se desplazo en un lugar el valor de **IN1**.

12.3 Int32BitOp (Operaciones de Bits en Variables Int32)



Agrega un componente que permite realizar operaciones de bits a valores del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

12.3.1 Funcionamiento:

- En el puerto de entrada "**IN1**" se especifica una variable del tipo **Int32**.
- En el puerto de entrada "**IN2**" se especifica una variable o constante del tipo **Int32**.
- Dependiendo de la operación, en el puerto de salida "**OUT**" debe especificarse una variable del tipo **Int32** que almacenará un resultado.

Luego desde las propiedades del componente, es posible elegir una de las siguientes operaciones de bits:

- **Clr**: Establece a 0 el bit **IN2** de la variable **IN1**.
- **Set**: Establece a 1 el bit **IN2** de la variable **IN1**.
- **Toggle**: Invierte el estado del bit **IN2** de la variable **IN1**.
- **Read**: Lee el valor del bit **IN2** en la variable **IN1** y lo copia a la variable **OUT**.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

12.3.2 TIPS:

- El tipo de operación se muestra para rápida visualización en el dibujo del componente con un símbolo.
- En las propiedades del componente se puede ver la operación y descripción de forma simple.
- Puede establecer flags dentro de variables tipo **Int32** con estas operaciones.

12.3.3 Notas:

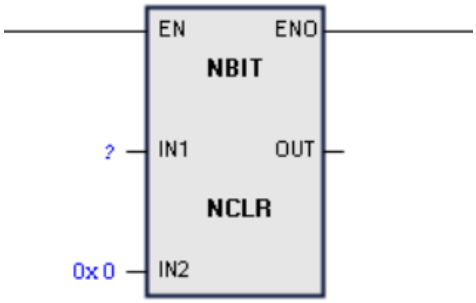
- Estas operaciones simplifican muchas operaciones lógicas que deberían ser implementadas con operaciones AND, OR, XOR, etc.
- Notar que algunas operaciones modifican el valor de **IN1** directamente.
- Operaciones muy utilizadas para manipular y leer bits.

12.3.4 Ejemplos:

- Si **IN1** = 3 y **IN2** = 1, una operación **Clr** (limpiar), pondrá a 0 el bit numero 1 de la entrada **IN1**. Entonces, BitClr(3, 1) = BitClr(0011b, 0001b), resultando **IN1** = 0001b = 1 decimal. Notar como el bit numero 1 se cambia del valor 1 a 0.

- Si **IN1** = 3 y **IN2** = 1, una operación **Read** (leer), retornara el valor del bit numero 1 de **IN1** en OUT. Es decir, **IN1** = 0011b, por lo tanto OUT = 1.

12.4 Int32NBitOp (Operaciones de N-Bit en Variables Int32)



Permite realizar operaciones sobre n-bits (múltiple bits) variables de tipo "**Int32**".

Agrega un componente que permite realizar operaciones sobre n-bits (múltiple bits) a valores del tipo "**Int32**".

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

12.4.1 Funcionamiento:

- En el puerto de entrada "**IN1**" se especifica una variable del tipo **Int32**.
- En el puerto de entrada "**IN2**" se especifica una variable o constante del tipo **Int32**.
- Dependiendo de la operación, en el puerto de salida "**OUT**" debe especificarse una variable del tipo **Int32** que almacenará un resultado.

Luego desde las propiedades del componente, es posible elegir una de las siguientes operaciones de bits:

- **NClr**: Establece a 0 los n-bits **IN2** en la variable **IN1**.
- **NSet**: Establece a 1 los n-bits **IN2** en la variable **IN1**.
- **NToggle**: Invierte el estado de los n-bits **IN2** en la variable **IN1**.
- **NRead**: Lee el valor de los n-bits **IN2** en la variable **IN1**. Si todos son 0, **OUT** es 0. Si alguno tiene un valor distinto a 0, **OUT** es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

12.4.2 TIPS:

- El tipo de operación se muestra para rápida visualización en el dibujo del componente con un símbolo.
- En las propiedades del componente se puede ver la operación y descripción de forma simple.
- Puede establecer flags dentro de variables tipo **Int32** con estas operaciones.

12.4.3 Notas:

- Estas operaciones simplifican muchas operaciones lógicas que deberían ser implementadas con operaciones AND, OR, XOR, etc.
- Notar que algunas operaciones modifican el valor de **IN1** directamente.
- Operaciones muy utilizadas para manipular y leer varios bits al mismo tiempo.
- En las propiedades del componente se especifican los valores de constantes en formato hexadecimal.

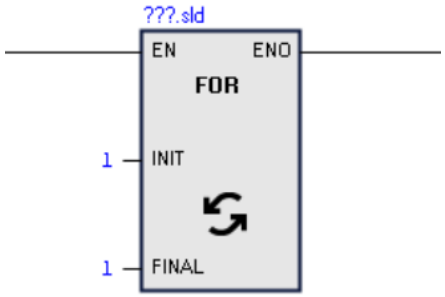
12.4.4 Ejemplos:

- Si **IN1** = 7 y **IN2** = 3, una operación **NClr** (limpiar), pondrá a 0 varios bits de la entrada **IN1**m especificados en **IN2**. Entonces, NBitClr(7, 3) = NBitClr(0111b, 0011b), resultando **IN1** = 0100b = 4 decimal. Notar como los bits que están en 1 en la entrada **IN2** son los bits modificados en la entrada **IN1**.
- Si **IN1** = 4 y **IN2** = 3, una operación **NRead** (leer), comprobará el valor de los bits especificados en **IN2** sobre la entrada **IN1**. Entonces, **OUT** = NBitRead(4, 3) = NBitRead(0100b, 0011b) = 0 en decimal. El resultado es cero, ya que todos los bits en IN1 comprobados por IN2 son 0.
- Si **IN1** = 6 y **IN2** = 3, una operación **NRead** (leer), comprobará el valor de los bits especificados en **IN2** sobre la entrada **IN1**. Entonces, **OUT** = NBitRead(6, 3) = NBitRead(0101b, 0011b) = 1 en decimal. El resultado es 1, ya que al menos un bit en IN1 comprobados por IN2 es 1.

13 Componentes Loops

Los componentes para Loops o Lazos, le permiten ejecutar operaciones de carácter cíclico X cantidad de veces o hasta que se cumpla una condición.

13.1 ForLoop (Ciclo o Lazo FOR)



Agrega un componente tipo loop **FOR**.

Un loop (ciclo o bucle) permite ejecutar fragmentos de código de forma repetitiva una cierta cantidad de veces.

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

13.1.1 Funcionamiento:

- En el puerto de entrada "**INIT**" se especifica un valor del tipo **Int32**, que representa el valor de inicio del ciclo FOR.
- En el puerto de entrada "**FINAL**" se especifica un valor del tipo **Int32**, que representa el valor final que alcanzará el ciclo FOR.

La cantidad de veces que se repetirá el ciclo FOR, va desde el valor de la entrada "**INIT**" hasta el valor de la entrada "**FINAL**". Es decir, se repite (**FINAL** - **INIT**) veces, siendo **FINAL** mayor o igual a **INIT**.

El loop **FOR** trabaja con el sistema de funciones Ladder. Para utilizarlo, primero defina una función con un diagrama conteniendo la lógica del loop. Luego desde las propiedades del componente, seleccione la función a llamar desde el componente.

La función Ladder definida, recibirá en cada llamada el número de repetición (iteración) de cada ciclo como valor de argumento "**InVal**". Puede usar el valor de **InVal** dentro del diagrama para conocer en que numero de ciclo se encuentra.

Si la función definida, retorna el numero "-1", el loop **FOR** se aborta sin importar si la cuenta de repeticiones ha llegado hasta el valor "**FINAL**".

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

13.1.2 Notas:

- Es posible anidar ciclos **FOR**. Para hacerlo, llame dentro de los diagramas a otro ciclo **FOR**.
- El valor **FINAL** debe ser mayor que **INIT**, para que el ciclo se ejecute al menos una vez.

14 Componentes para Display LCD (Texto)

Componentes para controlar el funcionamiento o imprimir texto en display LCD incluido en algunos modelos de PLC.

14.1 LcdTextClear (Limpiar Texto del Display)

Agrega un componente que permite limpiar el texto de un display LCD.

Este componente debe llamarse solo cuando sea necesario limpiar el display. Si se llama repetidamente a una frecuencia muy alta, puede generar un efecto visual llamado flickering (parpadeo) en el display LCD, que puede ser antiestético a fines prácticos.

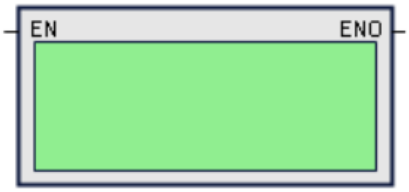
El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

14.1.1 Notas:

- Este componente puede insumir hasta 2 mS de tiempo al ejecutarse debido a características internas del display LCD. Por lo que es un componente de ejecución relativamente lenta y debe llamarse solo cuando es necesario.
- Algunos modelos de PLC no incorporan display LCD.

14.2 LcdTextPrint (Imprimir Texto en Display)



Agrega un componente que permite imprimir mensajes de texto en un display LCD.

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

14.2.1 Funcionamiento:

El texto a imprimir en cada línea del display se especifica en las propiedades del componente.

Si el mensaje excede la longitud del display, los caracteres se escriben en la línea siguiente disponible o al comienzo del display.

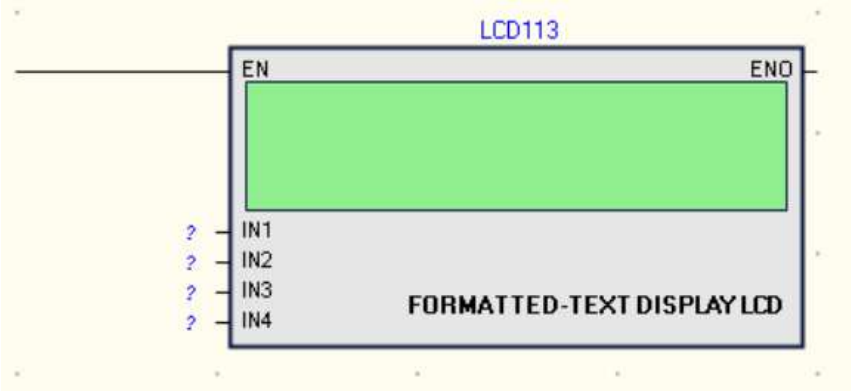
Para un correcto uso del componente, debe especificar el tamaño del display LCD (Líneas x Columnas) en las propiedades del mismo.

Es posible limpiar el LCD antes de escribir, pero esta opción puede retardar la ejecución del código 2 mS. La opción "***Limpiar antes de escribir***" puede utilizarse para tal fin.

14.2.2 Consejos:

- Para imprimir texto con formato, es decir mensajes y valores de variables combinados, utilice el componente "***LcdTextPrintf***". El tiempo de ejecución de dicho componente es superior.
- Puede limpiar el LCD imprimiendo espacios " " en todas las líneas del display. Este método es más eficiente que utilizar la opción "***Limpiar antes de imprimir***".

14.3 LcdTextPrintf (Imprimir Texto con Formato en Display)



Agrega un componente que permite imprimir mensajes de texto con formato en un display LCD. La impresión con formato permite mezclar texto con valores de variables en la impresión de mensajes.

Si busca imprimir el valor de una variable, este es el componente indicado.

El texto a imprimir en cada línea del display se especifica en las propiedades del componente.

Si el mensaje excede la longitud del display, los caracteres se escriben en la línea siguiente disponible o al comienzo del display.

Para un correcto uso del componente, debe especificar el tamaño del display LCD (Líneas x Columnas) en las propiedades del mismo.

Es posible limpiar el LCD antes de escribir, pero esta opción puede retardar la ejecución del código 2 mS. La opción "***Limpiar antes de imprimir***" puede utilizarse para tal fin.

14.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- Existen 4 entradas más, cuyo uso es opcional y permiten especificar las variables a leer e imprimir en la pantalla del LCD.

  Los puertos de entrada son "**IN1**", "**IN2**", "**IN3**" e "**IN4**". Dentro de las propiedades del componente, puede especificar las variables apuntadas por dichos puertos.

  No es necesario utilizar todos los puertos al mismo tiempo, ni seguir un orden especial.

  El tipo de variable o constante que puede seleccionar es amplio, puede ser **Int32**, **Float**, **String**, etc.

  Lea la sección de formato para imprimir valores de variables.

14.3.2 Salidas:

- El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

14.3.3 Formato:

Para imprimir mensajes con formato, es necesario especificar "***códigos de formato***" en el mensaje.

Los ***códigos de formato*** indican cómo debe interpretarse e imprimirse una variable.

Los ***códigos de formato*** están precedidos por un símbolo "%". Por ejemplo "%d" indica que en ese lugar del texto, debe imprimirse una variable entera.

Cada "***código de formato***" lee la variable apuntada por algún puerto de entrada. Los puertos que deben leerse se especifican en las propiedades del componente, en una caja de texto adyacente al mensaje a imprimir.

Ejemplo, para imprimir los RPM de un motor en el display LCD, cuyo valor se especifica en una variable entera apuntada por el puerto "**IN1**", hacemos:

"RPM = %d" → "IN1"

En el display, se reemplazará el ***código de formato*** %d por el valor de la variable apuntada por el puerto "**IN1**".

Si deseamos imprimir el valor de RPM y PRESION, podemos hacer:

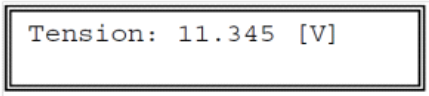
"RPM = %d PRESION = %d" → "**IN1**,**IN2**"

Notemos, como separamos con comas los puertos **IN1** e **IN2**.

Si tenemos un valor de tensión y deseamos imprimirlo como punto flotante, podemos hacer:

"Tension: %.3f [V]" → "**IN1**"

El resultado será el siguiente:



Note como fue especificada la precisión, mostrando solo 3 dígitos decimales.

14.3.4 Códigos de formatos disponibles:

Nombre	Tipo	Descripción
%d	Entero	Imprime un entero con signo.
%u	Entero	Imprime un entero sin signo.
%x	Entero	Imprime un entero con notación hexadecimal.
%X	Entero	Imprime un entero con notación hexadecimal en mayúscula.
%f	Flotante	Imprime un numero decimal o punto flotante.
%s	Cadena	Imprime una cadena de caracteres.
%c	Caracter	Imprime un solo carácter.

Un Entero es un valor tipo **Int32**.
Un Flotante es un valor tipo **Float**
Una Cadena es un valor tipo **String**.

14.3.5 Alineación y Relleno:

Se antepone un numero entre el "%" y el código. El número especifica la cantidad de espacios que deberán ser rellenados antes de imprimir la variable. Si el digito es negativo "-", la alineación será hacia la izquierda.

Si antes del digito de alineación, se antepone el numero 0, en vez de rellenar con espacios, la alineación será rellenada con ceros.

14.3.6 Precisión:

En un número flotante, es posible especificar la cantidad de dígitos decimales a imprimir (por defecto es dos). Por ejemplo, "%.000f", imprimimos un numero decimal con tres dígitos luego del punto solamente

Ejemplos de Formatos:

Ejemplo	Argumento	Salida en Display (cada columna es un carácter)							
%d	3	3							
%4d	3				3				
%04d	3	0	0	0	3				
%-4d	3	3							
%04d	-3	-	0	0	3				
%4d	-3			-	3				
%03.000f	5.12345	0	5	.	1	2	3		
%f	5.12345	5	.	1	2				
%.4f	5.12345	5	.	1	2	3	4	5	
%.02f	3.1	3	.	1	0				

Ejemplo	Argumento	Salida en Display (cada columna es un carácter)							
%x	255	f	f						
%X	255	F	F						
%s	"HOLA"	H	O	L	A				
%c	66	B							
%c	'B'	B							
%s	0	(	n	u	l	l	)		

En las tablas, el término "Argumento" se refiere al valor del puerto (**IN1**, **IN2**, etc).

14.3.7 Notas / Consejos:

- No se pueden imprimir líneas de texto vacías o que solo imprimen espacios. En dicho caso, no será generado código para la impresión de la línea vacía. Note que algunas líneas pueden estar vacías, mientras que otras no, lo cual es permitido.
- Para imprimir texto sin formato, es decir mensajes únicamente, utilice el componente "***LcdTextPrint***". El tiempo de ejecución de dicho componente es inferior.

- Si usted es un programador en lenguaje C, el uso de este componente le resultara similar a la función "**printf()"** de dicho lenguaje.

- Algunos PLC no disponen de un display LCD incorporado, pero puede utilizar los componentes NetHMI.

15 Componentes NetHMI (Network Hunman Machine Interface)

Los componentes NetHMI le permiten utilizar un panel remoto HMI (Interfaz Hombre Maquina) conectado a la red Ethernet.

El panel HMI remoto puede ser un dispositivo de hardware o una aplicación por software corriendo en una computadora tipo PC.

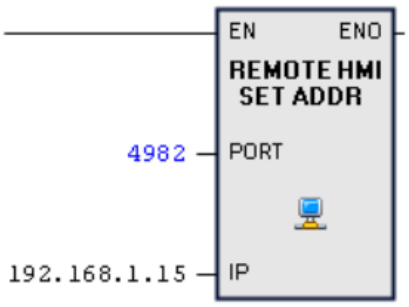
Además, se brindan librerías en Microsoft Visual C# para crear sus propios paneles HMI en Windows e integrarlos a los PLC de Slicetex Electronics.

Los componentes NetHMI le permitirán introducir y mostrar datos del PLC en tiempo real de forma muy simple.

Finalmente, puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Más información en nuestro sitio Web.

Nota: Si solo desea enviar o recibir datos vía Ethernet, consulte la página 140.

15.1 NetHmiSetAddress (Establecer Dirección IP del Panel HMI)



Agrega un componente que permite configurar los parámetros de conexión hacia un terminal HMI (Human Machine Interface - Interfaz Hombre Maquina) remoto conectado a la red Ethernet del PLC.

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

15.1.1 Funcionamiento:

- En la entrada "**PORT**" del componente se especifica en una variable o constante tipo **Int32** el puerto UDP del terminal HMI remoto.
- En la entrada "**IP**" del componente se especifica la dirección IP del terminal HMI remoto.
- El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

15.1.2 Importante:

- Es necesario llamar al menos una vez este componente en el programa antes de utilizar los componentes NetHMI para transmitir datos al terminal remoto.

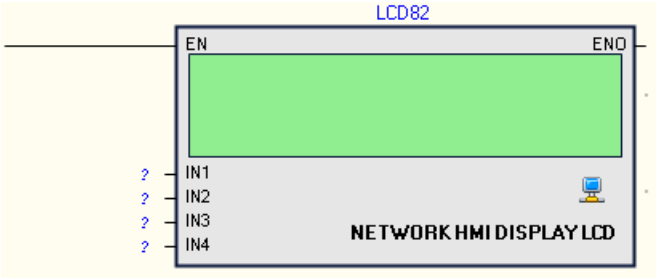
15.1.3 Notas:

- UDP (User Datagram Protocol) es un protocolo estándar de internet para la transmisión de datos sin handshaking.
- IP (Internet Protocol) es un protocolo estándar de internet, cuyo direccionamiento es una ***dirección IP*** que se utiliza para identificar inequívocamente un dispositivo conectado a la red. Una dirección IP puede ser por ejemplo: 192.168.1.15.
- Un puerto UDP es un número entero que va desde 0 a 65.535. El puerto 0 está reservado y no es aconsejable utilizar ninguno hasta el numero 1023. Un valor utilizado comúnmente en nuestros dispositivos el 4982, pero puede utilizar otro si encuentra alguna interferencia.

15.1.4 Consejos:

- Puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Adicionalmente se provee una librería gratuita para crear sus propios HMI con *Microsoft Visual C#*. Más información en nuestro sitio Web.

15.2 NetHmiLcdPrintf (Imprimir Texto con Formato en Display)



Agrega un componente que permite imprimir mensajes de texto con formato en un display HMI (Human Machine Interface) remoto conectado a la red Ethernet. La impresión con formato permite mezclar texto con valores de variables en la impresión de mensajes.

Si busca imprimir el valor de una variable, este es el componente indicado.

El texto a imprimir en cada línea del display se especifica en las propiedades del componente.

Es posible limpiar el LCD antes de escribir mediante la opción "***Limpiar antes de imprimir***" que puede utilizarse para tal fin. Tenga en cuenta que si limpia e imprime repetidamente el display, puede generar un efecto visual llamado flickering (parpadeo). Esto puede evitarse utilizando el componente "**NetHmiLcdClear**" y utilizándolo solo cuando sea necesario borrar la pantalla.

Importante:

- Es necesario configurar los parámetros de conexión NetHMI en el programa antes de utilizar los componentes NetHMI para transmitir datos al terminal remoto. Utilice el componente "**NetHmiSetAddress**".

15.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- Existen 4 entradas más, cuyo uso es opcional y permiten especificar las variables a leer e imprimir en la pantalla del HMI.

  Los puertos de entrada son "**IN1**", "**IN2**", "**IN3**" e "**IN4**". Dentro de las propiedades del componente, puede especificar las variables apuntadas por dichos puertos.

  No es necesario utilizar todos los puertos al mismo tiempo, ni seguir un orden especial.

  El tipo de variable o constante que puede seleccionar es amplio, puede ser **Int32**, **Float**, **String**, etc.

  Lea la sección de formato para imprimir valores de variables.

15.2.2 Salidas:

- El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

15.2.3 Formato:

Para imprimir mensajes con formato, es necesario especificar "***códigos de formato***" en el mensaje.

Los ***códigos de formato*** indican cómo debe interpretarse e imprimirse una variable.

Los ***códigos de formato*** están precedidos por un símbolo "%". Por ejemplo "%d" indica que en ese lugar del texto, debe imprimirse una variable entera.

Cada "***código de formato***" lee la variable apuntada por algún puerto de entrada. Los puertos que deben leerse se especifican en las propiedades del componente, en una caja de texto adyacente al mensaje a imprimir.

Ejemplo, para imprimir los RPM de un motor en el display HMI, cuyo valor se especifica en una variable entera apuntada por el puerto "**IN1**", hacemos:

"RPM = %d" → "IN1"

En el display, se reemplazará el ***código de formato*** %d por el valor de la variable apuntada por el puerto "**IN1**".

Si deseamos imprimir el valor de RPM y PRESION, podemos hacer:

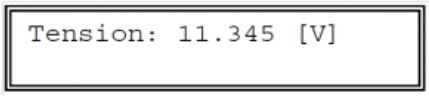
"RPM = %d PRESION = %d" → "**IN1**,**IN2**"

Notemos, como separamos con comas los puertos **IN1** e **IN2**.

Si tenemos un valor de tensión y deseamos imprimirlo como punto flotante, podemos hacer:

"Tension: %.3f [V]" → "**IN1**"

El resultado será el siguiente:



Note como fue especificada la precisión, mostrando solo 3 dígitos decimales.

15.2.4 Códigos de formatos disponibles:

Nombre	Tipo	Descripción
%d	Entero	Imprime un entero con signo.
%u	Entero	Imprime un entero sin signo.
%x	Entero	Imprime un entero con notación hexadecimal.
%X	Entero	Imprime un entero con notación hexadecimal en mayúscula.
%f	Flotante	Imprime un numero decimal o punto flotante.
%s	Cadena	Imprime una cadena de caracteres.
%c	Caracter	Imprime un solo carácter.

Un Entero es un valor tipo **Int32**.
Un Flotante es un valor tipo **Float**
Una Cadena es un valor tipo **String**.

15.2.5 Alineación y Relleno:

Se antepone un numero entre el "%" y el código. El número especifica la cantidad de espacios que deberán ser rellenados antes de imprimir la variable. Si el digito es negativo "-", la alineación será hacia la izquierda.

Si antes del digito de alineación, se antepone el numero 0, en vez de rellenar con espacios, la alineación será rellenada con ceros.

15.2.6 Precisión:

En un número flotante, es posible especificar la cantidad de dígitos decimales a imprimir (por defecto es dos). Por ejemplo, "%.000f", imprimimos un numero decimal con tres dígitos luego del punto solamente

Ejemplos de Formatos:

Ejemplo	Argumento	Salida en Display (cada columna es un carácter)							
%d	3	3							
%4d	3				3				
%04d	3	0	0	0	3				
%-4d	3	3							
%04d	-3	-	0	0	3				
%4d	-3			-	3				
%03.000f	5.12345	0	5	.	1	2	3		
%f	5.12345	5	.	1	2				
%.4f	5.12345	5	.	1	2	3	4	5	
%.02f	3.1	3	.	1	0				

Ejemplo	Argumento	Salida en Display (cada columna es un carácter)							
%x	255	f	f						
%X	255	F	F						
%s	"HOLA"	H	O	L	A				
%c	66	B							
%c	'B'	B							
%s	0	(	n	u	l	l	)		

En las tablas, el término "Argumento" se refiere al valor del puerto (**IN1**, **IN2**, etc).

15.2.7 Notas / Consejos:

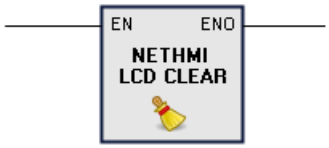
- No se pueden imprimir líneas de texto vacías o que solo imprimen espacios. En dicho caso, no será generado código para la impresión de la línea vacía. Note que algunas líneas pueden estar vacías, mientras que otras no, lo cual es permitido.
- Si usted es un programador en lenguaje C, el uso de este componente le resultara similar a la función "**printf()"** de dicho lenguaje.

- Algunos PLC no disponen de un display LCD incorporado, por ello puede resultarle muy útil este componente para mostrar información del PLC de forma simple!.

- No es conveniente llamar a este componente desde el diagrama “Inicio” debido a que el hardware Ethernet del PLC puede estar inicializándose y no se transmitirán datos por la red.

- Puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Adicionalmente se provee una librería gratuita para crear sus propios HMI con *Microsoft Visual C#*. Más información en nuestro sitio Web.

15.3 NetHmiClear (Limpiar Display)



Agrega un componente que permite limpiar el texto de un display HMI (Human Machine Interface) remoto conectado a la red Ethernet.

Este componente debe llamarse solo cuando sea necesario limpiar el display HMI. Si se llama repetidamente a una frecuencia muy alta, puede generar un efecto visual llamado flickering (parpadeo) en el display HMI, que puede ser antiestético a fines prácticos.

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

Importante:

- Es necesario configurar los parámetros de conexión NetHMI en el programa antes de utilizar los componentes NetHMI para transmitir datos al terminal remoto. Utilice el componente "**NetHmiSetAddress**".

15.3.1 Notas / Consejos:

- No es conveniente llamar a este componente desde el diagrama "Inicio" debido a que el hardware Ethernet del PLC puede estar inicializándose y no se transmitirán datos por la red.

- Puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Adicionalmente se provee una librería gratuita para crear sus propios HMI con *Microsoft Visual C#*. Más información en nuestro sitio Web.

15.4 NetHmiSetEvent (Activar Evento NetHMI)

Agrega un componente que permite activar el evento "**OnNetHMI**".

El evento **OnNetHMI** será llamado cuando se reciban datos desde un dispositivo HMI conectado a la red Ethernet del PLC.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

El evento **OnNetHMI** es un mecanismo muy útil para interactuar con terminales HMI remotos de forma muy simple.

15.4.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

15.4.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el evento pudo configurarse con éxito.

15.4.3 Notas:

- Recuerde crear el diagrama para procesar el evento.
- Una vez activado el evento, el PLC aceptara conexiones de terminales HMI conectados a la red Ethernet.

15.4.4 Consejos:

- Puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Adicionalmente se provee una librería gratuita para crear sus propios HMI con *Microsoft Visual C#*. Más información en nuestro sitio Web.

15.5 NetHmiClrEvent (Desactivar Evento NetHMI)

Agrega un componente que permite desactivar el evento "***OnNetHMI***".

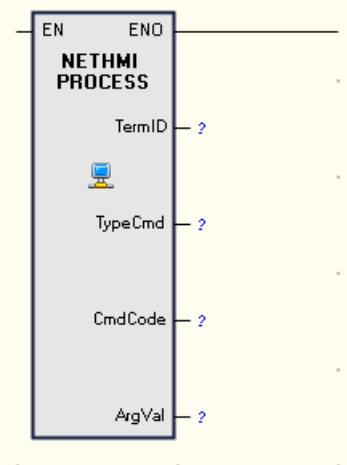
El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

Importante:

- Al desactivar el evento, todas las conexiones provenientes de los terminales HMI conectados a la red Ethernet del PLC serán rechazadas.

15.6 NetHmiDataProcess (Procesar Datos NetHMI)



Agrega un componente que permite procesar un paquete de datos proveniente de un terminal HMI remoto conectado a la red Ethernet del PLC.

Cuando un paquete con datos proveniente de un terminal HMI remoto es recibido por el PLC y el evento **OnNetHMI** esta activado, se procede a ejecutar el manejador de evento que pudo definirse como un diagrama dentro del entorno **StxLadder**.

Usualmente, este componente se llama una sola vez al principio del diagrama que maneja el evento **OnNetHMI**. Cuando el PLC ejecuta este componente, los datos provenientes del terminal HMI son decodificados y almacenados en variables para su fácil lectura y procesamiento.

La misión principal de este componente es la de procesar los datos provenientes de un terminal HMI remoto y depositarlos en variables de simple lectura.

Luego con el componente ***NetHmiCmdCheck*** podrá comprobar fácilmente el valor de las variables.

15.6.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

15.6.2 Salidas:

- El valor del puerto de salida "**ENO**" es 1 o verdadero, cuando los datos provenientes del terminal HMI fueron procesados correctamente. Si es falso, puede deberse a que no hay datos recibidos, el paquete es erróneo, o existe algún error en una variable ingresada.

- En el puerto de entrada "**TermID**" se especifica una variable tipo **Int32**, donde se almacenará el **número de identificación** de terminal HMI que envió los datos. Este número es definido por el terminal en el lugar de origen y sirve para diferenciar distintos terminales. El rango de valores va desde 0 a 255.

- En el puerto de entrada "**TypeCmd**" se especifica una variable tipo **Int32**, donde se almacenará el **tipo de comando** recibido. Este número permite saber a qué aplicación o función corresponden los datos recibidos. Por ejemplo, pueden ser del tipo de una tecla de función o de una barra analógica.

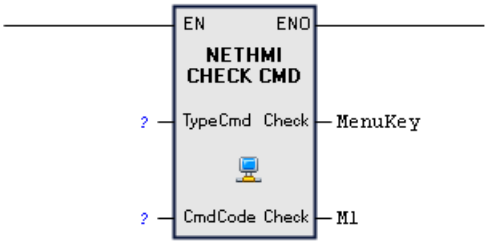
- En el puerto de entrada "**CmdCode**" se especifica una variable tipo **Int32**, donde se almacenará el **código de comando** recibido. Este número permite conocer que comando envía el terminal HMI remoto. Por ejemplo un número de tecla.

- En el puerto de entrada "**ArgVal**" se especifica una variable tipo **Int32**, donde se almacenará un valor asociado por el **código de comando** recibido. Este valor no siempre es definido por los diferentes comandos del terminal HMI remoto ya que es opcional y puede o no ser enviado por el terminal.

15.6.3 Notas / Consejos:

- Recuerde crear el diagrama para procesar el evento **OnNetHMI**.
- Con el componente ***NetHmiCmdCheck*** podrá comprobar fácilmente el valor de las variables y tomar decisiones lógicas o recibir datos del terminal HMI remoto.
- Puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Adicionalmente se provee una librería gratuita para crear sus propios HMI con *Microsoft Visual C#*. Más información en nuestro sitio Web.

15.7 NetHmiCmdCheck (Comprobar Comando NetHMI)



Agrega un componente que permite comprobar **tipos de comandos** y **códigos de comandos** provenientes de un terminal HMI remoto conectado a la red Ethernet del PLC.

El objetivo principal del componente es el de comprobar rápidamente los datos recibidos del terminal HMI y tomar decisiones lógicas.

Usualmente, este componente llama DESPUES de procesar los datos recibidos del terminal HMI remoto con el componente **NetHmiDataProcess** en diagrama que maneja el evento **OnNetHMI**.

Recuerde que el diagrama que maneja el evento **OnNetHMI** es activado cada vez que se recibe un paquete con datos del terminal HMI remoto.

15.7.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- El puerto de entrada "**TypeCmd"** admite una variable o constante del tipo **Int32** que contiene el **tipo de comando** enviado por el terminal HMI.
- El puerto de entrada "**CmdCode"** admite una variable o constante del tipo **Int32** que contiene el **código de comando** enviado por el terminal HMI.

15.7.2 Salidas:

- El primer puerto de salida "**Check"**, no retorna ningún valor, pero permite comprobar el valor de el puerto de entrada **TypeCmd**. Desde las propiedades del componente, es posible seleccionar la constante con la cual queremos comprobar el puerto de entrada **TypeCmd**.
- El segundo puerto de salida "**Check"**, no retorna ningún valor, pero permite comprobar el valor de el puerto de entrada **CmdCode**. Desde las propiedades del componente, es posible seleccionar la constante con la cual queremos comprobar el puerto de entrada **CmdCode**.
- El valor del puerto de salida "**ENO**" retorna "1" solo si los valores de ambos puertos de entradas coinciden con los valores de ambos puertos de salida. De lo contrario retorna el valor "0" o falso.

Cuando el componente es ejecutado, comprueba ambos valores de los puertos de entradas con los valores de los puertos de salida, si coinciden, el resultado del puerto **ENO** es 1 y usted puede implementar la lógica correspondiente, como por ejemplo activar un relé o imprimir un mensaje en un display.

En las propiedades del componente es posible seleccionar todos los **tipos y códigos comandos** admitidos provenientes de un terminal remoto HMI.

Algunos **tipos y códigos de comandos** tienen asociados un valor de argumento, llamado **ArgVal**. Dicho valor fue adquirido con el componente **NetHmiDataProcess** y almacenado apropiadamente en el diagrama del evento

OnNetHMI. Por ejemplo, si **TypeCmd** = AnalogControl y **CmdCode** = TrackBar1, la salida **ENO** es 1, y por lo tanto usted sabe que en **ArgVal** hay un valor analógico que corresponde a una barra analógica. Consulte la documentación de la interface **NetHMI** o **VirtualHMI**.

15.7.3 Ejemplos:

- Si **TypeCmd** = MenuKey y **CmdCode** = M1, la salida ENO es 1, y sabemos que el paquete recibido corresponde a que un usuario apretó la tecla de menú M1 en el terminal HMI remoto.
- Si **TypeCmd** = FunctionKey y **CmdCode** = F1, la salida ENO es 1, y sabemos que el paquete recibido corresponde a que un usuario apretó la tecla de función M1 en el terminal HMI remoto.
- Si **TypeCmd** = NumericValue y **CmdCode** = Int32, la salida ENO es 1, y sabemos que el paquete recibido corresponde a que un usuario envio un valor tipo Int32 desde el terminal HMI remoto, y está almacenado en **ArgVal**.
- Si **TypeCmd** = AnalogControl y **CmdCode** = TrackBar1, la salida ENO es 1, y sabemos que el paquete recibido corresponde a que un usuario envió un valor tipo **Int32** que corresponde a una dato analógico (potenciómetro, barra de desplazamiento, temperatura, etc) desde el terminal HMI remoto, y está almacenado en **ArgVal**.

15.7.4 Notas / Consejos:

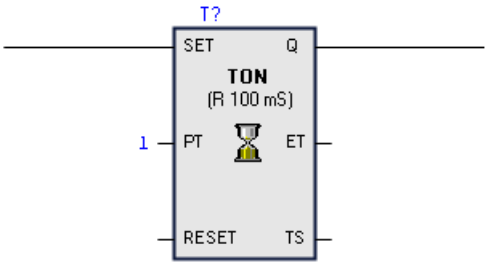
- Recuerde crear el diagrama para procesar el evento **OnNetHMI**.
- Con el componente **NetHmiDataProcess** obtiene los datos recibidos del terminal HMI remoto.
- Puede utilizar el programa ***VirtualHMI*** de Slicetex Electronics para utilizar su PC como un terminal HMI remoto sin costo. Adicionalmente se provee una librería gratuita para crear sus propios HMI con *Microsoft Visual C#*. Más información en nuestro sitio Web.
- Si bien los puertos de salida **Check** no retornan valores, permiten visualmente conocer rápidamente con que valores estamos comprobando las entradas. Ese fue el criterio de diseño utilizado para desarrollar este componente.

16 Componentes Temporizadores Ladder

Los temporizadores Ladder, son componentes que permiten realizar acciones de control de tiempo dentro de los diagramas Ladder. A diferencia de otros temporizadores, los temporizadores Ladder fueron diseñados para su uso en esquemas eléctricos. Muchos de los componentes descriptos en esta sección son “estándar” en la industria del PLC.

Los PLC de Slicetex Electronics soportan temporizadores Ladder con resoluciones de hasta 1 mS.

16.1 TimerTON (Temporizador con Retardo a la Conexión)



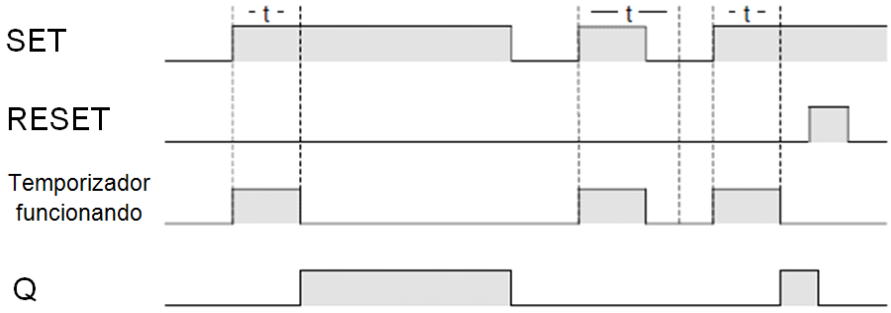
Agrega un componente que actúa como un temporizador (timer) del tipo **TON** con retardo a la conexión.

16.1.1 Funcionamiento general:

El temporizador arranca cuando hay un flanco creciente en la entrada de arranque **SET**. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador continúa en marcha con el valor de temporización indicado en la entrada **PT** (Preset Time) mientras el estado de señal en la entrada **SET** sea positivo.

El estado de señal en la salida **Q** será "**1**" si el tiempo ha transcurrido sin que se produjeran errores y si el estado de señal en la entrada **SET** es "**1**". Si el estado de señal en la entrada **SET** cambia de “**1**" a "**0**" mientras está en marcha el temporizador, éste se para. En este caso, el estado de señal en la salida **Q** será "**0**".

El temporizador se pone a “**0**” si la entrada de desactivación **RESET** del temporizador se pone a "**1**" mientras funciona el temporizador. El valor de temporización se pone a “**0**”. Entonces el estado de señal en la salida **Q** es "**0**". El temporizador también se pone a “**0**” si en la entrada de desactivación **RESET** el valor es "**1**", mientras el temporizador no está en marcha y la entrada **SET** es "**1**".



16.1.2 Entradas:

- La entrada **SET** activa el temporizador cuando hay una transición de 0 a 1 en el flujo de corriente.
- La entrada **PT** (Preset Time) permite establecer el tiempo de temporización, leyendo una variable o constante del tipo **Int32**. La resolución del depende del número de temporizador utilizado.
- La entrada **RESET** y admite una variable del tipo **Bool**. Si es "1", desactiva el temporizador, provoca que la salida **Q** y **TS** sea 0. Su uso es opcional.

16.1.3 Salidas:

- La salida **Q** es del tipo **Bool** y su valor depende del funcionamiento del temporizador.
- La salida **ET** (Elapsed Time) deposita en una variable del tipo **Int32** el tiempo transcurrido del temporizador. Para obtener el valor real, debe multiplicarse por la resolución del temporizador. Su uso es opcional.
- La salida **TS** (Timer Status) permite copiar el valor de la salida **Q** en una variable del tipo **Bool**. Su uso es opcional.

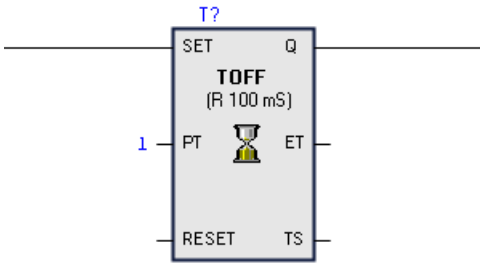
16.1.4 Notas:

- La resolución del temporizador depende del número de temporizador seleccionado. En las propiedades del componente es posible seleccionar el numero de temporizador y al mismo tiempo observar la resolución en mili-segundos utilizada por el temporizador.
- Las resoluciones de los temporizadores pueden ser 1 mS, 10 mS y 100 mS.
- No se puede repetir el número de temporizador en otro componente del tipo temporizador. Un error de alerta se generara si hay repetición.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.1.5 Consejos:

- Hay componentes que tienen **acceso directo** a la memoria del temporizador y le permiten resetear, obtener estado y ET (Elapsed Time). Vea los componentes tipo: **TimerGetET**, **TimerReset**, **TimerIsRunning** y **TimerIsTimeout**.
- Si necesita que un temporizador se resetee antes que el PLC ejecute el componente en el próximo ciclo SCAN, utilice el componente **TimerReset** para obtener acceso directo al mismo. Esto puede resultar útil en casos donde se requiera que el temporizador no sea ejecutado o se necesite reiniciarlo antes del próximo ciclo.

16.2 TimerTOFF (Temporizador con Retardo a la Desconexión)



Agrega un componente que actúa como un temporizador (timer) del tipo **TOFF** con retardo a la desconexión.

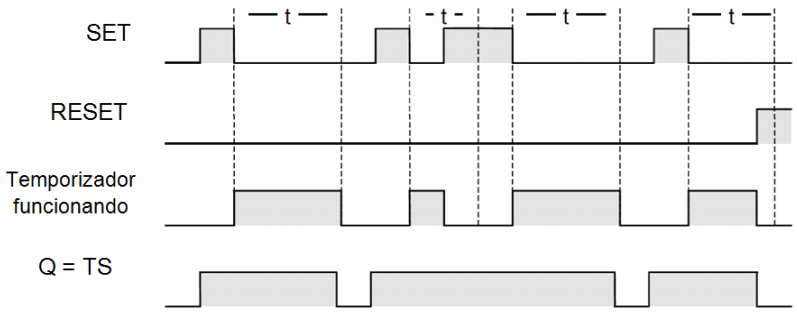
16.2.1 Funcionamiento general:

Arranca el temporizador indicado cuando hay un flanco decreciente en la entrada de arranque **SET**. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal.

El estado de señal en la salida **Q** será "**1**" si el estado de señal en la entrada **SET** es "**1**", y también mientras el temporizador esté en marcha. El temporizador se para sí el estado de señal en la entrada **SET** cambia de "**0**" a "**1**" mientras el temporizador está en marcha.

El temporizador sólo vuelve a arrancar si el estado de señal en la entrada **SET** vuelve a cambiar de "**1**" a "**0**".
La salida **Q** del temporizador se pone a "**0**" si la entrada de desactivación **RESET** se pone a "**1**" mientras el temporizador está en marcha.

El valor de temporización se especifica en la entrada **PT** (Preset Time).
En la salida **ET** (Elapsed Time) se retorna el tiempo transcurrido.
La Salida **TS** (Timer Status) copia el estado de la salida **Q** en una variable.



16.2.2 Entradas:

- La entrada **SET** activa el temporizador cuando hay una transición de 1 a 0 en el flujo de corriente.
- La entrada **PT** (Preset Time) permite establecer el tiempo de temporización, leyendo una variable o constante del tipo **Int32**. La resolución del depende del número de temporizador utilizado.
- La entrada **RESET** y admite una variable del tipo **Bool**. Si es "1", desactiva el temporizador, provoca que la salida **Q** y **TS** sea 0. Su uso es opcional.

16.2.3 Salidas:

- La salida **Q** es del tipo **Bool** y su valor depende del funcionamiento del temporizador.
- La salida **ET** (Elapsed Time) deposita en una variable del tipo **Int32** el tiempo transcurrido del temporizador. Para obtener el valor real, debe multiplicarse por la resolución del temporizador. Su uso es opcional.
- La salida **TS** (Timer Status) permite copiar el valor de la salida **Q** en una variable del tipo **Bool**. Su uso es opcional.

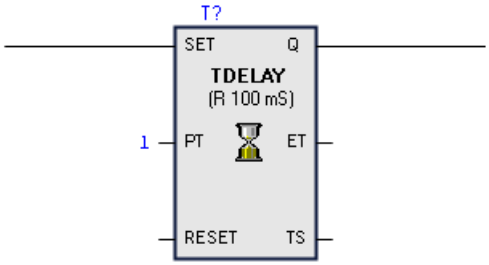
16.2.4 Notas:

- La resolución del temporizador depende del número de temporizador seleccionado. En las propiedades del componente es posible seleccionar el numero de temporizador y al mismo tiempo observar la resolución en mili-segundos utilizada por el temporizador.
- Las resoluciones de los temporizadores pueden ser 1 mS, 10 mS y 100 mS.
- No se puede repetir el número de temporizador en otro componente del tipo temporizador. Un error de alerta se generara si hay repetición.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.2.5 Consejos:

- Hay componentes que tienen **acceso directo** a la memoria del temporizador y le permiten resetear, obtener estado y ET (Elapsed Time). Vea los componentes tipo: **TimerGetET**, **TimerReset**, **TimerIsRunning** y **TimerIsTimeout**.
- Si necesita que un temporizador se resetee antes que el PLC ejecute el componente en el próximo ciclo SCAN, utilice el componente **TimerReset** para obtener acceso directo al mismo. Esto puede resultar útil en casos donde se requiera que el temporizador no sea ejecutado o se necesite reiniciarlo antes del próximo ciclo.

16.3 TimerTDELAY (Temporizador de Retardos)



Agrega un componente que actúa como un temporizador (timer) del tipo **TDELAY** que permite generar retardos (delays) en el programa.

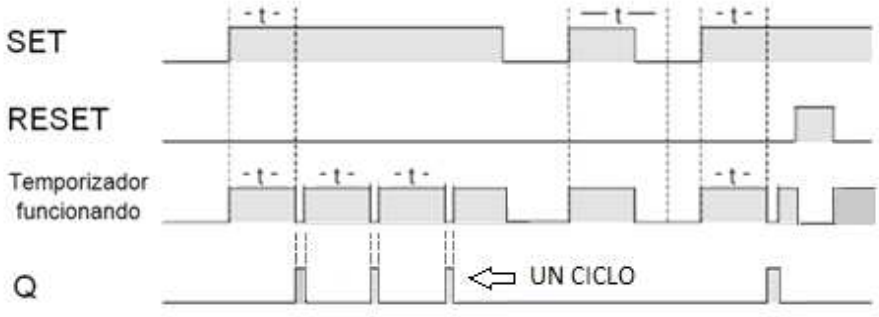
16.3.1 Funcionamiento general:

El temporizador arranca cuando hay un flanco creciente en la entrada de arranque **SET**. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador continúa en marcha con el valor de temporización indicado en la entrada **PT** (Preset Time) mientras el estado de señal en la entrada **SET** sea positivo.

El estado de señal en la salida **Q** será "**1**" si el tiempo ha transcurrido sin que se produjeran errores y si el estado de señal en la entrada **SET** es "**1**". Si el estado de señal en la entrada **SET** cambia de "**1**" a "**0**" mientras está en marcha el temporizador, éste se para. En este caso, el estado de señal en la salida **Q** será "**0**".

La salida **Q** del temporizador será "**1**" solo durante un ciclo de SCAN del PLC. En este caso el temporizador se resetea automáticamente y comienza a temporizar de nuevo. Si la entrada **SET** es "**1**" en el próximo ciclo de SCAN, el temporizador arrancará automáticamente sin necesidad de un flanco creciente de "**0**" a "**1**". Esto permite generar un oscilador.

El temporizador se pone a "**0**" si la entrada de desactivación **RESET** del temporizador se pone a "**1**" mientras funciona el temporizador. El valor de temporización se pone a "**0**". Entonces el estado de señal en la salida **Q** es "**0**". El temporizador también se pone a "**0**" si en la entrada de desactivación **RESET** el valor es "**1**", mientras el temporizador no está en marcha y la entrada **SET** es "**1**".



16.3.2 Entradas:

- La entrada **SET** activa el temporizador cuando hay una transición de 0 a 1 en el flujo de corriente.
- La entrada **PT** (Preset Time) permite establecer el tiempo de temporización, leyendo una variable o constante del tipo **Int32**. La resolución del depende del número de temporizador utilizado.
- La entrada **RESET** y admite una variable del tipo **Bool**. Si es "1", desactiva el temporizador, provoca que la salida **Q** y **TS** sea 0. Su uso es opcional.

16.3.3 Salidas:

- La salida **Q** es del tipo **Bool** y su valor depende del funcionamiento del temporizador.
- La salida **ET** (Elapsed Time) deposita en una variable del tipo **Int32** el tiempo transcurrido del temporizador. Para obtener el valor real, debe multiplicarse por la resolución del temporizador. Su uso es opcional.
- La salida **TS** (Timer Status) permite copiar el valor de la salida **Q** en una variable del tipo **Bool**. Su uso es opcional.

16.3.4 Importante:

- Cuando la salida **Q** es "**1**", el valor retornado en la salida **ET** es cero, debido a que el temporizador se auto-reseteo.

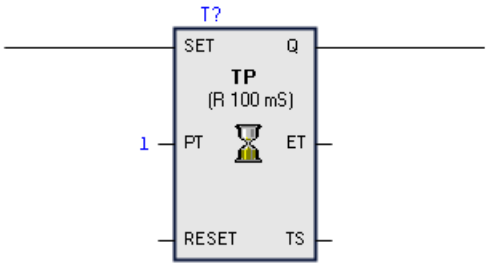
16.3.5 Notas:

- La resolución del temporizador depende del número de temporizador seleccionado. En las propiedades del componente es posible seleccionar el numero de temporizador y al mismo tiempo observar la resolución en mili-segundos utilizada por el temporizador.
- Las resoluciones de los temporizadores pueden ser 1 mS, 10 mS y 100 mS.
- No se puede repetir el número de temporizador en otro componente del tipo temporizador. Un error de alerta se generara si hay repetición.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.3.6 Consejos:

- Hay componentes que tienen **acceso directo** a la memoria del temporizador y le permiten resetear, obtener estado y ET (Elapsed Time). Vea los componentes tipo: **TimerGetET**, **TimerReset**, **TimerIsRunning** y **TimerIsTimeout**.
- Si necesita que un temporizador se resetee antes que el PLC ejecute el componente en el próximo ciclo SCAN, utilice el componente **TimerReset** para obtener acceso directo al mismo. Esto puede resultar útil en casos donde se requiera que el temporizador no sea ejecutado o se necesite reiniciarlo antes del próximo ciclo.

16.4 TimerTP (Temporizador por Impulsos)



Agrega un componente que actúa como un temporizador (timer) del tipo **TP** por impulsos.

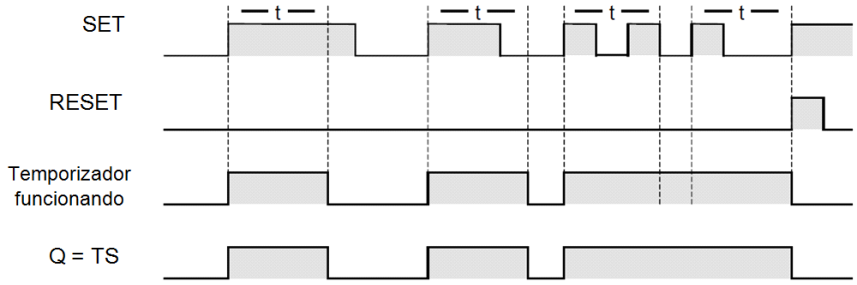
16.4.1 Funcionamiento general:

Arranca el temporizador indicado cuando hay un flanco creciente en la entrada de arranque **SET**. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador continúa en marcha durante el tiempo predeterminado -indicado en la entrada **PT**-, aunque el estado de señal en la entrada **SET** se ponga a "**0**" antes de haber transcurrido el intervalo de tiempo.

El estado de señal en la salida **Q** es "**1**" mientras el temporizador esté en marcha. El temporizador vuelve a arrancar con el valor de temporización predeterminado si el estado de señal en la entrada **SET** cambia de "**0**" a "**1**" mientras está en marcha el temporizador.

La salida **Q** del temporizador se pone a "**0**" si la entrada de desactivación **RESET** del temporizador se pone a "**1**" mientras el temporizador está funcionando. El valor de temporización actual se pone a "**0**".

El valor de temporización se especifica en la entrada **PT** (Preset Time).
En la salida **ET** (Elapsed Time) se retorna el tiempo transcurrido (valor de temporización actual).
La Salida **TS** (Timer Status) copia el estado de la salida **Q** en una variable.



16.4.2 Entradas:

- La entrada **SET** activa el temporizador cuando hay una transición de 0 a 1 en el flujo de corriente.
- La entrada **PT** (Preset Time) permite establecer el tiempo de temporización, leyendo una variable o constante del tipo **Int32**. La resolución del depende del número de temporizador utilizado.
- La entrada **RESET** y admite una variable del tipo **Bool**. Si es "1", desactiva el temporizador, provoca que la salida **Q** y **TS** sea 0. Su uso es opcional.

16.4.3 Salidas:

- La salida **Q** es del tipo **Bool** y su valor depende del funcionamiento del temporizador.
- La salida **ET** (Elapsed Time) deposita en una variable del tipo **Int32** el tiempo transcurrido del temporizador. Para obtener el valor real, debe multiplicarse por la resolución del temporizador. Su uso es opcional.
- La salida **TS** (Timer Status) permite copiar el valor de la salida **Q** en una variable del tipo **Bool**. Su uso es opcional.

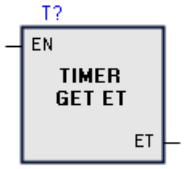
16.4.4 Notas:

- La resolución del temporizador depende del número de temporizador seleccionado. En las propiedades del componente es posible seleccionar el numero de temporizador y al mismo tiempo observar la resolución en mili-segundos utilizada por el temporizador.
- Las resoluciones de los temporizadores pueden ser 1 mS, 10 mS y 100 mS.
- No se puede repetir el número de temporizador en otro componente del tipo temporizador. Un error de alerta se generara si hay repetición.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.4.5 Consejos:

- Hay componentes que tienen **acceso directo** a la memoria del temporizador y le permiten resetear, obtener estado y ET (Elapsed Time). Vea los componentes tipo: **TimerGetET**, **TimerReset**, **TimerIsRunning** y **TimerIsTimeout**.
- Si necesita que un temporizador se resetee antes que el PLC ejecute el componente en el próximo ciclo SCAN, utilice el componente **TimerReset** para obtener acceso directo al mismo. Esto puede resultar útil en casos donde se requiera que el temporizador no sea ejecutado o se necesite reiniciarlo antes del próximo ciclo.

16.5 TimerGetET (Obtener Elapsed Time)



Agrega un componente que obtiene el tiempo transcurrido actual de temporización del timer (temporizador) seleccionado. El tiempo transcurrido actual de temporizador se conoce como Elapsed Time o **ET**.

Este componente **accede directamente** a la memoria del timer, por lo tanto obtiene el valor **inmediato** del timer en el punto de ejecución del diagrama. Se utiliza en casos donde es necesario obtener un valor actualizado de **ET** sin esperar un nuevo SCAN CYCLE del PLC. Muy útil para resoluciones bajas del timer.

16.5.1 Entradas:

- Si el flujo de corriente en la entrada **EN** (Enable) es "**1**", se actualiza la salida **ET** con el valor actual del temporizador. Si la entrada **EN** es "**0**", la salida **ET** no es actualizada (si hay una variable conectada, mantiene su valor previo al llamar al componente).

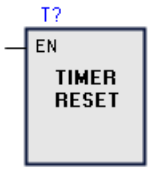
16.5.2 Salidas:

- La salida **ET** (Elapsed Time) deposita en una variable del tipo **Int32** el tiempo transcurrido del temporizador. Para obtener el valor real, debe multiplicarse por la resolución del temporizador.

16.5.3 Notas:

- El número de temporizador se especifica en las propiedades del componente.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.6 TimerReset (Resetear Timer)



Agrega un componente que al ejecutarse produce un RESET en el timer (temporizador) seleccionado de forma inmediata.

El efecto producido es el mismo que aplicar un "**1**" a la entrada RESET de un componente que define un timer.

Este componente **accede directamente** a la memoria del timer, por lo tanto tiene un efecto **inmediato** sobre el timer en el punto de ejecución del diagrama. Se utiliza en casos donde es necesario ***resetear*** el timer sin esperar un nuevo SCAN CYCLE del PLC.

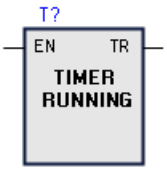
16.6.1 Entradas:

- Si el flujo de corriente en la entrada **EN** (Enable) es "**1**", se ***resetea*** el timer seleccionado.

16.6.2 Notas:

- El número de temporizador se especifica en las propiedades del componente.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.7 TimerIsRunning (Comprobar si Temporizador esta Corriendo)



Agrega un componente que comprueba si está corriendo (o temporizando) el timer seleccionado de forma **inmediata**.

Este componente **accede directamente** a la memoria del timer, por lo tanto obtiene un valor **inmediato** del timer en el punto de ejecución del diagrama.

Se utiliza en casos donde es necesario comprobar si el timer fue habilitado o esta temporizando, sin esperar un nuevo SCAN CYCLE del PLC.

16.7.1 Entradas:

- Si el flujo de corriente en la entrada **EN** (Enable) es "**1**", se ejecuta la operación actualizando la salida **TR**.

16.7.2 Salidas:

- La salida **TR** (Timer Running) es "**1**" si el timer está corriendo o "**0**" si el timer está detenido.

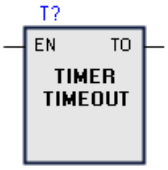
16.7.3 Importante:

- Si la entrada **EN** es "**0**", la salida **TR** es "**0**", independientemente del estado del timer.

16.7.4 Notas:

- El número de temporizador se especifica en las propiedades del componente.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

16.8 TimerIsTimeout (Comprobar si Temporizador ha Finalizado)



Agrega un componente que comprueba si ha finalizado la temporización (Timeout) en el timer seleccionado de forma **inmediata**.

Este componente **accede directamente** a la memoria del timer, por lo tanto obtiene un valor **inmediato** del timer en el punto de ejecución del diagrama.

Se utiliza en casos donde es necesario comprobar si el timer fue habilitado y su cuenta ha llegado al valor **PT** (Preset Time), sin esperar un nuevo SCAN CYCLE del PLC.

16.8.1 Entradas:

- Si el flujo de corriente en la entrada **EN** (Enable) es "**1**", se ejecuta la operación actualizando la salida **TO**.

16.8.2 Salidas:

- La salida **TO** (Timeout) es "**1**" si el timer ha finalizado la temporización o "**0**" si el timer está corriendo todavía.

16.8.3 Importante:

- Si la entrada **EN** es "**0**", la salida **TO** es "**0**", independientemente del estado del timer.

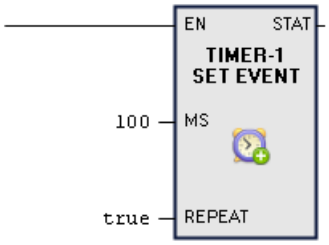
16.8.4 Notas:

- El número de temporizador se especifica en las propiedades del componente.
- La cantidad de temporizadores depende del modelo de PLC seleccionado.

17 Componentes de Temporizadores por Evento

Los temporizadores por Evento permiten generar eventos de temporización asíncronos, es decir, al concluir un tiempo de temporización se llama a un diagrama Ladder que contiene la lógica para procesar el evento. Es de gran utilizad para realizar tareas periódicas que necesitan tiempos muy precisos en el orden de los mili-segundos o cientos de mili-segundos.

17.1 OnTimer1SetEvent (Activar Evento OnTimer1)



Agrega un componente que permite activar el evento "**OnTimer1**".

El evento **OnTimer1** será llamado cada X mili-segundos según como las entradas del componente sean configuradas.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

El evento **OnTimer1** es muy útil para ejecutar tareas periódicas en lapsos precisos de tiempo.

17.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**MS**" se especifican la cantidad de mili-segundos entre cada evento.
- En el puerto "**REPEAT**" se especifica con "true" si el evento se repite cada "**MS**" milisegundos. Si es "false" se generará un solo evento pasado el tiempo del puerto "**MS**".

17.1.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el evento pudo configurarse con éxito.

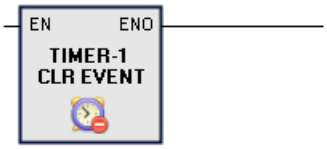
17.1.3 Importante:

- Este componente no puede ser utilizado si algún 'temporizador ladder' (por ejemplo TON, TOFF, TDELAY, etc) es definido o algún componente de aquella librería esta en uso.

17.1.4 Notas:

- Recuerde crear el diagrama para procesar el evento.
- Mientras más pequeño es el valor de "**MS**", el PLC tiene una mayor sobrecarga de procesamiento debido a que debe ejecutar más seguido el evento. Utilice el mayor valor posible de acuerdo a su proyecto.

17.2 OnTimer1ClrEvent (Desactivar Evento OnTimer1)



Agrega un componente que permite desactivar el evento "**OnTimer1**".

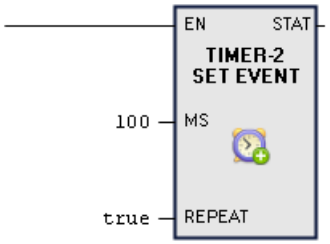
El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

17.2.1 Importante:

- Este componente no puede ser utilizado si algún 'temporizador ladder' (por ejemplo TON, TOFF, TDELAY, etc) es definido o algún componente de aquella librería esta en uso.

17.3 OnTimer2SetEvent (Activar Evento OnTimer2)



Agrega un componente que permite activar el evento "**OnTimer2**".

El evento **OnTimer2** será llamado cada X mili-segundos según como las entradas del componente sean configuradas.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

El evento **OnTimer2** es muy útil para ejecutar tareas periódicas en lapsos precisos de tiempo.

17.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**MS**" se especifican la cantidad de mili-segundos entre cada evento.
- En el puerto "**REPEAT**" se especifica con "true" si el evento se repite cada "**MS**" milisegundos. Si es "false" se generará un solo evento pasado el tiempo del puerto "**MS**".

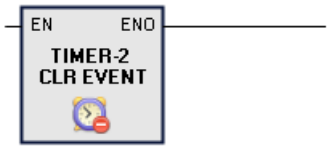
17.3.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el evento pudo configurarse con éxito.

17.3.3 Notas:

- Recuerde crear el diagrama para procesar el evento.
- Mientras más pequeño es el valor de "**MS**", el PLC tiene una mayor sobrecarga de procesamiento debido a que debe ejecutar más seguido el evento. Utilice el mayor valor posible de acuerdo a su proyecto.

17.4 OnTimer2ClrEvent (Desactivar Evento OnTimer2)



Agrega un componente que permite desactivar el evento “**OnTimer2**”.

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada “**EN**” es 1.

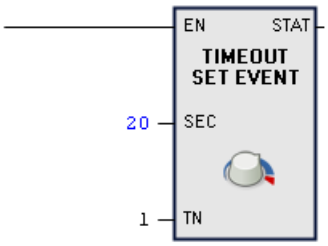
El valor del puerto de salida “**ENO**” es una copia del valor recibido en el puerto “**EN**”.

18 Componentes para Temporizadores Timeout

Un temporizador tipo **Timeout** genera un evento de naturaleza efímera. Es decir señalan que se agotó un tiempo de espera o pasó un tiempo determinado desde que se agotó el temporizador.

Por ejemplo, puede activar un **Timeout** en 120 segundos si la entrada digital DIN1 del PLC está activa. Cuando transcurran los 120 segundos, el evento **OnTimeout** será llamado y usted puede activar un relé dentro del diagrama evento.

18.1 TimeoutSetEvent (Activa un Timeout para el Evento OnTimeout)



Agrega un componente que permite activar un temporizador tipo **Timeout** para el evento "**OnTimeout**".

El evento **OnTimeout** será llamado cuando el **Timeout** definido por este componente expire en X segundos.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

Un temporizador tipo **Timeout** genera un evento de naturaleza efímera. Es decir señalan que se agotó un tiempo de espera o pasó un tiempo determinado desde que se agotó el temporizador.

Por ejemplo, puede activar un **Timeout** en 120 segundos si la entrada digital DIN1 del PLC está activa. Cuando transcurran los 120 segundos, el evento **OnTimeout** será llamado y usted puede activar un relé dentro del diagrama evento.

18.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**SEC**" se especifican la cantidad de segundos para que el timeout sea generado.
- En el puerto "**TN**" se especifica el numero de timeout a activar.

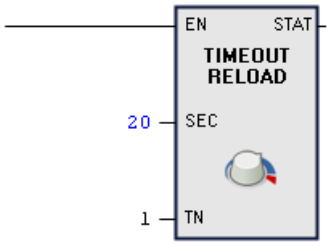
18.1.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el evento pudo configurarse con éxito.

18.1.3 Notas:

- Recuerde crear el diagrama para procesar el evento.
- Una vez creado un **Timeout**, es posible eliminarlo con el componente **TimeoutClrEvent**.
- En el caso de querer modificar el valor del **Timeout** o reactivarlo (si ya expiró), puede llamar nuevamente a este componente o al componente **TimeoutReload**.

18.2 TimeoutReload (Recargar un Temporizador Timeout)



Agrega un componente que permite recargar un temporizador tipo **Timeout** para el evento "**OnTimeout**".

El evento **OnTimeout** será llamado cuando el **Timeout** recargado por este componente expire en X segundos.

Por ejemplo, puede reactivar un **Timeout** desde el diagrama del evento **OnTimeout**.

18.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**SEC**" se especifican la cantidad de segundos para que el timeout sea generado.
- En el puerto "**TN**" se especifica el numero de timeout a recargar.

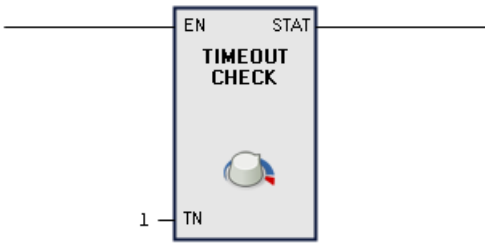
18.2.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el Timeout pudo recargarse con éxito.

18.2.3 Notas:

- Recuerde crear el diagrama para procesar el evento.
- El timeout debe ser previamente activado con el componente **TimeoutSetEvent** al menos una vez, antes de realizar la recarga.
- Una vez creado un **Timeout**, es posible eliminarlo con el componente **TimeoutClrEvent**.
- En el caso de querer modificar el valor del **Timeout** o reactivarlo (si ya expiró), puede llamar nuevamente a este componente o al componente **TimeoutSetEvent** también.
- Dependiendo del modelo del PLC, el número de temporizadores del tipo Timeout puede variar.

18.3 TimeoutCheck (Comprobar Timeout del Temporizador)



Agrega un componente que permite comprobar si un temporizador del tipo **Timeout** ha expirado.

Si múltiples temporizadores **Timeout** están activos, este componente nos permite identificar cual ha expirado, ya sea desde el diagrama evento "**OnTimeout**" o desde cualquier otro diagrama.

18.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**TN**" se especifica el numero de timeout a comprobar.

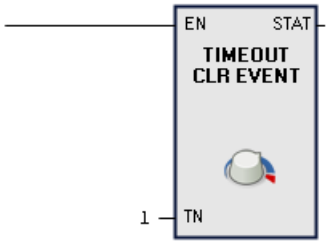
18.3.2 Salidas:

- La salida "**STAT**" del componente es verdadera solo si el timeout ha expirado.

18.3.3 Notas:

- Dependiendo del modelo del PLC, el número de temporizadores del tipo Timeout puede variar.
- Si el componente no se ejecuta (**EN** = 0), la salida **STAT** es 0. Conviene conectar este componente a la barra de alimentación del diagrama evento "**OnTimeout**" para que siempre EN = 1, y el componente siempre sea ejecutado.

18.4 TimeoutClrEvent (Desactivar un Temporizador del Tipo Timeout)



Agrega un componente que permite desactivar un temporizador del tipo **Timeout**.

18.4.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**TN**" se especifica el numero de timeout a desactivar.

18.4.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el timeout pudo configurarse con éxito.

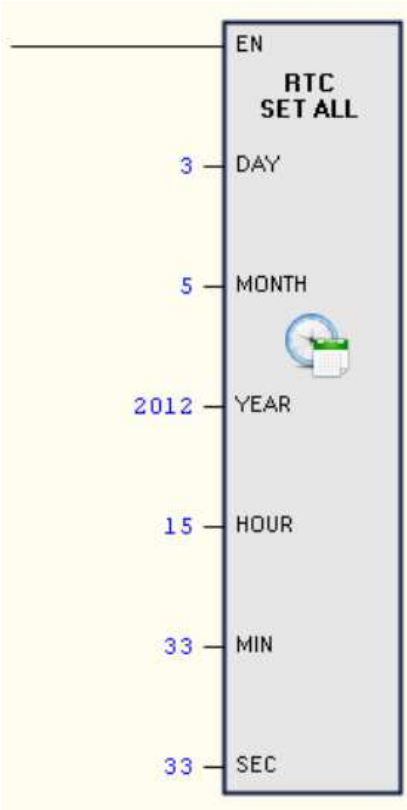
18.4.3 Notas:

- Una vez desactivado un **Timeout**, solo es posible volver a utilizarlo si se llama al componente **TimeoutSetEvent** para activarlo nuevamente.
- Si hay otros **Timeouts** activados y pendientes, el evento "**OnTimeout**" será generado.
- Dependiendo del modelo del PLC, el número de temporizadores del tipo Timeout puede variar.

19 Componentes del RTC (Reloj de Tiempo Real)

Componentes que acceden al RTC (Reloj de Tiempo Real) que contiene hora y fecha actual del sistema. Puede configurar el reloj, obtener información o establecer alarmas. Puede serle útil consultar la página 87 para ver más componentes que emplean alarmas de tiempo.

19.1 RtcSetAll (Establecer Hora y Fecha en RTC)



Agrega un componente que permite establecer hora y fecha en el ***RTC*** (Real Time Clock o Reloj de Tiempo Real) del PLC. Luego de establecer hora y fecha, el reloj mantendrá actualizado los datos conforme pase el tiempo.

Este componente es útil para cambiar fecha y hora, o para restablecer en caso de pérdida.

El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.

19.1.1 Entradas:

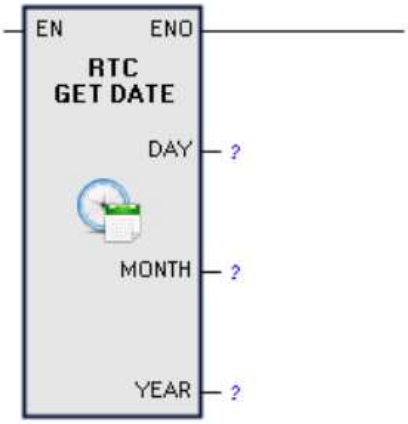
- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**DAY**" permite establecer el día del mes (rango 1 a 31) a partir de una variable o constante tipo **Int32**.
- La entrada "**MONTH**" permite establecer el mes del año (rango 1 a 12) a partir de una variable o constante tipo **Int32**.

- La entrada "**YEAR**" permite establecer el año (rango 0 a 4095) a partir de una variable o constante tipo **Int32**.
- La entrada "**HOUR**" permite establecer hora del día (rango 0 a 23) a partir de una variable o constante tipo **Int32**.
- La entrada "**MIN**" permite establecer minuto (rango 0 a 59) a partir de una variable o constante tipo **Int32**.
- La entrada "**SEC**" permite establecer segundos (rango 0 a 59) a partir de una variable o constante tipo **Int32**.

19.1.2 Notas / Consejos:

- Es posible también establecer fecha y hora en el PLC a través de una PC conectada a la red Ethernet con el entorno ***StxLadder*** de programación. Remítase a la documentación.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.2 RtcGetDate (Obtener Fecha del RTC)



Agrega un componente que permite obtener la fecha desde el ***RTC*** (Real Time Clock o Reloj de Tiempo Real) del PLC.

19.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

19.2.2 Salidas:

El componente retorna la fecha actual en tres variables, especificadas en sus puertos de salida:

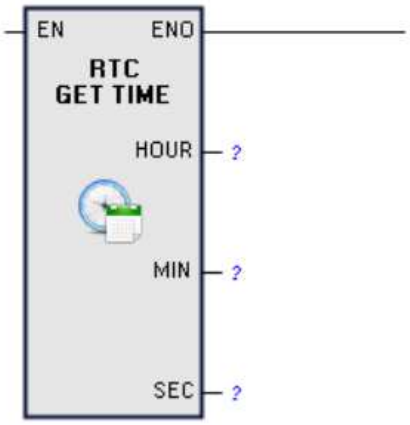
- Salida: "**DAY**" retorna día del mes (rango 1-31). Tipo de variable **Int32**.
- Salida: "**MONTH**" retorna mes del año (rango 1-12). Tipo de variable **Int32**.
- Salida: "**YEAR**" retorna año (rango 0-4095). Tipo de variable **Int32**.

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

19.2.3 Notas:

- La variables de salidas no son alteradas si el componente no se ejecuta (**EN** = 0).
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.3 RtcGetTime (Obtener Hora de RTC)



Agrega un componente que permite obtener la hora desde el ***RTC*** (Real Time Clock o Reloj de Tiempo Real) del PLC.

19.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

19.3.2 Salidas:

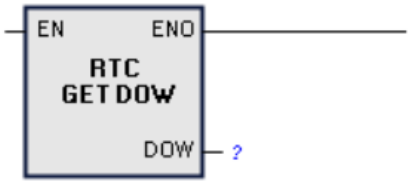
El componente retorna la hora actual en tres variables, especificadas en sus puertos de salida:

- Salida: "**HOUR**" retorna hora del día (rango 0-23). Tipo de variable **Int32**.
- Salida: "**MIN**" retorna minutos (rango 0-59). Tipo de variable **Int32**.
- Salida: "**SEC**" retorna segundos (rango 0-59). Tipo de variable **Int32**.
- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

19.3.3 Notas:

- La variables de salidas no son alteradas si el componente no se ejecuta (**EN** = 0).
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.4 RtcGetDOW (Obtener Día de la Semana del RTC)



Agrega un componente que permite obtener el día de la semana actual desde el ***RTC*** (Real Time Clock o Reloj de Tiempo Real) del PLC.

19.4.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

19.4.2 Salidas:

- El puerto de salida **DOW** (Day Of Week) retorna en una variable del tipo **Int32** el día de la semana actual.

  El rango de valores es el siguiente:

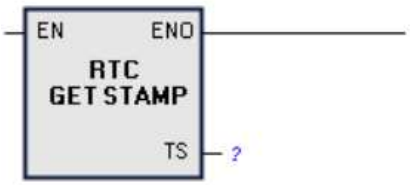
  1 = Domingo
  2 = Lunes
  3 = Martes
  4 = Miércoles
  5 = Jueves
  6 = Viernes
  7 = Sábado

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

19.4.3 Notas:

- La variables de salidas no son alteradas si el componente no se ejecuta (**EN** = 0).
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.5 RtcGetTimestamp (Obtener Timestamp del RTC)



Agrega un componente que permite obtener el "***timestamp***" de la fecha y hora actual desde el ***RTC*** (Real Time Clock o Reloj de Tiempo Real) del PLC.

19.5.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

19.5.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".
- En el puerto de salida "**TS**", se retorna en una variable tipo **Int32** el "***timestamp***" de la fecha y hora actual

19.5.3 Información:

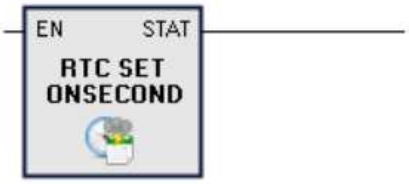
Un "***timestamp***" es un valor entero que representa el numero de segundos transcurridos desde el 01/01/1970 a las 00:00 hs. La ventaja de un "***timestamp***" radica en que nos permite mantener una fecha y hora específica en una sola variable entera. Luego, podemos almacenarla en alguna memoria no volátil o convertirla nuevamente a un formato simple de leer.

Un "***timestamp***" es muy útil para manipular información del tiempo y almacenarlo en registros.

19.5.4 Notas:

- La variables de salidas no son alteradas si el componente no se ejecuta (**EN** = 0).
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.6 RtcOnSecondSetEvent (Activar Evento RtcOnSecond)



Agrega un componente que permite activar el evento "***RtcOnSecond***" del **RTC** (Real Time Clock).

El evento "***RtcOnSecond***" se genera cada un segundo cuando esta activado y se debe al incremento del tiempo en el reloj **RTC**.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

El evento "***RtcOnSecond***" es muy útil conocer cuando se actualiza la hora/fecha en el ***RTC*** o para realizar lógicas que deban procesarse cada 1 segundo.

19.6.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

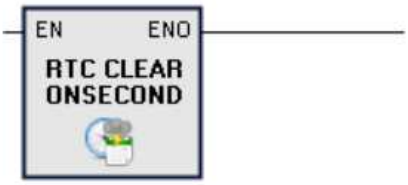
19.6.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el evento pudo configurarse con éxito.

19.6.3 Notas:

- Recuerde crear el diagrama para procesar el evento.
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.7 RtcOnSecondClrEvent (Desactivar Evento RtcOnSecond)

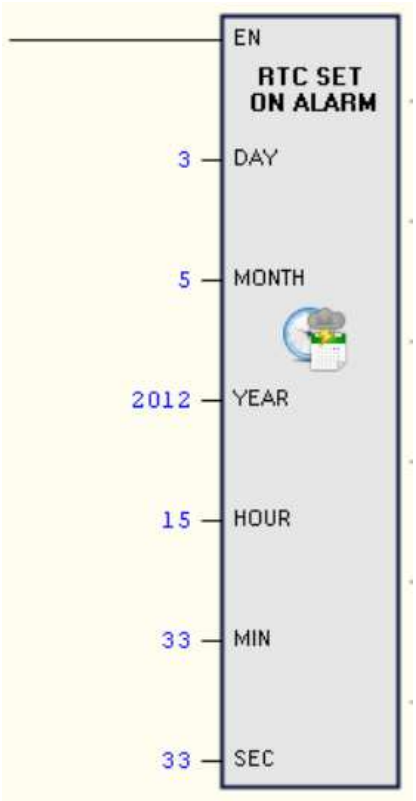


Agrega un componente que permite desactivar el evento "***RtcOnSecond***" del **RTC** (Real Time Clock).

El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

19.8 RtcOnAlarmSetEvent (Activar Evento RtcOnAlarm)



Agrega un componente que permite activar el evento "***RtcOnAlarm***" del **RTC** (Real Time Clock).

El evento "***RtcOnAlarm***" se producirá a una hora y fecha especificada por el componente.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

El evento "***RtcOnAlarm***" es muy útil para realizar alguna tarea a una hora y fecha específica.

La hora y fecha de activación del evento se especifican en los puertos de entrada del componente.

19.8.1 Entradas:

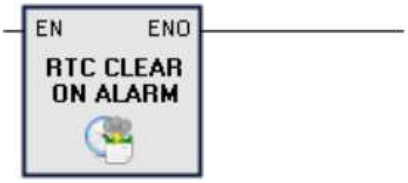
- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**DAY**" permite establecer el día del mes (rango 1 a 31) a partir de una variable o constante tipo **Int32**.
- La entrada "**MONTH**" permite establecer el mes del año (rango 1 a 12) a partir de una variable o constante tipo **Int32**.
- La entrada "**YEAR**" permite establecer el año (rango 0 a 4095) a partir de una variable o constante tipo **Int32**.

- La entrada "**HOUR**" permite establecer hora del día (rango 0 a 23) a partir de una variable o constante tipo **Int32**.
- La entrada "**MIN**" permite establecer minuto (rango 0 a 59) a partir de una variable o constante tipo **Int32**.
- La entrada "**SEC**" permite establecer segundos (rango 0 a 59) a partir de una variable o constante tipo **Int32**.

19.8.2 Notas:

- Recuerde crear el diagrama para procesar el evento.
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

19.9 RtcOnAlarmClrEvent (Desactivar Evento RtcOnAlarm)



Agrega un componente que permite desactivar el evento "***RtcOnAlarm***" del **RTC** (Real Time Clock).

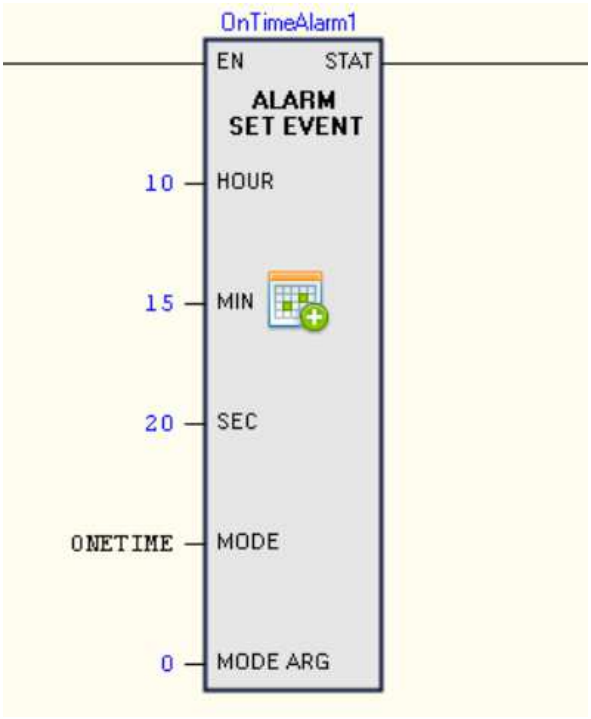
El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

20 Componentes para Alarmas de Tiempo

Las alarmas de tiempo permiten configurar eventos que se dispararán a una hora determinada, según un modo de repetición (todos los días, de lunes a viernes, un día de mes específico, un día de la semana específico, una sola vez, etc.).

20.1 TimeAlarmSetEvent (Activar un Evento de Alarma OnTimeAlarmX)



Agrega un componente que permite activar un evento de alarma de tiempo tipo "***OnTImeAlarmX".***

El evento se genera según la configuración del componente de forma repetitiva o única.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

Según el modelo del PLC, pueden configurarse varias alarmas al mismo tiempo (por ejemplo hasta 5). Cada número de alarma se selecciona desde la configuración del componente. Visualmente, el número de alarma se indica en la parte superior del componente dibujado en el diagrama.

Una alarma se configura con un horario y un modo de funcionamiento.

20.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

Las entradas del componente que configuran la hora de la alarma son:

- La entrada "**HOUR**" permite establecer hora del día (rango 0 a 23) a partir de una variable o constante tipo **Int32**.
- La entrada "**MIN**" permite establecer minuto (rango 0 a 59) a partir de una variable o constante tipo **Int32**.
- La entrada "**SEC**" permite establecer segundos (rango 0 a 59) a partir de una variable o constante tipo **Int32**.

Las entradas del componente que configuran el **modo de la alarma** son:

- Entrada "**MODE**", especifica el **modo de la alarma** (no repetitiva, repetitiva, lunes a viernes, etc). Los **modos de alarma** admitidos se listan más adelante.
- Entrada "**MODE ARG**", es opcional y permite agregar un argumento al **modo de alarma** a través de una variable o constante del tipo **Int32**.

20.1.2 Salidas:

- La salida "**STAT**" del componente es "**1**" (verdadera) si el evento pudo configurarse con éxito. En caso de ser falso, el evento no pudo ser creado debido a errores de parámetros u otros errores internos.

20.1.3 Modos de alarma admitidos:

Los modos de alarma admitidos para la entrada **MODE** se seleccionan desde propiedades del componente y puede ser alguno de los siguientes:

- **ONETIME**: Ejecutar la alarma una sola vez. No repetir.
- **REPEAT**: Repetir la alarma todos los días.
- **REPEAT_DOW**: Repetir la alarma todos los días de semana (Day Of Week) especificado en la entrada **MODE_ARG**. Rango 1 a 7, donde: Domingo = 1, Lunes = 2, …,Sábado = 7.
- **REPEAT_DOM**: Repetir la alarma todos los días X de cada mes (Day Of Month) especificado en la entrada **MODE_ARG**. Rango 1 a 31.
- **REPEAT_MF**: Repetir la alarma de Lunes a Viernes.
- **REPEAT_MS**: Repetir la alarma de Lunes a Sábados.
- **REPEAT_SS**: Repetir la alarma de Sábado a Domingo.

20.1.4 Ejemplos

- Para ejecutar una alarma todos los días a las 15:33 hs, especificamos: **HOUR**=15, **MIN**=33, **SEC**=0, **MODE** = REPEAT, **MODE_ARG** = 0.
- Para ejecutar una alarma todos lunes a las 7:00 hs, especificamos: **HOUR**=7, **MIN**=0, **SEC**=0, **MODE** = REPEAT_DOW, **MODE_ARG** = 2.
- Para ejecutar una alarma de lunes viernes a las 15:30 hs, especificamos: **HOUR**=15, **MIN**=30, **SEC**=0, **MODE** = REPEAT_MF, **MODE_ARG** = 0.

Observe como la entrada **MODE_ARG** es cero si no es utilizada por el modo de alarma.

20.1.5 Notas

- Recuerde crear el diagrama para procesar el evento.
- Las alarmas utilizan el ***RTC*** (Real Time Clock) del PLC para obtener horario y fechas.
- El ***RTC*** es un componente interno del PLC que brinda fecha y hora actualizada. Dicha información se mantiene en todo momento, a menos que la batería del RTC este agotada y se produzca un corte energético.
- Controle periódicamente la batería del RTC para evitar perdida de fecha y hora luego de una interrupción eléctrica. El PLC tiene un indicador especial que avisa cuando los datos del RTC son incoherentes.

20.2 TimeAlarmClrEvent (Desactivar un Evento de Alarma OnTimeAlarmX)

Agrega un componente que permite desactivar el evento "***OnTimeAlarmX***" de una alarma de tiempo.

El número de alarma a desactivar se especifica en las propiedades del componente.

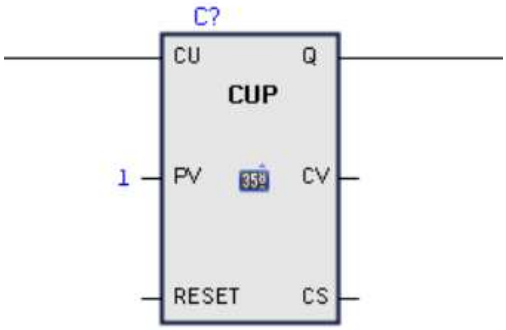
Si la entrada "**EN**" del componente es verdadera, se procede a ejecutar el componente por el PLC.

La salida "**STAT**" del componente es verdadera si el evento pudo ser desactivado. De lo contrario, se produjo un error (alarma inválida, etc).

21 Componentes para Contadores Ladder

Los contadores Ladder le permiten contar eventos de acuerdo al flujo de corriente que llega a su entrada de cuenta en un diagrama Ladder. Son de propósito general y estándar en la mayoría de los PLC del mercado.

21.1 CounterUp (Contador Ascendente CUP)



Agrega un componente que actúa como un contador del tipo ascendente "**CUP**".

21.1.1 Funcionamiento:

El contador **CUP** comienza a contar de forma ascendente a partir de "**0**" con cada transición positiva en la entrada **CU**. Cuando el valor actual de la cuenta **CV** (Count Value) es igual o mayor al valor final **PV** (Preset Value), la salida **Q** se pone a "**1**" y el contador detiene su cuenta.

El contador es reiniciado al valor "**0**" cuando la entrada **RESET** se pone a "**1**".

21.1.2 Entradas:

- La entrada **CV** activa el contador cuando hay una transición de **0** a **1** en el flujo de corriente.
- La entrada **PV** (Preset Value) permite establecer el valor final del contador a través una variable o constante del tipo **Int32**.
- La entrada **RESET** admite una variable del tipo **Bool**. Si es "**1**" reinicia el contador y provoca que la salida **Q** y **CS** sea **0**. Su uso es opcional.

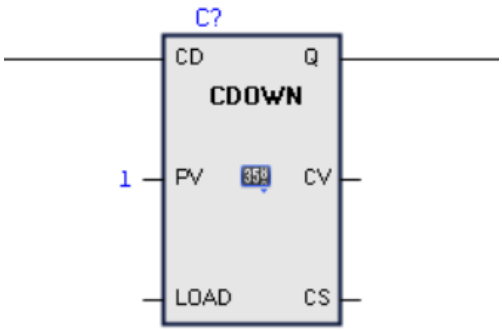
21.1.3 Salidas:

- La salida **Q** es del tipo **Bool** y su valor depende del funcionamiento del contador.
- La salida **CV** (Count Value) deposita en una variable del tipo **Int32** la cuenta actual del contador. Su uso es opcional.
- La salida **CS** (Counter Status) permite copiar el valor de la salida **Q** en una variable del tipo **Bool**. Su uso es opcional.

21.1.4 Notas:

- El contador comienza su cuenta en el valor **0**.
- Cuando el contador llega al final de la cuenta y **Q** es "**1**", no se incrementará mas la cuenta interna hasta que se aplique un "**1**" en la entrada **RESET**.
- El número de contador se especifica en las propiedades del componente.
- La cantidad de contadores depende del modelo de PLC seleccionado.

21.2 CounterDown (Contador Descendente CDOWN)



Agrega un componente que actúa como un contador del tipo descendente "**CDOWN**".

21.2.1 Funcionamiento:

El contador **CDOWN** comienza a contar en forma descendente desde el valor **PV** (Preset Value) con cada transición positiva en la entrada **CD**. Cuando el valor actual de la cuenta **CV** (Count Value) es igual a "**0**", la salida **Q** se pone a "**1**" y el contador detiene su cuenta.

El contador se carga al valor en **PV** (Preset Value) cuando la entrada **LOAD** se pone a "**1**".

21.2.2 Entradas:

- La entrada **CD** activa el contador cuando hay una transición de **0** a **1** en el flujo de corriente.
- La entrada **PV** (Preset Value) permite establecer el inicial del contador a través una variable o constante del tipo **Int32**.
- La entrada **LOAD** admite una variable del tipo **Bool**. Si es "**1**" carga al contador con el valor almacenado en **PV** y provoca que la salida **Q** y **CS** sea **0**. Su uso es opcional.

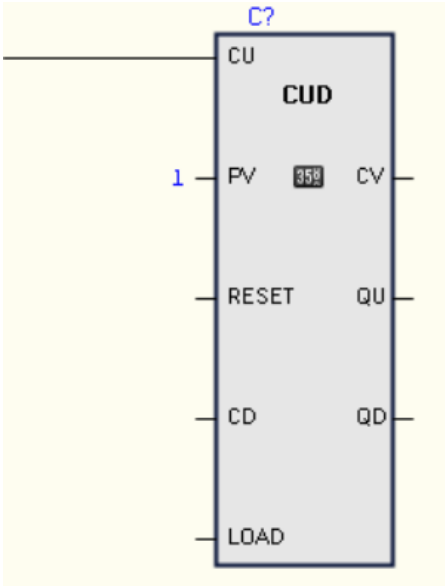
21.2.3 Salidas:

- La salida **Q** es del tipo **Bool** y su valor depende del funcionamiento del contador.
- La salida **CV** (Count Value) deposita en una variable del tipo **Int32** la cuenta actual del contador. Su uso es opcional.
- La salida **CS** (Counter Status) permite copiar el valor de la salida **Q** en una variable del tipo **Bool**. Su uso es opcional.

21.2.4 Notas:

- El contador comienza su cuenta en el valor almacenado en **PV**.
- Cuando el contador llega al final de la cuenta y **Q** es "**1**", no contará mas hasta que se aplique un "**1**" en la entrada **LOAD**.
- El número de contador se especifica en las propiedades del componente.
- La cantidad de contadores depende del modelo de PLC seleccionado.

21.3 CounterUpDown (Contador Ascendente/Descendente CUD)



Agrega un componente que actúa como un contador del tipo ascendente/descendente "**CUD**". Permite contar hacia arriba y hacia abajo.

21.3.1 Funcionamiento:

El contador **CUD** cuenta en forma ascendente con cada transición positiva en la entrada **CU** y cuenta en forma descendente con cada transición positiva en la entrada **CD**.

Cuando el valor de cuenta actual **CV** (Count Value) es igual a cero la salida **QD** se pone a "**1**".

Cuando el valor de cuenta actual **CV** es igual o mayor a **PV** (Preset Value) la salida **QU** se pone a "**1**".

Si la entrada **RESET** es "**1**", la cuenta del contador se reinicia a cero y la salida **QD** se pone en "**1**", mientras que la salida **QU** se pone en "**0**".

Si la entrada **LOAD** es "**1**", la cuenta del contador se carga con el valor del puerto **PV** y la salida **QU** se pone en "**1**", mientras que la salida **QD** se pone en "**0**".

La entrada **RESET** tiene prioridad sobre la entrada **LOAD**. Es decir, que si ambas tienen el valor "**1**", se ejecutara la operación **RESET** únicamente.

21.3.2 Entradas:

- La entrada **CU** incrementa la cuenta cuando hay una transición de **0** a **1** en el flujo de corriente.
- La entrada **PV** (Preset Value) permite establecer el final de cuenta ascendente a través una variable o constante del tipo **Int32**.
- La entrada **RESET** admite una variable del tipo **Bool**. Si es "**1**" reinicia el contador y provoca que la salida **QD** sea **1** y **QU** sea **0**. Su uso es opcional.
- La entrada **CD** admite una variable del tipo **Bool** y decrementa la cuenta cuando hay una transición de **0** a **1** en su valor. Su uso es opcional.

- La entrada **LOAD** admite una variable del tipo **Bool**. Si es "**1**" carga al contador con el valor almacenado en **PV** y provoca que la salida **QU** sea **1** y **QD** sea **0**. Su uso es opcional.

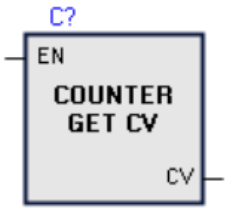
21.3.3 Salidas:

- La salida **CV** (Count Value) deposita en una variable del tipo **Int32** la cuenta actual del contador. Su uso es opcional.
- La salida **QU** es del tipo **Bool** y su valor es "**1**" cuando el contador alcanza el valor **PV**.
- La salida **QD** es del tipo **Bool** y su valor es "**1**" cuando el contador llega al valor cero.

21.3.4 Notas:

- El contador comienza su cuenta inicialmente en el valor cero. Si necesita que comience en el valor **PV**, debe aplicar un "**1**" a la entrada **LOAD** al menos una vez para cargar el contador.
- Cuando el contador llega al valor **PV**, ignora la entrada **CU**.
- Cuando el contador llega al valor cero, ignora la entrada **CD**.
- El número de contador se especifica en las propiedades del componente.
- La cantidad de contadores depende del modelo de PLC seleccionado.

21.4 CounterGetCV (Obtener Count Value del Contador)



Agrega un componente que obtiene el valor de cuenta del contador seleccionado. La cuenta actual del contador se conoce como *Count Value* o **CV**.

Este componente **accede directamente** a la memoria del contador, por lo tanto obtiene el valor **inmediato** del contador en el punto de ejecución del diagrama. Se utiliza en casos donde es necesario obtener un valor actualizado de **CV** sin esperar un nuevo SCAN CYCLE del PLC.

21.4.1 Entradas:

- Si el flujo de corriente en la entrada **EN** (Enable) es "**1**", se actualiza la salida **CV** con el valor actual del contador. Si la entrada **EN** es "**0**", la salida **CV** no es actualizada (si hay una variable conectada, mantiene su valor previo al llamar al componente).

21.4.2 Salidas:

- La salida **CV** (Counter Value) deposita en una variable del tipo **Int32** el valor actual de cuenta del contador.

21.4.3 Notas:

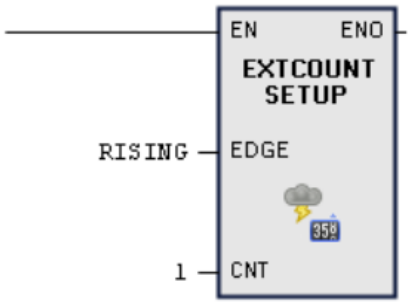
- El número de contador se especifica en las propiedades del componente.
- La cantidad de contadores depende del modelo de PLC seleccionado.

22 Componentes para Contadores Externos

Los contadores externos le permiten contar eventos externos al PLC que por lo general se presentan en alguna entrada externa del mismo. Dichos contadores pueden contar pulsos, determinar frecuencia, medir periodo entre pulsos y hasta calcular RPM (Revoluciones Por Minuto).

A diferencia de los contadores Ladder, los contadores externos cuentan en asincrónicamente, es decir únicamente cuando un evento externo se presenta, haciéndolos rapidos para tareas de tiempo real.

22.1 ExtCountSetup (Configurar Contador Externo)



Agrega un componente que permite configurar y activar las entradas externas para contadores del PLC.

Con los contadores externos puede contar eventos (pulsos), medir frecuencia de una onda cuadrada, medir tiempo entre dos pulsos y calcular RPM (revoluciones por minutos).

22.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**EDGE**" del componente debe especificarse el tipo de flanco del pulso a medir (por ejemplo flanco ascendente, descendente o ambos). El tipo de flanco determina cuando un pulso es considerando como un evento para contar o medir.
- El puerto "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.1.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

22.1.3 Importante:

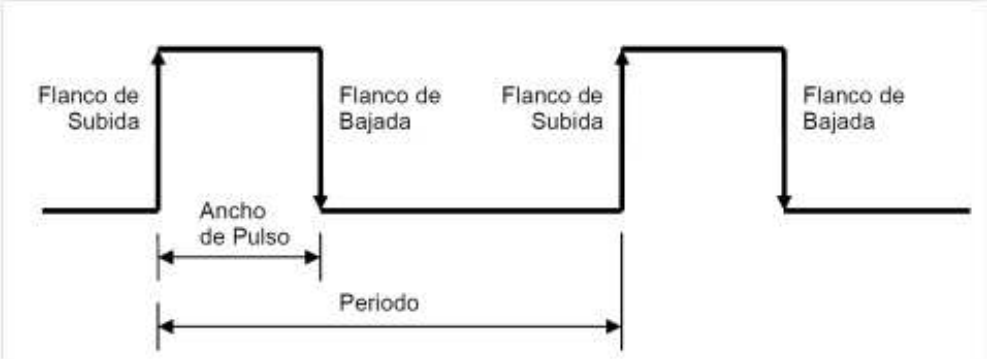
- Se recomienda utilizar este componente para configurar los contadores externos antes de utilizar cualquier otro componente asociado a los contadores externos. Generalmente se llama a este componente desde el diagrama "Inicio" para configurar los contadores externos al iniciar el PLC.

22.1.4 Tipos de flancos admitidos:

La entrada "**EDGE**" admite los siguientes flancos que determinan cuando un pulso es considerando como un evento para contar o medir:

- **RISING**: Flanco ascendente/subida. Transición de un nivel bajo de tensión a nivel alto.
- **FALLING**: Flanco descendente/Bajada. Transición de un nivel alto de tensión a nivel bajo.
- **BOTH**: Ambos flancos. Se genera una cuenta o evento de medición al ocurrir un flanco ascendente o descendente.

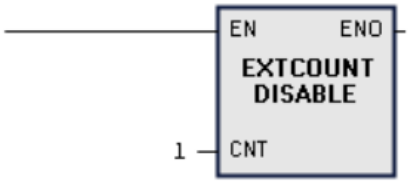
La siguiente figura muestra los flancos de una señal cuadrada a la entrada de un contador externo.



22.1.5 Notas:

- No confunda los **contadores externos** (este componente), con los **contadores básicos** del PLC. Los contadores externos responden a eventos externos del PLC (como ser un cambio de estado en una entrada discreta). Los **contadores básicos** del PLC cuentan operaciones de software realizadas por el programador.
- El número de contadores externos depende del modelo de PLC.

22.2 ExtCountDisable (Desactivar Contador Externo)



Agrega un componente que permite desactivar un contador externo del PLC.

Detiene todas las operaciones en progreso. Luego de ejecutarse el componente, el contador no contara más eventos.

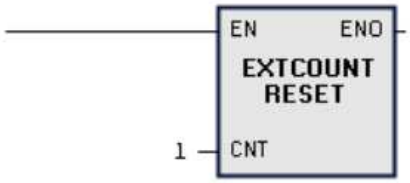
22.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el contador externo a desactivar.

22.2.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

22.3 ExtCountReset (Resetear Contador Externo)



Agrega un componente que permite resetear o limpiar todos los registros internos de un contador externo del PLC.

Valores "***match***" y registros de eventos son reseteados o puestos a su valor de inicio.

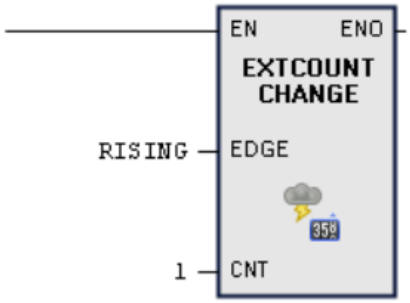
22.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el contador externo a limpiar.

22.3.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

22.4 ExtCountChangeEdge (Cambiar Flanco de Cuenta del Contador Externo)



Agrega un componente que permite cambiar el flanco de cuenta de un contador externo del PLC.

22.4.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**EDGE**" del componente debe especificarse el tipo de flanco del pulso a medir (por ejemplo flanco ascendente, descendente o ambos). El tipo de flanco determina cuando un pulso es considerando como un evento para contar o medir.
- La entrada "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.4.2 Salidas:

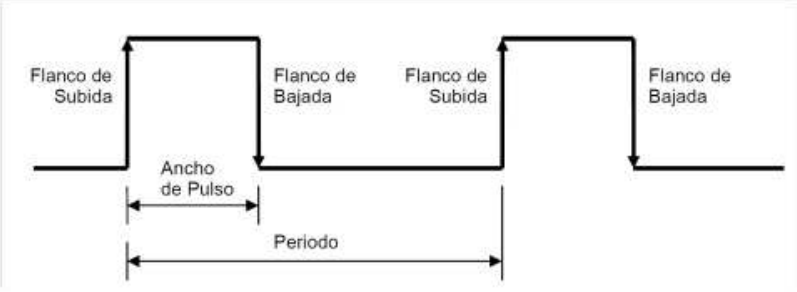
- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

22.4.3 Tipos de flancos admitidos:

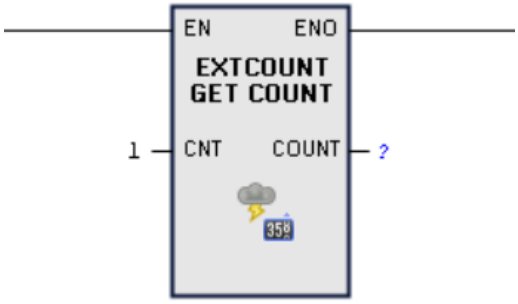
La entrada "**EDGE**" admite los siguientes flancos que determinan cuando un pulso es considerando como un evento para contar o medir:

- **RISING**: Flanco ascendente/subida. Transición de un nivel bajo de tensión a nivel alto.
- **FALLING**: Flanco descendente/Bajada. Transición de un nivel alto de tensión a nivel bajo.
- **BOTH**: Ambos flancos. Se genera una cuenta o evento de medición al ocurrir un flanco ascendente o descendente.

La siguiente figura muestra los flancos de una señal cuadrada a la entrada de un contador externo.



22.5 ExtCountGetEventCnt (Obtener Cuenta del Contador Externo)



Agrega un componente que permite obtener el valor del contador de eventos de un **contador externo** del PLC.

Este componente accede a un registro interno del **contador externo**.

22.5.1 Entradas:

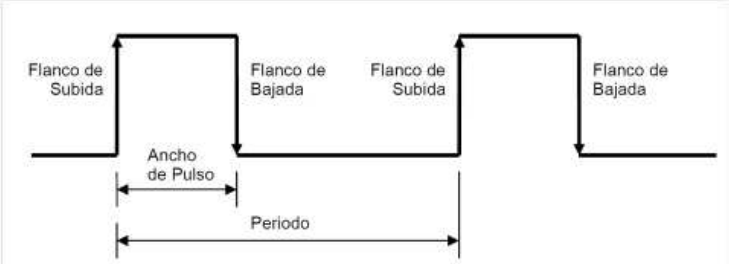
- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.5.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".
- La salida "**COUNT**" retorna el número de eventos externos o pulsos contados. El valor es depositado en una variable del tipo **Int32**.

22.5.3 Información:

La siguiente figura muestra los flancos de una señal cuadrada a la entrada de un contador externo.

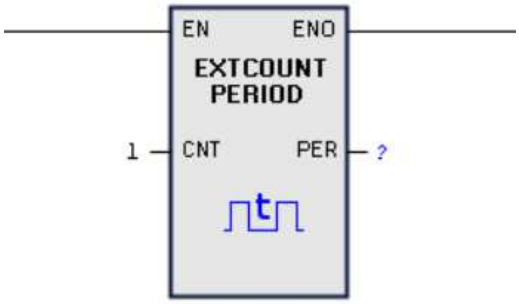


El contador puede utilizar cualquier tipo de flanco o ambos.

22.5.4 Notas:

- Recuerde que el contador retorna el número de eventos externos o pulsos contados. La cuenta puede incluir eventos por flanco de subida, bajada o ambos. Los tipos de flancos de cuenta pueden ser definidos con los componentes: **ExtCountSetup** y **ExtCountChangeEdge**.
- Es posible reiniciar la cuenta del contador con el componente **ExtCountReset**.

22.6 ExtCountGetEventPeriod (Obtener Periodo Entre Eventos)



Agrega un componente que permite obtener el tiempo entre dos eventos externos contados por el **contador externo** del PLC.

Este componente accede a un registro interno del **contador externo**.

22.6.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.6.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".
- La salida "**PER**" retorna el tiempo en micro-segundos entre dos eventos externos sucesivos contados por el contador (por ejemplo entre dos pulsos). El valor es almacenado en una variable del tipo **Int32**.

  El valor retornado en "**PER**" es mayor a "**0**" si la medición es correcta. Si es "**0**" puede deberse a:

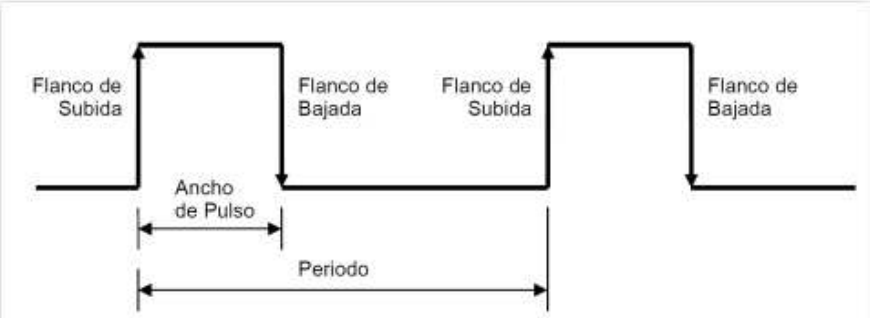
  1. Tiempo entre eventos es menor a 1 uS.
  2. No se han contados eventos todavía.

22.6.3 Notas:

- Si a la entrada del contador hay una señal periódica y la cuenta es por flanco descendente o ascendente, esta función retorna el periodo de la señal en uS (micro-segundos).
- Recuerde que el contador mide tiempo entre eventos externos o pulsos. Los tipos de flancos de medición pueden ser definidos con los componentes: **ExtCountSetup** y **ExtCountChangeEdge**.
- Es posible reiniciar los registros internos del contador con el componente **ExtCountReset**.

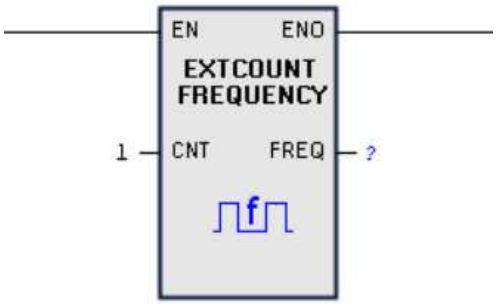
22.6.4 Información:

La siguiente figura muestra los flancos de una señal cuadrada a la entrada de un contador externo.



El contador puede utilizar cualquier tipo de flanco o ambos. Observar cómo es posible medir ancho de pulso o periodo entre los flancos.

22.7 ExtCountGetEventFreq (Obtener Frecuencia Entre Eventos)



Agrega un componente que permite obtener la frecuencia entre dos eventos externos contados por el **contador externo** del PLC.

Este componente accede a un registro interno del **contador externo**.

22.7.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.7.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".
- La salida "**FREQ**" retorna la frecuencia (en Hertz) entre dos eventos externos sucesivos contados por el contador (por ejemplo entre dos pulsos). El valor es almacenado en una variable del tipo **Int32**.

  El valor retornado en "**FREQ**" es mayor a "**0**" si la medición es correcta. Si es "**0**" puede deberse a:

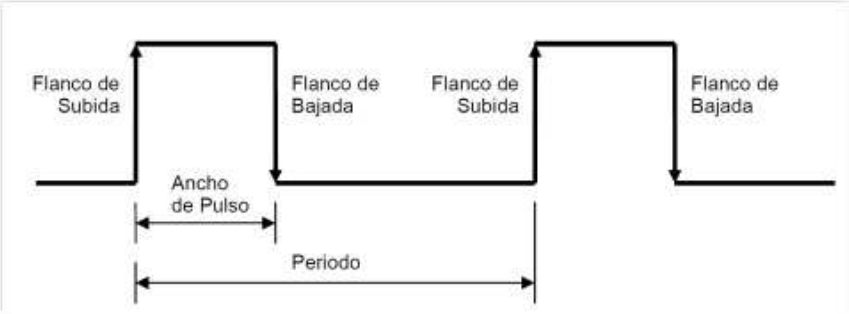
  1. Tiempo entre eventos es menor a 1 uS.
  2. No se han contados eventos todavía.

22.7.3 Notas:

- Si a la entrada del contador hay una señal periódica y la cuenta es por flanco descendente o ascendente, esta función retorna la frecuencia de la señal en Hertz.
- Recuerde que el contador mide frecuencia entre eventos externos o pulsos. Los tipos de flancos de medición pueden ser definidos con los componentes: **ExtCountSetup** y **ExtCountChangeEdge**.
- Es posible reiniciar los registros internos del contador con el componente **ExtCountReset**.

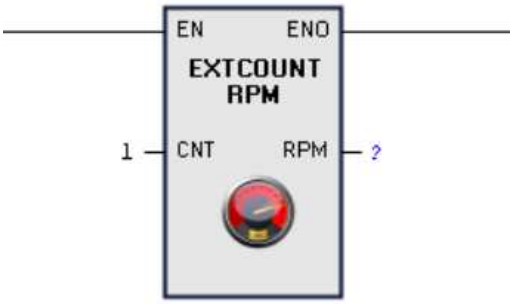
22.7.4 Información:

La siguiente figura muestra los flancos de una señal cuadrada a la entrada de un contador externo.



El contador puede utilizar cualquier tipo de flanco o ambos. Observar cómo es posible medir frecuencia de la señal a partir del periodo.

22.8 ExtCountGetEventRPM (Obtener RPM Entre Eventos)



Agrega un componente que permite calcular los RPM (Revoluciones por Minutos) de una señal periódica presente a la entrada del **contador externo** del PLC.

Este componente accede a un registro interno del **contador externo**.

22.8.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.8.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".
- La salida "**RPM**" retorna los RPM calculados en una variable del tipo **Int32**.

  El valor retornado en "**RPM**" es mayor a "**0**" si la medición es correcta. Si es "**0**" puede deberse a:

  1. Tiempo entre eventos es menor a 1 uS.
  2. No se han contados eventos todavía.

22.8.3 Importante:

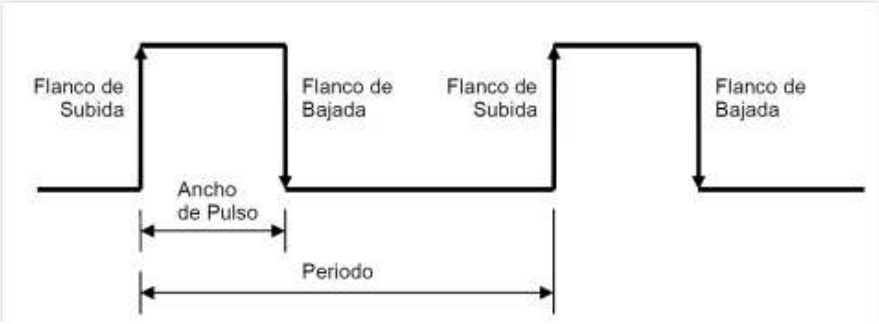
- En la entrada del contador **debe** haber una señal periódica y la cuenta **debe** ser por flanco descendente o ascendente (asegúrese de configurar el contador externo correctamente).

22.8.4 Notas:

- Recuerde que el contador realiza la medición entre eventos externos o pulsos. Los tipos de flancos de medición pueden ser definidos con los componentes: **ExtCountSetup** y **ExtCountChangeEdge**.
- Es posible reiniciar los registros internos del contador con el componente **ExtCountReset**.

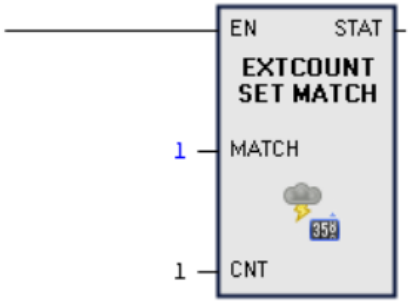
22.8.5 Información:

La siguiente figura muestra los flancos de una señal cuadrada a la entrada de un contador externo.



El contador puede utilizar cualquier tipo de flanco o ambos. Observar cómo es posible medir RPM de la señal a partir del periodo y un factor de conversión.

22.9 ExtCountSetMatchEvent (Activa Evento OnCountXMatch)



Agrega un componente que permite configurar un **contador externo** del PLC para generar un evento ladder del tipo **MATCH**.

El evento **MATCH** se produce cuando el contador llega a una determinada cuenta.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

Un evento **MATCH** es muy útil para ejecutar una lógica exactamente cuando el contador alcance un determinado valor. El contador genera el evento asincrónicamente.

El nombre del evento **MATCH** depende del número de contador externo utilizado. Si por ejemplo utilizamos el contador externo 1, el evento se llamará **OnCount1Match**. Para el contador externo 2, el evento **MATCH** se llamará **OnCount2Match**.

22.9.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**MATCH**" del componente debe especificarse el valor de cuenta para el cual será generado el evento match. El puerto lee el valor de una constante o variable **Int32**.
- La entrada "**CNT**" especifica el contador a utilizar y su valor depende del modelo de PLC. El número de contador determina también la entrada externa del PLC a utilizar. Consulte documentación.

22.9.2 Salidas:

- La salida "**STAT**" del componente es "**1**" (verdadera) si el evento pudo configurarse con éxito.

22.9.3 Notas:

- Recuerde crear el diagrama para procesar el evento **MATCH**.
- Luego del evento match, la cuenta del contador es reseteada a cero.
- Recuerde que el contador retorna el número de eventos externos o pulsos contados. La cuenta puede incluir eventos por flanco de subida, bajada o ambos. Los tipos de flancos de cuenta pueden ser definidos con los componentes: **ExtCountSetup** y **ExtCountChangeEdge**.

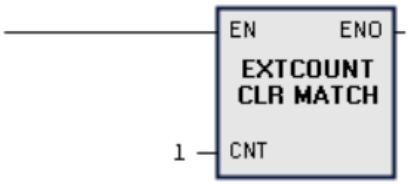
- Es posible reiniciar la cuenta del contador con el componente **ExtCountReset** de forma inmediata.

- No confundir evento de cuenta del contador que permite incrementar su número, con un evento ladder que llama a un diagrama para ejecutar una lógica.

22.9.4 Ejemplo:

- Si la entrada **MATCH** del componente es 1000 y se utiliza el contador externo **1** configurado para contar eventos por flanco ascendente, entonces, se producirá un evento **OnCount1Match**, cuando ocurran **1000** flancos ascendentes a la entrada del terminal del contador externo. El PLC ejecutará la lógica contenida por el diagrama asociado al evento **OnCount1Match**, permitiéndole actuar en el momento preciso de la cuenta.

22.10 ExtCountClrMatchEvent (Desactivar Evento OnCountXMatch)



Agrega un componente que permite desactivar los eventos **MATCH** generados por un contador externo del PLC.

22.10.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**CNT**" especifica el número de contador asociado al evento a desactivar y su valor depende del modelo de PLC.

22.10.2 Salidas:

- El valor del puerto de salida "**ENO**" es una copia del valor recibido en el puerto "**EN**".

22.10.3 Notas:

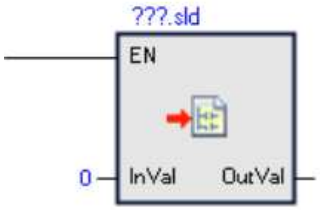
- El nombre del evento **MATCH** depende del número de contador externo utilizado. Si por ejemplo utilizamos el contador externo 1, el evento se llamará **OnCount1Match**. Para el contador externo 2, el evento **MATCH** se llamará **OnCount2Match**.

23 Componentes para Funciones Ladder

Las funciones Ladder le permiten estructurar y organizar mejor la lógica de su proyecto, creando diagramas Ladder independientes que pueden ser llamados desde otros diagramas Ladder según se cumpla o no alguna condición.

Al utilizar funciones Ladder, puede reutilizar código, ahorrando memoria en el PLC. Ademas, como las funciones se ejecutan solo cuando son llamadas de acuerdo a alguna condición, el procesador del PLC no tiene permanentemente que ejecutarlas, por lo tanto mejora el desempeño de velocidad del PLC.

23.1 FuncCall (Llamar a Función Ladder)



Agrega un componente que permite llamar a una función implementada con un diagrama Ladder.

Es posible agregar una función Pawn desde el **Explorador de Proyecto** del entorno **StxLadder**. Para crear una función, diríjase a la carpeta "**Diagramas**" y sobre la carpeta "**Funciones**" haga click derecho del mouse. Luego seleccione "**Agregar nuevo diagrama …**". Una ventana le preguntará el nombre de la función a crear.

El nombre de la función creada, es el nombre de la función Ladder.

Desde las propiedades del componente debe seleccionar la función Ladder a llamar.

El nombre del diagrama seleccionado se mostrará luego en la parte superior del componente para fácil identificación.

23.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**InVal**" lee una variable o constante del tipo **Int32**, y pasa su valor como argumento de entrada al diagrama definido como función Ladder. Su uso es opcional.

23.1.2 Salidas:

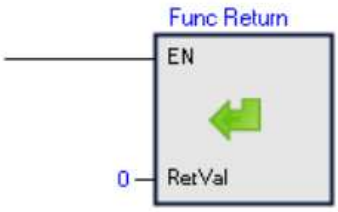
- En el puerto de salida "**OutVal**" se retorna un valor a una variable del tipo **Int32**. El valor retornado es un valor devuelto por la función Ladder. Su uso es opcional.

23.1.3 Notas:

- Cuando se pasa un valor en el puerto **InVal** del componente, el mismo es accesible desde una variable local con el mimo nombre (**InVal**) en el diagrama de la función Ladder.
- La función ladder puede retornar un valor con el componente **FuncReturn**. El valor retornado será copiado a la variable definida en el puerto **OutVal** del componente.

- Si necesita pasar otros valores, puede utilizar variables globales.
- Note que si modifica el valor de la variable local **InVal** dentro del diagrama con la función Ladder, no afectará a la variable especificada en el puerto del componente.

23.2 FuncReturn (Retornar de Funcion Ladder)



Agrega un componente que permite retornar inmediatamente de una función Ladder implementada con un diagrama.

23.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**RetVal**" lee una variable o constante del tipo **Int32**. El valor leído, será devuelto por la función Ladder al componente que llamo a la función (por ejemplo al componente **FuncCall**).

23.2.2 Importante:

- Este componente solo debe llamarse desde una función Ladder.

23.2.3 Información:

Es posible agregar una función Ladder desde el **Explorador de Proyecto** del entorno **StxLadder**. Para crear una función, diríjase a la carpeta "**Diagramas**" y sobre la carpeta "**Funciones**" haga click derecho del mouse. Luego seleccione "**Agregar nuevo diagrama …**". Una ventana le preguntará el nombre de la función a crear.

El nombre del diagrama creado, es el nombre de la función Ladder.

23.2.4 Notas:

- Al ejecutarse este componente, la ejecución del diagrama de la función Ladder detiene su ejecución inmediatamente y retorna. El PLC continuará ejecutando el diagrama que llamo a la función.

24 Componentes Pawn

Los componentes Pawn le permiten insertar código en lenguaje Pawn o llamar funciones Pawn dentro de un diagrama Ladder. Esto le permite obtener gran flexibilidad del entorno, ya que puede implementar algoritmos complejos que no son posibles de realizar con la programación grafica Ladder.

Además, los PLC de Slicetex Electronics tienen gran cantidad de funciones nativas Pawn que usted puede utilizar y sacar mejor provecho al PLC.

24.1 PawnInsert (Insertar Código Pawn)

Agrega un componente que permite insertar código Pawn directamente en el punto donde se ubica el componente en el diagrama.

El código se inserta en el flujo del programa, sin llamar a ninguna función intermedia, por lo que es optimo para operaciones donde se requiere gran velocidad.

Se debe prestar especial atención en el código introducido y se recomienda leer la documentación de referencia.

24.1.1 Información:

El código Pawn es sumamente útil para modificar variables con factores, procesarlas o llamar a funciones nativas del PLC que no están disponible como componentes Ladder.

Puede implementar pequeños algoritmos que en lenguaje Ladder serian muy complicados de lograr.

24.1.2 Ejemplo:

El siguiente código le permite modificar el valor de una variable entera Ladder de forma muy simple, si el flujo de corriente es “1” o verdadero:

```
if(CurrentFlow == 1)
{
   gi_Temp = (gi_Temp * 25 + 87) / 5
}
```

El código anterior comprueba con la instrucción “**if**” si la variable “**CurrentFlow**” (que contiene el valor del flujo de corriente al ejecutarse el componente) tiene el valor “**1**”. Si es el flujo de corriente es “1”, multiplica la variable ladder global del tipo **Int32** llamada **Temp** por **25**, luego le suma **87**, luego divide por **5** y finalmente deposita el resultado en la misma variable.

Notar que el nombre de la variable en código Pawn es distinto al nombre utilizado en el diagrama Ladder. El prefijo “gi” se utiliza para una variable **Global** y del tipo **Int32**. El nombre Pawn de la variable Ladder puede obtenerse desde las propiedades de la variable en la Tabla de variables del proyecto.

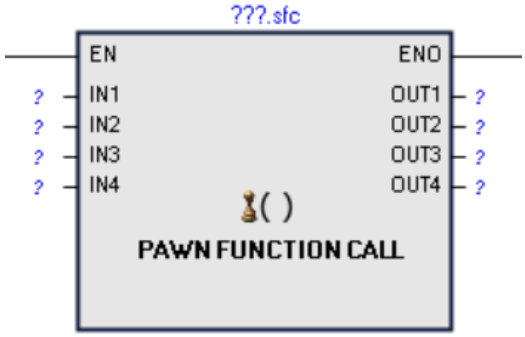
Finalmente, si altera el valor de **CurrentFlow** a “**0**”, cualquier componente conectado a continuación no se ejecutará debido a que el flujo de corriente es “**0**”.

Como se aprecia en este ejemplo, insertar código Pawn es muy útil para implementar algoritmos o optimizar la velocidad de ejecución del PLC evitando llamar a componentes para realizar la misma tarea.

24.1.3 Consejos:

- Si desea crear variables intermedias al mismo tiempo, se recomienda llamar a una función Pawn y no utilizar este componente.
- No utilice instrucciones Pawn del tipo "**return**" desde el código insertado, ya que el comportamiento no está definido.
- Cuando necesite implementar grandes cantidades de código, se recomienda utilizar las funciones Pawn (SFC) o los Scripts Pawn (archivos .p). El objetivo de este componente es la de permitir una forma simple y rápida de insertar código.
- Para aprender el lenguaje Pawn en profundidad, ingrese a nuestro sitio Web, disponemos de documentación al respecto. También puede consultar el manual de usuario del PLC para programar con lenguaje Pawn y así conocer las funciones nativas disponibles.

24.2 PawnFunc (Llamar a Función Pawn)



Agrega un componente que permite llamar a una función Pawn definida por el usuario.

Es posible agregar una función Ladder desde el **Explorador de Proyecto** del entorno **StxLadder**. Para crear una función, diríjase a la carpeta "**Scripts**" y sobre la carpeta "**Funciones**" haga click derecho del mouse. Luego seleccione "**Agregar nueva función …**". Una ventana le preguntará el nombre de la función a crear.

El nombre de la función creada, es el nombre de la función Pawn.

Cuando cree la función Pawn, debe especificar las entradas y salidas que utilizará.

Desde las propiedades del componente debe seleccionar la función Pawn a llamar.

El nombre de la función seleccionada se mostrará luego en la parte superior del componente para fácil identificación.

Una función Pawn le será muy útil para crear rutinas de código que deben llamarse repetidamente, pudiendo reutilizar el código escrito.

Se debe prestar especial atención en el código introducido y se recomienda leer la documentación de referencia.

24.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- Las entradas **IN1**, **IN2**, **IN3** e **IN4** permiten pasar un valor como argumento desde el componente a la función Pawn. Su uso es opcional y el tipo de dato depende de cómo fueron definidas la entradas en la función Pawn.

24.2.2 Salidas:

- La salida "**ENO**" retorna el valor del flujo de corriente. Si la función Pawn no retorna un valor con la instrucción "**return**", su valor es una copia del valor del puerto "**EN**". La función Pawn puede retornar un valor para el flujo de corriente con la instrucción "**return**".
- Las salidas **OUT1**, **OUT 2**, **OUT3** e **OUT4** devuelven valores desde la función Pawn al componente. Su uso es opcional y el tipo de dato depende de cómo fueron definidas las salidas en la función Pawn.

24.2.3 Información:

El código Pawn es sumamente útil para modificar variables con factores, procesarlas o llamar a funciones nativas del PLC que no están disponible como componentes Ladder.

Puede implementar algoritmos complejos que en lenguaje Ladder serian muy complicados de lograr.

24.2.4 Ejemplo:

Implementaremos un ejemplo trivial, pero muy didáctico: una función para sumar dos valores.

Primero debemos crear la función desde el **Explorador de Proyecto.** Luego creamos una función Pawn llamada "**Suma**" y definimos "2" entradas y "1" salida, todas del tipo "**Int32**".

Creada la función, la abrimos para introducir código haciendo doble-click en su archivo del **Explorador de Proyecto**.

Introduzca el siguiente código Pawn:

```
OUT1 = IN1 + IN2
```

El código introducido es muy simple, suma las entradas **IN1** e **IN2** y asigna el valor resultante a la salida **OUT1**.

Desde las propiedades del componente "**PawnFunc**" seleccione el nombre de la función Pawn y luego desde las pestañas "Entradas" y "Salidas", asigne dos variables **Int32** para las entradas **IN1** e **IN2**, y una variable tipo **Int32** para la salida **OUT1**.

Ahora puede compilar el proyecto, y el componente llamara a la función Pawn "**Suma**" que sumara dos números. El resultado estará disponible en la variable apuntada por el puerto **OUT1** del componente.

Como se puede apreciar, es muy simple y útil el uso de funciones Pawn.

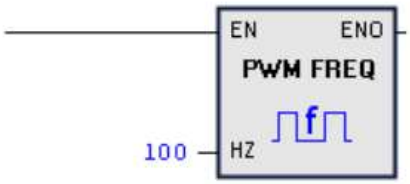
24.2.5 Consejos:

- Si desea insertar código directamente, utilice el componente "**PawnInsert**".
- Puede crear variables dentro de la función Pawn.
- En el código de la función Pawn, puede opcionalmente retornar el valor del flujo de corriente escribiendo con "**return 1**" (para un valor verdadero) o con "**return 0**" (para un valor falso). El flujo de corriente se reflejará en la salida **ENO** del componente.
- Para aprender el lenguaje Pawn en profundidad, ingrese a nuestro sitio Web, disponemos de documentación al respecto. También consulte el manual de usuario del PLC para programar con lenguaje Pawn y así conocer las funciones nativas disponibles.

25 Componentes para Salidas PWM

En esta sección podrá encontrar componentes para operar las salidas PWM (Modulación por Ancho de Pulso) del PLC (solo en modelos que disponen de ellas).

25.1 PwmFreq (Establecer Frecuencia del PWM)



Agrega un componente que permite modificar la frecuencia de las salidas PWM (Modulación por Ancho de Pulso) del PLC.

25.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**HZ**" se especifica la frecuencia del PWM, cuya magnitud esta expresada en Hertz. El puerto puede leer un valor de una constante o variable tipo **Int32**.

25.1.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

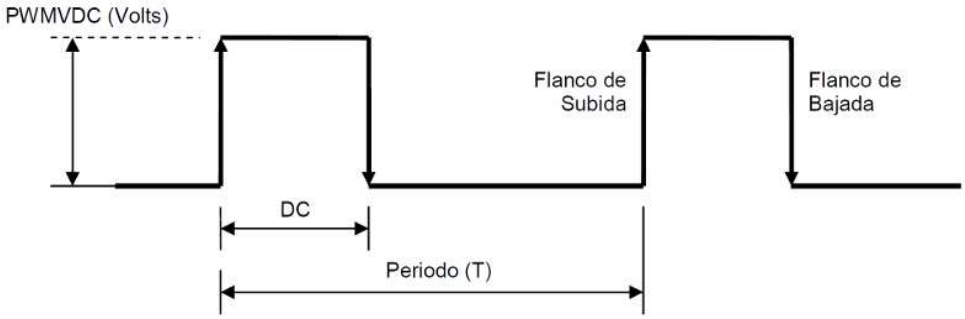
25.1.3 Atención:

- Al cambiar la frecuencia de las salidas PWM, es posible que el duty-cycle de las salidas se modifique a un valor aleatorio.
- Se recomienda, que luego de cambiar la frecuencia, actualice el valor del duty-cycle, para evitar discrepancias.

(Continua en página siguiente)

25.1.4 Forma de onda del PWM:

La siguiente imagen muestra la forma de onda generada por el PWM.



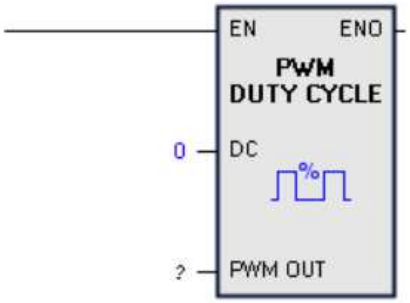
Donde:

- DC (DutyCycle): Ciclo de trabajo de la señal, y se especifica como un porcentaje de 0-100% respecto del total de periodo T.
- Periodo (T): Periodo de la señal.
- PWMVDC: Valor en voltaje de la tensión fuente externa de corriente continúa.
- Frecuencia: Reciproco de T, medido en Hertz, es decir 1/T.

25.1.5 Notas:

- El número de salidas PWM del PLC depende del modelo seleccionado.
- Duty-Cycle se refiere al ciclo de trabajo de la onda cuadrada generada.

25.2 PwmDC (Establecer Duty-Cycle en Salida PWM)



Agrega un componente que permite modificar el ciclo de trabajo (duty-cycle) de las salidas PWM del PLC.

25.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**DC**" se pasa a través de una variable o constante del tipo **Int32** el valor del duty-cycle. El duty-cycle se expresa como un valor entero entre 0 y 1000. El valor 0 representa el 0% y el valor 1000 el 100% del duty-cycle. La resolución es del 0.1 %.
- La entrada "**PWM OUT"** muestra los canales PWM seleccionados, separados por una barra "/" en el caso de varios canales. Se especifica desde las propiedades del componente.

25.2.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

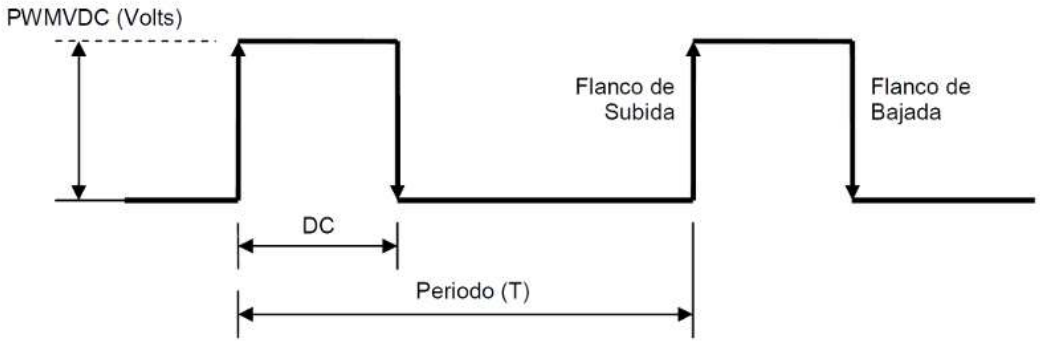
25.2.3 Ejemplos:

- **DC** = 0 representa un 0% de duty-cycle.
- **DC** = 123 representa un 12.3% de duty-cycle.
- **DC** = 568 representa un 56.8% de duty-cycle.
- **DC** = 1000 representa un 100% de duty-cycle.

(Continua en página siguiente)

25.2.4 Forma de onda del PWM:

La siguiente imagen muestra la forma de onda generada por el PWM.



Donde:

- DC (DutyCycle): Ciclo de trabajo de la señal, y se especifica como un porcentaje de 0-100% respecto del total de periodo T.
- Periodo (T): Periodo de la señal.
- PWMVDC: Valor en voltaje de la tensión fuente externa de corriente continúa.
- Frecuencia: Reciproco de T, medido en Hertz, es decir 1/T.

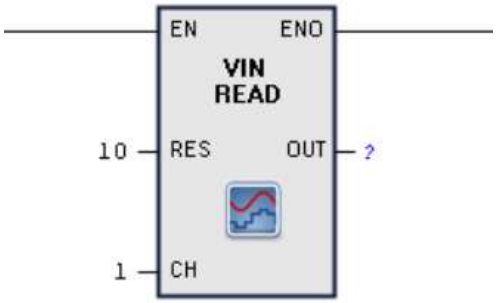
25.2.5 Notas:

- El número de salidas PWM del PLC depende del modelo seleccionado.
- Duty-Cycle se refiere al ciclo de trabajo de la onda cuadrada generada.
- Es posible establecer el duty-cycle a varios canales PWM simultáneamente.

26 Componentes para Entradas Analógicas

Componentes para obtener lecturas de las entradas analógicas del PLC (si están disponibles en el modelo seleccionado). Las entradas analógicas le permiten conectar sensores y realizar mediciones de magnitudes analógicas.

26.1 VinRead (Leer Entrada)



Agrega un componente que permite leer una entrada analógica.

Al ejecutar este componente, el PLC realizará una conversión analógica a digital (A/D).

26.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**RES**" se especifica la resolución de lectura de la conversión. El valor se determina automáticamente. Depende del modelo de PLC. En las propiedades del componente se mostrará el valor utilizado.
- En la entrada "**CH**" se especifica el canal o entrada analógica a leer.

26.1.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".
- En el puerto "**OUT**" se retorna el resultado de la conversión A/D en una variable del tipo **Int32**. Un resultado correcto es un valor positivo e igual o mayor a cero. Pero si el valor de la conversión es negativo (-1 por ejemplo) existe un error (que puede ser causado por un **sampler** que este activado en el canal).

26.1.3 Resolución:

La resolución de la conversión depende del modelo del PLC, pero a efectos prácticos, mostramos a continuación los posibles valores de la muestra obtenida:

- Si la resolución del canal es 8-bits, el rango de valores de salida va desde 0 a 255.
- Si la resolución del canal es 10-bits, el rango de valores de salida va desde 0 a 1023.
- Si la resolución del canal es 12-bits, el rango de valores de salida va desde 0 a 4095.

26.1.4 Consejos:

- Puede activar los filtros digitales (ver componentes **Vin12Filter** y **Vin34Filter**) si tiene señales que varían muy lentamente en las entradas analógicas y desea eliminar pequeños ruidos o perturbaciones. Una vez activados los filtros, este componente devuelve un valor filtrado de la entrada.

- Si los filtros digitales del PLC no están activados, este componente accede inmediatamente a la conversión analógica/digital de la entrada.

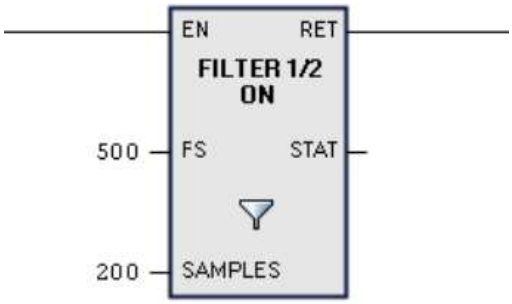
26.1.5 Notas:

- Es posible que la conversión no esté disponible, si algún **sampler** del PLC esta activado sobre el canal.

- Algunos modelos de PLC permiten modificar el rango de voltaje de entrada admitido en el canal, remitirse a la documentación.

- La cantidad de entradas analógicas depende del modelo de PLC seleccionado.

26.2 Vin12Filter (Filtro para Entradas VIN1 y VIN2)



Agrega un componente permite activar o desactivar el **filter 1 / 2 (filtro digital)** del PLC para las entradas analógicas VIN1 y VIN2.

Los filtros digitales del PLC le permiten eliminar pequeñas fluctuaciones o ruido de las entradas analógicas.

Se aconseja utilizar los filtros digitales cuando hay señales en las entradas analógicas que varían muy lentamente (por ejemplo el valor de un potenciómetro, un sensor de temperatura o una tensión cuasi constante).

El filtro digital utilizado por este componente es del tipo "**pasa bajos**" y funciona promediando muestras de las entradas analógicas.

En las propiedades del componente se especifica si el PLC activa (ON) o desactiva (OFF) el filtro digital.

Una vez activado el filtro digital, el componente **VinRead** devolverá conversiones analógicas filtradas de los canales afectados por el filtrado.

26.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**FS**" se especifica la frecuencia de muestreo en Hertz. Este parámetro indica la velocidad a la que se leerá el canal analógico para tomar cada muestra. Las frecuencias límites varían de acuerdo al modelo del PLC, pero el componente las limita automáticamente de acuerdo al modelo elegido en el proyecto.
- En el puerto "**SAMPLES**" se especifica la cantidad de muestras a tomar antes de realizar el promedio (filtrado). Tenga en cuenta que las muestras serán tomadas a la velocidad "**FS**".

26.2.2 Salidas:

- La salida "**RET**" del componente es verdadera si el evento pudo configurarse con éxito.
- La salida "**STAT**" del componente, es de uso opcional, pero permite determinar si el **filter** pudo ser activado. Es posible especificar una variable que almacenará un código de error. Si el retorno es cero, la operación es exitosa, si es negativo se produjo un error de activación.
    - Retorno 0: Operación exitosa.
    - Retorno -10: Error, operación desconocida.
    - Retorno -11: Error, el **sampler 1 / 2** esta activado.
    - Retorno -12: Error, el filtro ya esta activado. Desactive y vuelva activar para aplicar cambios.

26.2.3 Importante:

- Al activar el **filter 1 / 2**, no es posible utilizar el sampler **1 / 2,** y viceversa.

26.2.4 Notas:

- El filtro digital necesita un tiempo igual a (1/**FS** * **SAMPLES**) segundos para actualizar su salida. Note que el tiempo descripto anteriormente determina la tasa de refresco de las entradas analógicas afectadas, esto quiere decir que si necesita medir señales muy rápidas, no es aconsejable el uso del filtro digital.

- El filtro digital funciona adquiriendo "**SAMPLES**" muestras a una frecuencia "**FS**" y luego promediándolas cada (1/**FS** * **SAMPLES**) segundos.

- El filtro digital funciona independientemente de la ejecución del programa del PLC.

26.2.5 Consejos:

- Un **filter** activado consume recursos. A menor frecuencia "**FS**", menor es la sobrecarga de procesamiento para el PLC. Utilice el menor valor posible.

- A mayor valor de **SAMPLES**, más estable será la lectura, pero la respuesta será más lenta.

- Utilice el filtrado digital para medir señales que cambian muy lentamente en el tiempo. Con el filtrado digital puede eliminar pequeñas variaciones de la señal de entrada y obtener lecturas más estables.

- No confunda EXACTITUD con PRECISION. La exactitud es la diferencia entre la realidad y el valor medido por la entrada analógica. La precisión está asociada a la repetición del valor medido en el tiempo (cuanto defiere según transcurre un tiempo). Si tiene problemas de PRESICION, es conveniente utilizar un filtro digital.

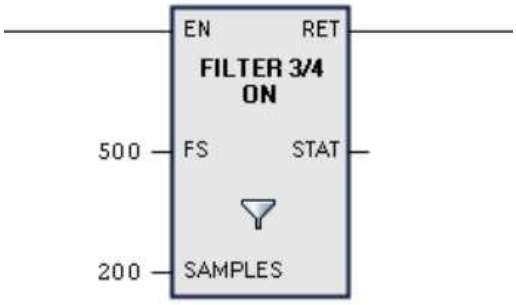
- Usualmente este componente se llama una sola vez desde el diagrama "Inicio".

- Hay más instrucciones Pawn que puede utilizar para las entradas analógicas. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.

26.2.6 Ejemplo:

- Si **SAMPLES** = 200 y **FS** = 500, la lectura de la entrada analógica será actualizada cada: 1/**FS** * **SAMPLES** = 1/500 * 200 = 0.4 segundos. Es decir, podrá registrar cambios cada 0.4 segundos sobre la entrada analógica.

26.3 Vin34Filter (Filtro para Entradas VIN3 y VIN4)



Agrega un componente permite activar o desactivar el **filter 3 / 4 (filtro digital)** del PLC para las entradas analógicas VIN3 y VIN4.

Los filtros digitales del PLC le permiten eliminar pequeñas fluctuaciones o ruido de las entradas analógicas.

Se aconseja utilizar los filtros digitales cuando hay señales en las entradas analógicas que varían muy lentamente (por ejemplo el valor de un potenciómetro, un sensor de temperatura o una tensión cuasi constante).

El filtro digital utilizado por este componente es del tipo "**pasa bajos**" y funciona promediando muestras de las entradas analógicas.

En las propiedades del componente se especifica si el PLC activa (ON) o desactiva (OFF) el filtro digital.

Una vez activado el filtro digital, el componente **VinRead** devolverá conversiones analógicas filtradas de los canales afectados por el filtrado.

26.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**FS**" se especifica la frecuencia de muestreo en Hertz. Este parámetro indica la velocidad a la que se leerá el canal analógico para tomar cada muestra. Las frecuencias límites varían de acuerdo al modelo del PLC, pero el componente las limita automáticamente de acuerdo al modelo elegido en el proyecto.
- En el puerto "**SAMPLES**" se especifica la cantidad de muestras a tomar antes de realizar el promedio (filtrado). Tenga en cuenta que las muestras serán tomadas a la velocidad "**FS**".

26.3.2 Salidas:

- La salida "**RET**" del componente es verdadera si el evento pudo configurarse con éxito.
- La salida "**STAT**" del componente, es de uso opcional, pero permite determinar si el **filter** pudo ser activado. Es posible especificar una variable que almacenará un código de error. Si el retorno es cero, la operación es exitosa, si es negativo se produjo un error de activación.
    - Retorno 0: Operación exitosa.
    - Retorno -10: Error, operación desconocida.
    - Retorno -11: Error, el **sampler 3 / 4** esta activado.
    - Retorno -12: Error, el filtro ya esta activado. Desactive y vuelva activar para aplicar cambios.

26.3.3 Importante:

- Al activar el **filter 3 / 4**, no es posible utilizar el sampler **3 / 4,** y viceversa.

26.3.4 Notas:

- El filtro digital necesita un tiempo igual a (1/**FS** * **SAMPLES**) segundos para actualizar su salida. Note que el tiempo descripto anteriormente determina la tasa de refresco de las entradas analógicas afectadas, esto quiere decir que si necesita medir señales muy rápidas, no es aconsejable el uso del filtro digital.

- El filtro digital funciona adquiriendo "**SAMPLES**" muestras a una frecuencia "**FS**" y luego promediándolas cada (1/**FS** * **SAMPLES**) segundos.

- El filtro digital funciona independientemente de la ejecución del programa del PLC.

26.3.5 Consejos:

- Un **filter** activado consume recursos. A menor frecuencia "**FS**", menor es la sobrecarga de procesamiento para el PLC. Utilice el menor valor posible.

- A mayor valor de **SAMPLES**, más estable será la lectura, pero la respuesta será más lenta.

- Utilice el filtrado digital para medir señales que cambian muy lentamente en el tiempo. Con el filtrado digital puede eliminar pequeñas variaciones de la señal de entrada y obtener lecturas más estables.

- No confunda EXACTITUD con PRECISION. La exactitud es la diferencia entre la realidad y el valor medido por la entrada analógica. La precisión está asociada a la repetición del valor medido en el tiempo (cuanto defiere según transcurre un tiempo). Si tiene problemas de PRESICION, es conveniente utilizar un filtro digital.

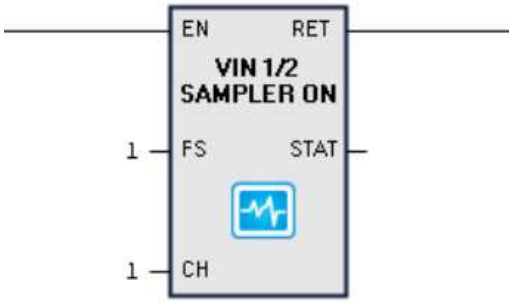
- Usualmente este componente se llama una sola vez desde el diagrama "Inicio".

- Hay más instrucciones Pawn que puede utilizar para las entradas analógicas. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.

26.3.6 Ejemplo:

- Si **SAMPLES** = 200 y **FS** = 500, la lectura de la entrada analógica será actualizada cada: 1/**FS** * **SAMPLES** = 1/500 * 200 = 0.4 segundos. Es decir, podrá registrar cambios cada 0.4 segundos sobre la entrada analógica.

26.4 Vin12SamplerOn (Activar Sampler 1 / 2)



Agrega un componente permite activar el **sampler 1 / 2** del PLC.

Los **samplers** permiten muestrear entradas analógicas del PLC a una frecuencia determinada por el usuario.

Las muestras del canal o entrada analógica son almacenadas en un buffer interno del PLC. Una vez que el buffer interno del canal se llena, se genera un evento y el usuario puede leer las muestras recolectadas.

El proceso de muestreo se lleva a cabo en segundo plano, es decir, se ejecuta independientemente del programa principal del usuario.

El usuario debe crear un diagrama-evento que será el encargado de procesar las muestras una vez que se llene el buffer.

En el diagrama-evento el usuario puede leer o grabar las muestras recolectadas. El proceso se repetirá indefinidamente hasta el **sampler** sea desactivado.

Es posible definir el diagrama del evento desde el **Explorador de Proyecto** del entorno **StxLadder**.

Este componente puede generar dos eventos que indican buffer lleno:

1. **OnVin1BufFull** : Se genera cuando el buffer del sampler para el canal **VIN1** está lleno.
2. **OnVin2BufFull** : Se genera cuando el buffer del sampler para el canal **VIN2** está lleno.

26.4.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**FS**" se especifica la frecuencia de muestreo. La unidad de muestreo es el Hertz. Las frecuencias límites varían de acuerdo al modelo del PLC, pero el componente las limita automáticamente de acuerdo al modelo elegido en el proyecto.
- En el puerto "**CH**" se especifican los canales a muestrear. Varios canales pueden especificarse al mismo tiempo. Recuerde que cada canal genera un evento de buffer lleno.

26.4.2 Salidas:

- La salida "**RET**" del componente es verdadera si el evento pudo configurarse con éxito.
- La salida "**STAT**" del componente, es de uso opcional, pero permite determinar si el **sampler** pudo ser activado. Es posible especificar una variable que almacenará un código de error. Si el retorno es cero, la operación es exitosa, si es negativo se produjo un error de activación.
    - Retorno 0: Operación exitosa.
    - Retorno -1: Error, canal seleccionado no es válido.
    - Retorno -10: Error, operación desconocida.
    - Retorno -11: Error, el **filter 1 / 2** esta activado.
    - Retorno -12: Error, el sampler ya esta activado. Desactive y vuelva activar para aplicar cambios.

26.4.3 Importante:

- Al activar el **sampler 1 / 2**, no es posible utilizar el **filter 1 / 2,** y viceversa.

26.4.4 Notas:

- Recuerde crear el diagrama para procesar el o los eventos del **sampler**.
- La resolución de las muestras del **sampler** normalmente es de 8-bits, pero puede variar según modelo del PLC.
- Cuando un **sampler** es activado, algunas entradas analógicas no podrán ser leídas con componentes de lectura A/D. Consulte documentación.
- En general, para que un evento de buffer lleno se genere, 127 muestras deben ser adquiridas primero (esto depende del modelo de PLC).
- Utilice el componente **VinSamplerGetBuf** para recolectar las muestras del buffer del **sampler**.
- Un buffer es una memoria temporal, que sirve en este caso, para guardar muestras del **sampler**.

26.4.5 Consejos:

- Un **sampler** activado consume recursos y tiempo del procesador del PLC. Si no los utiliza constantemente en su proyecto, desactívelos cuando no sean necesarios. También puede optimizar el rendimiento general del sistema disminuyendo la velocidad de muestreo.
- Es posible mediante el método de Polling y el componente **VinSamplerGetBuf** determinar si hay muestras disponibles, sin utilizar eventos.
- Usualmente este componente se llama una sola vez desde el diagrama "Inicio".
- Hay más instrucciones Pawn que puede utilizar para las entradas analógicas. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.

26.5 Vin12SamplerOff (Desactivar Sampler 1 / 2)

Agrega un componente que permite desactivar el **sampler 1 / 2** del PLC.

Con este componente puede desactivar los eventos **OnVin1BufFull** y **OnVin2BufFull**.

26.5.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

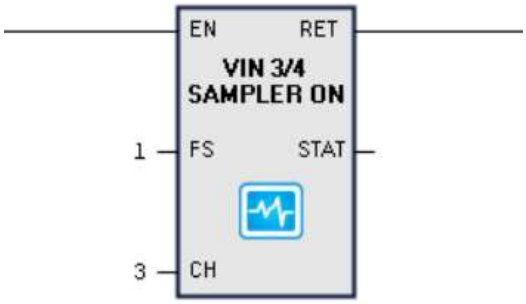
26.5.2 Salidas:

- La salida "**RET**" del componente es verdadera si el evento pudo desactivarse con éxito.
- La salida "**STAT**" del componente, es de uso opcional, pero permite determinar si el **sampler** pudo ser desactivado. Es posible especificar una variable que almacenará un código de error. Si el retorno es cero, la operación es exitosa, si es negativo se produjo un error de activación.
    - Retorno 0: Operación exitosa.
    - Retorno -10: Error, operación desconocida.

26.5.3 Consejos:

- Un **sampler** activado consume recursos y tiempo del procesador del PLC. Si no los utiliza constantemente en su proyecto, desactívelos cuando no sean necesarios. También puede optimizar el rendimiento general del sistema disminuyendo la velocidad de muestreo.

26.6 Vin34SamplerOn (Activar Sampler 3 / 4)



Agrega un componente permite activar el **sampler 3 / 4** del PLC.

Los **samplers** permiten muestrear entradas analógicas del PLC a una frecuencia determinada por el usuario.

Las muestras del canal o entrada analógica son almacenadas en un buffer interno del PLC. Una vez que el buffer interno del canal se llena, se genera un evento y el usuario puede leer las muestras recolectadas.

El proceso de muestreo se lleva a cabo en segundo plano, es decir, se ejecuta independientemente del programa principal del usuario.

El usuario debe crear un diagrama-evento que será el encargado de procesar las muestras una vez que se llene el buffer.

En el diagrama-evento el usuario puede leer o grabar las muestras recolectadas. El proceso se repetirá indefinidamente hasta el **sampler** sea desactivado.

Es posible definir el diagrama del evento desde el **Explorador de Proyecto** del entorno **StxLadder**.

Este componente puede generar dos eventos que indican buffer lleno:

1. **OnVin3BufFull** : Se genera cuando el buffer del sampler para el canal **VIN3** está lleno.
2. **OnVin4BufFull** : Se genera cuando el buffer del sampler para el canal **VIN4** está lleno.

26.6.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En el puerto "**FS**" se especifica la frecuencia de muestreo. La unidad de muestreo es el Hertz. Las frecuencias límites varían de acuerdo al modelo del PLC, pero el componente las limita automáticamente de acuerdo al modelo elegido en el proyecto.
- En el puerto "**CH**" se especifican los canales a muestrear. Varios canales pueden especificarse al mismo tiempo. Recuerde que cada canal genera un evento de buffer lleno.

26.6.2 Salidas:

- La salida "**RET**" del componente es verdadera si el evento pudo configurarse con éxito.
- La salida "**STAT**" del componente, es de uso opcional, pero permite determinar si el **sampler** pudo ser activado. Es posible especificar una variable que almacenará un código de error. Si el retorno es cero, la operación es exitosa, si es negativo se produjo un error de activación.
    - Retorno 0: Operación exitosa.
    - Retorno -1: Error, canal seleccionado no es válido.
    - Retorno -10: Error, operación desconocida.
    - Retorno -11: Error, el **filter 3 / 4** esta activado.
    - Retorno -12: Error, el sampler ya esta activado. Desactive y vuelva activar para aplicar cambios.

26.6.3 Importante:

- Al activar el **sampler 3 / 4**, no es posible utilizar el **filter 3 / 4,** y viceversa.

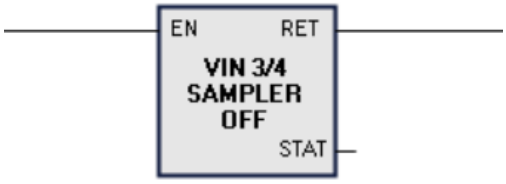
26.6.4 Notas:

- Recuerde crear el diagrama para procesar el o los eventos del **sampler**.
- La resolución de las muestras del **sampler** normalmente es de 8-bits, pero puede variar según modelo del PLC.
- Cuando un **sampler** es activado, algunas entradas analógicas no podrán ser leídas con componentes de lectura A/D. Consulte documentación.
- En general, para que un evento de buffer lleno se genere, 127 muestras deben ser adquiridas primero (esto depende del modelo de PLC).
- Utilice el componente **VinSamplerGetBuf** para recolectar las muestras del buffer del **sampler**.
- Un buffer es una memoria temporal, que sirve en este caso, para guardar muestras del **sampler**.

26.6.5 Consejos:

- Un **sampler** activado consume recursos y tiempo del procesador del PLC. Si no los utiliza constantemente en su proyecto, desactívelos cuando no sean necesarios. También puede optimizar el rendimiento general del sistema disminuyendo la velocidad de muestreo.
- Es posible mediante el método de Polling y el componente **VinSamplerGetBuf** determinar si hay muestras disponibles, sin utilizar eventos.
- Usualmente este componente se llama una sola vez desde el diagrama "Inicio".
- Hay más instrucciones Pawn que puede utilizar para las entradas analógicas. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.

26.7 Vin34SamplerOff (Desactivar Sampler 3 / 4)



Agrega un componente que permite desactivar el **sampler 3 / 4** del PLC.

Con este componente puede desactivar los eventos **OnVin3BufFull** y **OnVin4BufFull**.

26.7.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

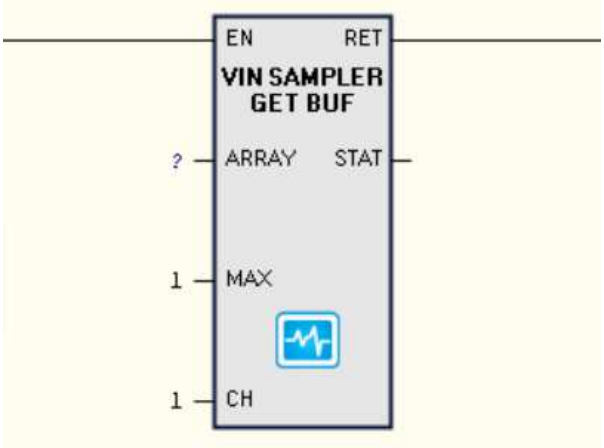
26.7.2 Salidas:

- La salida "**RET**" del componente es verdadera si el evento pudo desactivarse con éxito.
- La salida "**STAT**" del componente, es de uso opcional, pero permite determinar si el **sampler** pudo ser desactivado. Es posible especificar una variable que almacenará un código de error. Si el retorno es cero, la operación es exitosa, si es negativo se produjo un error de activación.
    - Retorno 0: Operación exitosa.
    - Retorno -10: Error, operación desconocida.

26.7.3 Consejos:

- Un **sampler** activado consume recursos y tiempo del procesador del PLC. Si no los utiliza constantemente en su proyecto, desactívelos cuando no sean necesarios. También puede optimizar el rendimiento general del sistema disminuyendo la velocidad de muestreo.

26.8 VinSamplerGetBuf (Obtener Buffer de Muestras de un Sampler)



Agrega un componente que permite obtener las muestras recolectadas desde los buffers internos de los **samplers** del PLC.

Usualmente, este componente se llama al principio de un evento de buffer lleno para algún sampler, se obtienen las muestras y luego se procesan o se copian en una variable global.

26.8.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**ARRAY**" debe apuntar a un array del tipo **Int32_Array**, donde se copiarán las muestras almacenadas en el **sampler**. El componente le preguntara a partir de que elemento del array se copiaran las muestras. Se copia una muestra por cada elemento del array.
- La entrada "**MAX**" especifica el número máximo de muestras a copiar en el array. Las muestras se copian a partir del índice especificado en "**ARRAY**" hasta la cantidad de elementos especificados por "**MAX**". Asegúrese de tener un array lo suficientemente grande para almacenar las muestras solicitadas.
- La entrada "**CH**" selecciona el canal analógico de donde se obtendrá el buffer interno para copiar las muestras.

26.8.2 Salidas:

- La salida "**RET**" del componente es verdadera solo si se copiaron desde el **sampler** las muestras al array especificado en el puerto "**ARRAY**". De lo contrario permanece falsa.
- La salida "**STAT**" del componente es opcional y nos permite copiar un código de error en una variable del tipo **Int32**, que representa el resultado de la operación:
  - Salida 0 : Operación exitosa. Muestras copiadas al array.
  - Salida -1: No hay muestras disponibles para copiar. Esto se puede deber a que el buffer interno del canal no fue llenado aún o el **sampler** no está activado. Puede ser útil este valor de retorno para comprobar si hay muestras disponibles (polling).
  - Salida -2 : Error, el número máximo de muestras a copiar (**MAX**) es invalido.

- Salida -3 : Error, el array pasado para almacenar las muestras es incorrecto.

26.8.3 Notas:

- Recuerde activar los **samplers** para llenar los buffer internos con muestras. Utilice los componentes **Vin12SamplerOn** y **Vin34SamplerOn**.
- Las muestras se copian una sola vez, luego el buffer interno se vacía forzosamente, independientemente a si leyó todas las muestras.
- En general, para que un evento de buffer lleno se genere, 127 muestras deben ser adquiridas primero (esto depende del modelo de PLC).
- La resolución de las muestras del **sampler** normalmente es de 8-bits, pero puede variar según modelo del PLC.

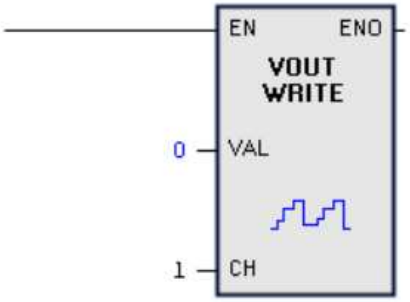
26.8.4 Consejos:

- Puede utilizar el valor de la salida "**STAT**" para hacer polling y comprobar constantemente si hay muestras disponibles.
- Hay más instrucciones Pawn que puede utilizar para las entradas analógicas. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.

27 Componentes para Salidas Analógicas

En esta sección puede encontrar componentes para controlar las salidas analógicas del PLC (si el modelo dispone). Los componentes le permiten establecer voltajes y hasta generar forma de ondas senoidales con un generador embebido.

27.1 VoutWrite (Escribir en Salida Analógica)



Agrega un componente que permite escribir un valor entero (digital) sobre la salida analógica especificada.

Al ejecutar este componente, el PLC realizará una conversión digital a analógica (D/A).

27.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**VAL"** se especifica una variable o constante del tipo **Int32**, cuyo valor se escribirá sobre la salida analógica.
- En la entrada "**CH**" se especifica el canal o salida analógica a escribir.

27.1.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

27.1.3 Resolución:

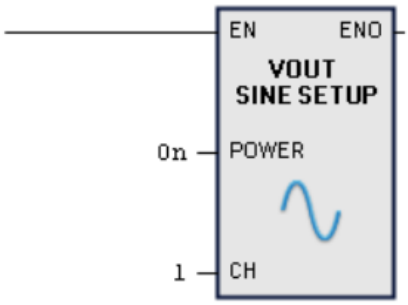
La resolución de la conversión D/A depende del modelo del PLC, pero a efectos prácticos, mostramos a continuación el rango de valores que pueden escribirse en la entrada **VAL**:

- Si la resolución del canal es 8-bits, el rango de valores va desde 0 a 255.
- Si la resolución del canal es 10-bits, el rango de valores va desde 0 a 1023.
- Si la resolución del canal es 12-bits, el rango de valores va desde 0 a 4095.

27.1.4 Notas:

- Algunos modelos de PLC, permiten generar una onda senoidal sobre las salidas analógicas de forma simple. Más información en el manual de usuario del PLC.
- La cantidad de salidas analógicas depende del modelo de PLC seleccionado.

27.2 VoutSineGenSetup (Configurar Generador Senoidal)



Agrega un componente que permite configurar el generador de onda sinusoidal sobre la salida analógica especificada.

27.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**POWER**" se especifica si el generador se activa (On) o desactiva (Off).
- En la entrada "**CH**" se especifica el canal o salida analógica utilizada por el generador.

27.2.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

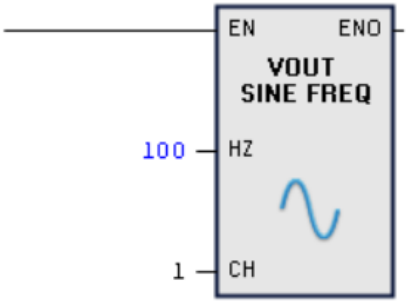
27.2.3 Notas:

- El uso del generador sinusoidal, agrega una pequeña sobrecarga al procesador del **PLC** debido que se genera mediante técnicas digitales, por lo tanto, si no lo usa continuamente en su proyecto, actívelo solo cuando sea necesario y luego vuelva a desactivarlo.
- La amplitud de la onda de salida, puede modificarse mediante jumpers en algunos modelos de PLC, remítase a la documentación.
- La frecuencia del generador puede modificarse con el componente **VoutSineGenFreq**.
- Cuando utiliza el generador, no es posible escribir valores digitales en la salida analógica empleada por el generador.

27.2.4 Aplicaciones:

- Tonos de audio.
- Trasmisión de datos por modulación **FSK** (Frequency Shift Keying).
- Barrido de frecuencias para sistemas mecánicos, determinación del ancho de banda.
- Elaboración de diagramas de bode, respuesta en frecuencia.
- Prueba de válvulas hidráulicas proporcionales.
- Etc…

27.3 VoutSineGenFreq (Modificar Frecuencia del Generador Senoidal)



Agrega un componente que permite modificar la frecuencia de la onda sinusoidal generada sobre la salida analógica especificada.

27.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- En la entrada "**HZ**" se especifica la frecuencia de la onda en magnitud Hertz, a partir de una variable o constante del tipo **Int32**.
- En la entrada "**CH**" se especifica el canal o salida analógica utilizada por el generador.

27.3.2 Salidas:

- La salida "**ENO**" del componente retorna una copia del valor de la entrada "**EN**".

27.3.3 Importante:

- La frecuencia máxima o mínima, se encuentra especificada en la hoja de datos del PLC.

27.3.4 Notas:

- Recuerde activar el generador sinusoidal previamente con el componente **VoutSineGenEnable** antes de utilizar por primera vez este componente.

27.3.5 Aplicaciones:

- Tonos de audio.
- Trasmisión de datos por modulación **FSK** (Frequency Shift Keying).
- Barrido de frecuencias para sistemas mecánicos, determinación del ancho de banda.
- Elaboración de diagramas de bode, respuesta en frecuencia.
- Prueba de válvulas hidráulicas proporcionales.
- Etc…

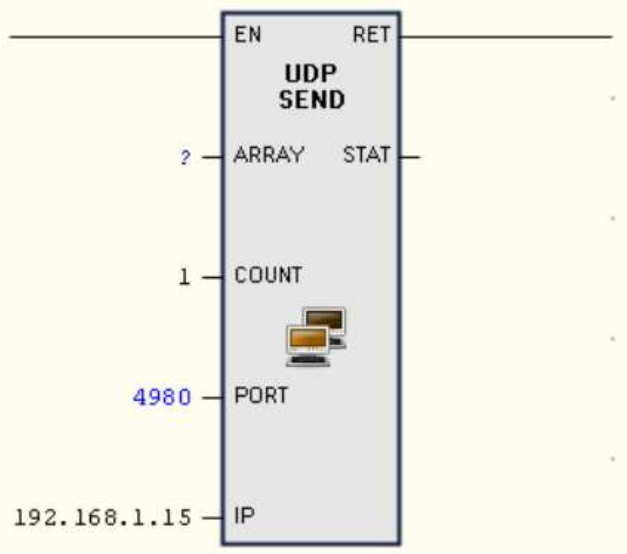
28 Componentes para Conexión Ethernet / UDP

Los componentes de esta sección le permitirán transmitir y recibir datos desde la red Ethernet a la cual el PLC este conectado. Las conexiones se realizan a través del protocolo UDP (User Datagram Protocol), por lo tanto puede transmitir y recibir datos desde internet a una dirección IP fija.

En nuestro sitio Web puede bajar ejemplos y librerías en Microsoft Visual C# para crear sus propios programas en Windows e intercambiar datos desde una computadora y el PLC.

Nota: Vea la pagina 43 para comunicarse con un panel HMI vía Ethernet.

28.1 UdpSend (Enviar Datos vía UDP)



Agrega un componente que permite enviar datos utilizando el protocolo UDP a través del puerto Ethernet del PLC.

28.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**ARRAY**" debe apuntar a un array del tipo **Int32_Array**, que contiene los datos a enviar. Si los datos no están empaquetados (PACKED), solo transmite el byte menos significativo de cada elemento del array. Lea más abajo (Datos Empaquetados).
- La entrada "**COUNT**" establece la cantidad de elementos que se enviaran del array. No puede superar el tamaño del array.
- En la entrada "**PORT**" se especifica una constante o variable del tipo **Int32**, que define el puerto UDP de destino.
- En la entrada "**IP**" se especifica la dirección internet de destino.

28.1.2 Salidas:

- La salida "**RET**" es verdadera o "**1**" si no existieron errores. Si es "**0**", existió un error en la transmisión. Puede encontrar más información del error en el puerto "**STAT**".

- En la salida "**STAT**" se retorna un código de error en una variable tipo **Int32** (uso opcional):

    - El valor retornado es "**0**" si la transmisión fue realizada sin errores.
    - El valor es "**-1**" si el tamaño especificado en "**COUNT**" es incorrecto.
    - El valor es "**-2**" si la dirección del **ARRAY** no es válida.

28.1.3 Ejemplo:

Para transmitir datos a un dispositivo remoto o computadora, primero debemos conocer la dirección IP y puerto UDP donde el dispositivo remoto escucha por una conexión UDP.

En el puerto **ARRAY** debemos especificar los datos a enviar. Si tenemos un array de 6 elementos, podemos enviarlos a todos especificando el valor 6 en **COUNT**.

Si el dispositivo remoto tiene una IP igual a "192.168.1.15" y puerto 4980, los especificamos en los puertos **IP** y **PORT**.

28.1.4 Datos Empaquetados (PACKED)

Es posible especificar si los datos a transmitir en el puerto **ARRAY** están o no empaquetados (PACKED). En las propiedades del componente se puede tildar la casilla "**Packed data**" para declarar que los datos fueron empaquetados previamente.

Cuando los datos están empaquetados en **ARRAY**, significa el componente transmitirá los 4 bytes de cada elemento **Int32** contenido en el Array. De lo contrario solo se transmite el byte menos significativo de cada elemento.

El empaquetamiento es útil para ahorrar memoria RAM o para enviar valores de 32 bits (**Int32**) por la red.

28.1.5 Notas:

- UDP (User Datagram Protocol) es un protocolo estándar de internet para la transmisión de datos sin handshaking.

- IP (Internet Protocol) es un protocolo estándar de internet, cuyo direccionamiento es una ***dirección IP*** que se utiliza para identificar inequívocamente un dispositivo conectado a la red. Una dirección IP puede ser por ejemplo: 192.168.1.15.

- Un puerto UDP es un número entero que va desde 0 a 65.535. El puerto 0 está reservado y no es aconsejable utilizar ninguno hasta el numero 1023. Un valor utilizado comúnmente en nuestros dispositivos el 4982, pero puede utilizar otro si encuentra alguna interferencia.

- Con el programa **PawnUdpRx** de Slicetex Electronics puede utilizar su computadora como receptor y escuchar los bytes recibidos desde el PLC. Muy útil para realizar pruebas. También ofrecemos de una librería en Microsoft Visual C# para que pueda adquirir datos del PLC y realizar programas profesionales en muy poco tiempo. Consulte nuestro sitio de internet para descargar software y documentación.

28.2 UdpOnRxSetEvent (Activar Evento OnUdpRx)

Agrega un componente que permite activar el evento "**OnUdpRx**".

El evento **OnUdpRx** solo será llamado cuando se reciban datos UDP desde un dispositivo remoto conectado a la red Ethernet del PLC.

Al dispararse el evento, el PLC llama al diagrama asignado para manejar el evento. Dicho evento se define desde el **Explorador de Proyecto** del entorno **StxLadder**. El programador procesará el evento con la lógica especificada en el diagrama.

El evento **OnUdpRx** es un mecanismo muy útil para recibir datos desde dispositivos remotos de forma muy simple.

28.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.

28.2.2 Salidas:

- La salida "**STAT**" del componente es verdadera si el evento pudo configurarse con éxito.

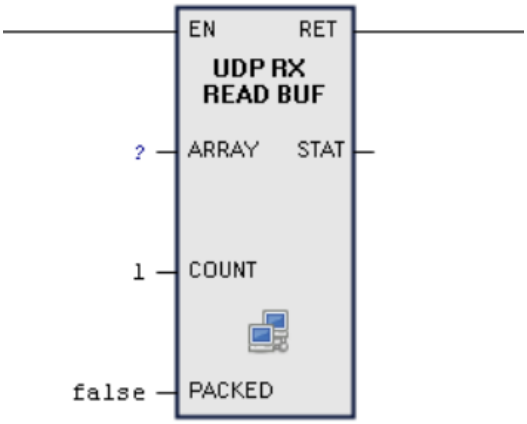
28.2.3 Notas:

- Recuerde crear el diagrama para procesar el evento.
- Una vez activado el evento, el PLC aceptará conexiones UDP provenientes de la red Ethernet.

28.2.4 Consejos:

- Puede utilizar el programa **PawnUdpTx** de Slicetex Electronics para enviar datos UDP desde su PC al PLC de forma simple. También proveemos una librería en Microsoft Visual C# para crear programas profesionales y enviar datos al PLC.

28.3 UdpRxDataRead (Obtener Datos Recibidos por UDP)



Agrega un componente que permite obtener los datos recibidos desde una conexión UDP al PLC.

El componente lee el buffer interno del PLC que almacena el paquete UDP recibido.

28.3.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**ARRAY**" debe apuntar a un array del tipo **Int32_Array**, donde se copiarán los bytes recibidos. El componente le preguntara a partir de que elemento se copiaran los bytes recibidos.
- La entrada "**COUNT**" establece la cantidad de bytes que se copiaran al **ARRAY**, máximo 252. Si el valor es "0" se copiaran todos los bytes recibidos (debe asegurarse de no sobrepasar el tamaño del **ARRAY**).
- La entrada "**PACKED**", si es "true", se empaquetaran 4 bytes recibidos por cada elemento del **ARRAY**. Si es "false" se copiara un byte por cada elemento del **ARRAY**.

28.3.2 Salidas:

- La salida "**RET**" es verdadera o "**1**" si pudieron copiarse bytes al **ARRAY**. Si es "**0**" no se copiaron bytes porque existió un error o el buffer está vacío. Puede encontrar más información del error en el puerto "**STAT**".
- En la salida "**STAT**" se retorna un código de error en una variable tipo **Int32** (uso opcional):
    - El valor retornado es "**0**" si se copiaron bytes al **ARRAY**.
    - Si el valor retornado es "**-1**", no se copiaron bytes debido a que el buffer de recepción esta vacio.
    - El valor es "**-2**" si el tamaño especificado en "**COUNT**" es incorrecto.
    - El valor es "**-4**" si la dirección del **ARRAY** no es válida.

28.3.3 Ejemplo:

En general, este componente se llama al inicio del diagrama que maneja el evento "**OnUdpRx**", que se llama cuando el buffer de recepción UDP del PLC contiene un nuevo paquete de datos.

Entonces, ubique este componente en el diagrama del evento "**OnUdpRx**", obtenga los datos recibidos en el puerto **ARRAY** y luego procéseselo o utilícelo desde otros diagramas, compartiendo los datos con un array global.

Puede llamar a este componente constantemente y verificar si hay nuevos datos cuando la salida "**STAT**" sea igual a "**0**".

28.3.4 Datos Empaquetados (PACKED)

Es posible especificar si los datos a copiar en el puerto **ARRAY** están o no empaquetados (**PACKED**). En las propiedades del componente se puede tildar la casilla "**Packed data**" para declarar que los datos serán empaquetados previamente.

Cuando los datos están empaquetados en **ARRAY**, significa que hay 4 bytes por cada elemento **Int32** contenido en el array. De lo contrario solo se escribe un byte por cada elemento.

Si recibió 4 bytes, y **PACKED** = true, solo le hará falta un elemento **Int32** del **ARRAY** para almacenarlo. De lo contrario se utilizarán 4 elementos **Int32** del **ARRAY** para copiarlos.

El empaquetamiento es útil para ahorrar memoria RAM.

28.3.5 Consejos:

- Puede utilizar el programa **PawnUdpTx** de Slicetex Electronics para enviar datos UDP desde su PC al PLC de forma simple. También proveemos una librería en Microsoft Visual C# para crear programas profesionales y enviar datos al PLC.

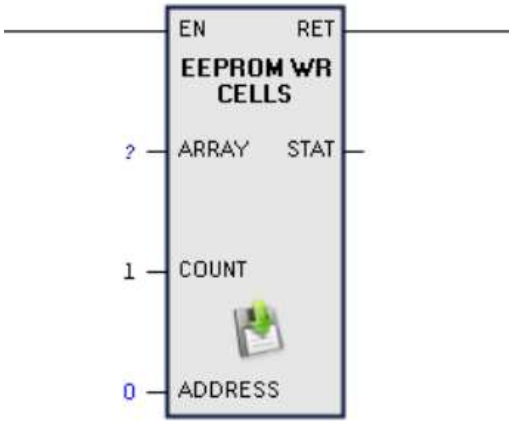
28.3.6 Notas:

- UDP (User Datagram Protocol) es un protocolo estándar de internet para la transmisión de datos sin handshaking.
- Una vez que el buffer de recepción del PLC es leído (total o parcialmente), el PLC estará listo para recibir nuevos bytes. De lo contrario le indicara al transmisor que el PLC no está listo para recibir datos.
- No se recomienda utilizar el argumento **COUNT** = 0, a menos que se utilice un **ARRAY** con la cantidad de elementos suficientes para almacenar todo el buffer de recepción completo.
- Hay instrucciones Pawn que puede utilizar para determinar si hay nuevos datos en el buffer o para limpiarlo. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.

29 Componentes para Memoria EEPROM

Los componentes para la memoria EEPROM le permiten leer o escribir datos en una memoria no-volátil, lo cual es muy útil para registrar valores que no se perderán si el PLC es desenergizado. Aquí puede guardar datos como valores analógicos de sensores, muestras, fechas, horas, parámetros de configuración de su proyecto, etc.

29.1 EepromWriteCells (Escribir Array Int32 en Memoria)



Agrega un componente que permite escribir datos en la memoria EEPROM del PLC.

La memoria EEPROM es una memoria no-volátil que permite retener los datos escritos luego de quitar la energía eléctrica del PLC.

29.1.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**ARRAY**" debe apuntar a un array del tipo **Int32_Array**, que contiene los datos a escribir en la memoria.
- La entrada "**COUNT**" establece la cantidad de elementos que se escribirán del array. No puede superar el tamaño del array.
- En la entrada "**ADDRESS**" se especifica una constante o variable del tipo **Int32**. El valor de este puerto indica a partir de qué dirección de la memoria EEPROM se comenzaran a escribir los elementos del **ARRAY**.

29.1.2 Salidas:

- La salida "**RET**" es verdadera o "**1**" si no existieron errores. Si es "**0**", existió un error en la escritura. Puede encontrar más información del error en el puerto "**STAT**".
- En la salida "**STAT**" se retorna un código de error en una variable tipo **Int32** (uso opcional):
    - El valor retornado es "**0**" si la escritura fue realizada sin errores.
    - El valor es "**-1 a -19**" si hay un error con el bus I2C. Verifique hardware.
    - El valor es "**-70**" si la dirección del **ARRAY** no es válida.
    - El valor es "**-101**" si no hay suficiente tamaño en la memoria para escribir.

- El valor es "**-102**" si la dirección de memoria EEPROM es invalida.
- El valor es "**-103**" si no hay respuesta de la memoria. Verifique que no esté ocupada.

29.1.3 Ejemplo:

Si tenemos un **ARRAY** con los valores "55", "-87866" y "1000" en cada elemento, primero debemos especificarlo en el puerto **ARRAY**. Luego en **COUNT** especificamos el valor de 3 elementos. Podemos guardar en la memoria a partir de la dirección **ADDRESS** = 0.

Cuando ejecutemos el componente, se guardaran los datos.

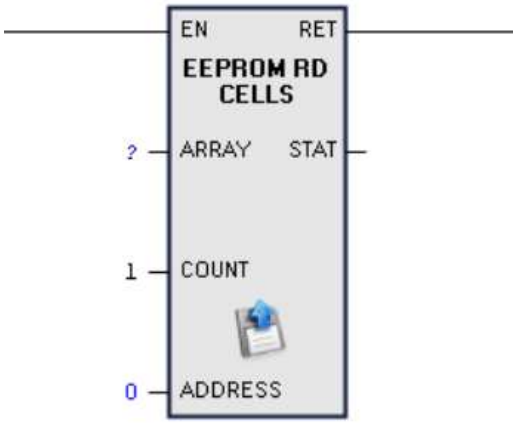
Con el componente **EepromReadCells** puede leer los elementos guardados. Recuerde leer 3 elementos a partir de la dirección 0.

Es conveniente utilizar direcciones de memoria EEPROM múltiplo de 4 para guardar los datos de forma ordenada y aprovechar al máximo el espacio de memoria.

29.1.4 Notas:

- Recuerde que cada elemento del tipo **Int32** requiere de 4 bytes en la memoria EEPROM para poder almacenarse.
- Si sus datos son muy importantes, recuerde verificar el retorno del componente en **STAT** para asegurarse que la escritura fue realizada exitosamente.
- El tiempo mínimo para escribir de 1 a 64 bytes es aproximadamente 5 mS. El tiempo requerido se calcula en múltiplos de 64 bytes. Por ejemplo, si tiene que escribir 65 a 128 bytes, se tardará 10 mS. El error es de 5 mS.
- Hay más instrucciones Pawn que puede utilizar para la memoria EEPROM. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.
- Algunos modelos de PLC no cuentan con memoria EERPOM.

29.2 EepromReadCells (Leer Array Int32 de Memoria)



Agrega un componente que permite leer datos de la memoria EEPROM del PLC.

La memoria EEPROM es una memoria no-volátil que permite retener los datos escritos luego de quitar la energía eléctrica del PLC.

29.2.1 Entradas:

- El componente se ejecuta cuando el valor del flujo de corriente en el puerto de entrada "**EN**" es 1.
- La entrada "**ARRAY**" debe apuntar a un array del tipo **Int32_Array**, que se utilizará para copiar los datos leídos de la memoria EEPROM.
- La entrada "**COUNT**" establece la cantidad de elementos que se leerán de la memoria. No puede superar el tamaño del array. El máximo valor es 32.
- En la entrada "**ADDRESS**" se especifica una constante o variable del tipo **Int32**. El valor de este puerto indica a partir de qué dirección de la memoria EEPROM se comenzaran a leer los datos.

29.2.2 Salidas:

- La salida "**RET**" es verdadera o "**1**" si no existieron errores. Si es "**0**", existió un error en la lectura. Puede encontrar más información del error en el puerto "**STAT**".
- En la salida "**STAT**" se retorna un código de error en una variable tipo **Int32** (uso opcional):
    - El valor retornado es "**0**" si la escritura fue realizada sin errores.
    - El valor es "**-1 a -19**" si hay un error con el bus I2C. Verifique hardware.
    - El valor es "**-70**" si la dirección del **ARRAY** no es válida.
    - El valor es "**-101**" si no hay suficiente tamaño en la memoria para escribir.
    - El valor es "**-102**" si la dirección de memoria EEPROM es invalida.
    - El valor es "**-103**" si no hay respuesta de la memoria. Verifique que no esté ocupada.

29.2.3 Ejemplo:

Si tenemos 3 valores del tipo **Int32** en la memoria EEPROM, por ejemplo: los valores “55”, “-87866” y “1000”, a partir de la dirección 0, debemos leerlos especificando en el componente un array de al menos 3 elementos en el puerto **ARRAY**, el valor 3 en el puerto **COUNT** y la dirección 0 en el puerto **ADDRESS**.

Cuando ejecutemos el componente, se leerán los datos de la memoria EEPROM y se copiaran al puerto **ARRAY**.

Con el componente **EepromWriteCells** puede escribir datos en memoria EEPROM.

Si solo queremos leer el valor “1000” en la memoria, especificamos “**COUNT** = 1” y “**ADDRESS** = 8”. Note que se utilizo la dirección 8, porque suponemos que los 3 elementos estaban almacenados a partir de la dirección 0 y cada uno ocupa 4 bytes.

29.2.4 Notas:

- Recuerde que si tiene elementos del tipo **Int32** almacenados, los mismos estarán separados por 4 bytes c/u.
- Si sus datos son muy importantes, recuerde verificar el retorno del componente en **STAT** para asegurarse que la lectura fue realizada exitosamente.
- Hay más instrucciones Pawn que puede utilizar para la memoria EEPROM. Consulte con el manual de usuario para ver las instrucciones disponibles. Puede agregar código Pawn con los componentes **PawnInsert** o **PawnFunc**.
- Algunos modelos de PLC no cuentan con memoria EERPOM.

30 Abreviaciones y Términos Empleados

- **PLC**: Programable Logic Controller (Controlador Lógico Programable).
- **DAQ**: Data Aquisition (Adquisición de Datos).
- **Modo PLC**: Permite programar el PLC mediante lenguaje Ladder o Pawn.
- **Modo DAQ**: Permite controlar el hardware del PLC a través de una computadora conectada a la interfaz Ethernet, ya sea para adquirir datos o controlar las salidas de la placa.
- **UDP**: User Datagram Protocol. Protocolo orientado a la transmisión/recepción de datos. En el PLC se utiliza para intercambiar datos mediante la interfaz Ethernet.
- **Bootloader**: Programa que corre en el PLC y permite actualizar el firmware.
- **Firmware**: Programa embebido en el PLC y que contiene la lógica de funcionamiento o sistema operativo.
- **IP**: Dirección Internet, conformada por cuatro octetos, por ejemplo 192.168.1.81.
- **RS232**: Define interfaz eléctrica de una comunicación serie.
- **Ethernet**: Red de computadoras, que generalmente se utilizan el protocolo de internet TCP/IP o UDP/IP.
- **Evento**: Circunstancia asíncrona que genera un aviso, el PLC puede atender el aviso ejecutando código desde una función pública Pawn o diagrama Ladder.

31 Historial de Revisiones

Tabla: Historia de Revisiones del Documento

Revisión	Cambios	Descripción	Estado
01 22/AGO/2012	1	1. Versión preliminar liberada.	Preliminar

32 Referencias

Ninguna.

33 Información Legal

33.1 Aviso de exención de responsabilidad

General: La información de este documento se da en buena fe, y se considera precisa y confiable. Sin embargo, Slicetex Electronics no da ninguna representación ni garantía, expresa o implícita, en cuanto a la exactitud o integridad de dicha información y no tendrá ninguna responsabilidad por las consecuencias del uso de la información proporcionada.

El derecho a realizar cambios: Slicetex Electronics se reserva el derecho de hacer cambios en la información publicada en este documento, incluyendo, especificaciones y descripciones de los productos, en cualquier momento y sin previo aviso. Este documento anula y sustituye toda la información proporcionada con anterioridad a la publicación de este documento.

Idoneidad para el uso: Los productos de Slicetex Electronics no están diseñados, autorizados o garantizados para su uso en aeronaves, área médica, entorno militar, entorno espacial o equipo de apoyo de vida, ni en las aplicaciones donde el fallo o mal funcionamiento de un producto de Slicetex Electronics pueda resultar en lesiones personales, muerte o daños materiales o ambientales graves. Slicetex Electronics no acepta ninguna responsabilidad por la inclusión y / o el uso de productos de Slicetex Electronics en tales equipos o aplicaciones (mencionados con anterioridad) y por lo tanto dicha inclusión y / o uso es exclusiva responsabilidad del cliente.

Aplicaciones: Las aplicaciones que aquí se describen o por cualquiera de estos productos son para fines ilustrativos. Slicetex Electronics no ofrece representación o garantía de que dichas aplicaciones serán adecuadas para el uso especificado, sin haber realizado más pruebas o modificaciones.

Los valores límites o máximos: Estrés por encima de uno o más valores límites (como se define en los valores absolutos máximos de la norma IEC 60134) puede causar daño permanente al dispositivo. Los valores límite son calificaciones de estrés solamente y el funcionamiento del dispositivo en esta o cualquier otra condición por encima de las indicadas en las secciones de Características de este documento, no está previsto ni garantizado. La exposición a los valores limitantes por períodos prolongados puede afectar la fiabilidad del dispositivo.

Documento: Prohibida la modificación de este documento en cualquier medio electrónico o impreso, sin autorización previa de Slicetex Electronics por escrito.

34 Información de Contacto

Para mayor información, visítenos en www.slicetex.com

Para información técnica, envíe un mail a: devel@slicetex.com

Para información general, envíe un mail a: info@slicetex.com

Para ventas, envíe un mail a: ventas@slicetex.com

Slicetex Electronics
Córdoba, Argentina

© Slicetex Electronics, todos los derechos reservados.

35 Contenido

1 DESCRIPCIÓN GENERAL 1

2 LECTURAS RECOMENDADAS 2

3 ALCANCE DEL DOCUMENTO 2

3.1 ESTRUCTURA DEL DOCUMENTO 2

4 SELECCIÓN DE COMPONENTES LADDER 3

4.1 VER DESCRIPCION DEL COMPONENTE 4
4.2 ACCESO A LAS PROPIEDADES DEL COMPONENTE 6

5 COMPONENTES BÁSICOS 8

5.1 SWITCHNO (CONTACTO NORMAL ABIERTO) 8
5.1.1 FUNCIONAMIENTO: 8
5.1.2 ACCESO INMEDIATO: 8
5.1.3 TIPS: 8
5.2 SWITCHNC (CONTACTO NORMAL CERRADO) 9
5.2.1 FUNCIONAMIENTO: 9
5.2.2 ACCESO INMEDIATO: 9
5.2.3 TIPS: 9
5.3 SWITCHPE (DETECTOR DE FLANCO POSITIVO) 10
5.3.1 FUNCIONAMIENTO: 10
5.3.2 ACCESO INMEDIATO: 10
5.3.3 NOTAS IMPORTANTES: 10
5.4 SWITCHNE (DETECTOR DE FLANCO NEGATIVO) 11
5.4.1 FUNCIONAMIENTO: 11
5.4.2 ACCESO INMEDIATO: 11
5.4.3 NOTAS IMPORTANTES: 11
5.5 COILNO (BOBINA NORMAL ABIERTA) 12
5.5.1 FUNCIONAMIENTO: 12
5.5.2 ACCESO INMEDIATO: 12
5.6 COILNC (BOBINA NORMAL CERRADA) 13
5.6.1 FUNCIONAMIENTO: 13
5.6.2 ACCESO INMEDIATO: 13
5.7 COILSET (BOBINA SET) 14
5.7.1 ACCESO INMEDIATO: 14
5.7.2 NOTAS: 14
5.8 COILRESET (BOBINA RESET) 15
5.8.1 NOTAS: 15
5.8.2 ACCESO INMEDIATO: 15

5.9 CoilToggle (Bobina Toggle) 16
5.9.1 Funcionamiento: 16
5.9.2 Acceso Inmediato: 16
5.10 Not (Inversor) 17
5.10.1 Funcionamiento: 17
5.10.2 Notas: 17

6 COMPONENTES DEL SISTEMA 18

6.1 DinDelay (Retardo de Lectura de Entradas Discretas) 18
6.1.1 Entradas: 18
6.1.2 Salidas: 18
6.1.3 Atención: 18
6.1.4 Notas: 18

7 COMPONENTES PARA COMENTARIOS 19

7.1 CommentShort (Comentario Corto) 19
7.1.1 Nota: 19
7.2 ComentLong (Comentario Largo) 20
7.2.1 Nota: 20

8 COMPONENTES LEDS 21

8.1 LedDebug (Led para Depuración) 21
8.1.1 Funcionamiento: 21
8.1.2 Notas: 21

9 COMPONENTES PARA OPERACIONES DE VARIABLES 22

9.1 Int32Copy (Copia/Asignación de variable Int32) 22
9.1.1 Funcionamiento: 22
9.2 Int32Dec (Decrementar Variable Int32) 23
9.2.1 Funcionamiento: 23
9.3 Int32Inc (Incrementar Variable Int32) 24
9.3.1 Funcionamiento: 24

10 COMPONENTES DE COMPARACIÓN 25

10.1 Int32Comp (Comparación de Variables Int32) 25
10.1.1 Funcionamiento: 25
10.1.2 TIPS: 25

11 COMPONENTES PARA OPERACIONES ARITMÉTICAS 26

11.1 INT32ARITHMETIC (OPERACIONES ARITMÉTICAS PARA VARIABLES INT32) 26
11.1.1 FUNCIONAMIENTO: 26
11.1.2 TIPS: 27
11.1.3 NOTAS: 27

12 COMPONENTES PARA OPERACIONES LÓGICAS 28

12.1 INT32LOGIC (OPERACIONES LÓGICAS PARA VARIABLES INT32) 28
12.1.1 FUNCIONAMIENTO: 28
12.1.2 TIPS: 29
12.1.3 NOTAS: 29
12.1.4 EJEMPLOS: 29
12.2 INT32SHIFT (OPERACIONES DE DESPLAZAMIENTO PARA VARIABLES INT32) 30
12.2.1 FUNCIONAMIENTO: 30
12.2.2 TIPS: 30
12.2.3 NOTAS: 30
12.2.4 EJEMPLOS: 31
12.3 INT32BITOP (OPERACIONES DE BITS EN VARIABLES INT32) 32
12.3.1 FUNCIONAMIENTO: 32
12.3.2 TIPS: 32
12.3.3 NOTAS: 32
12.3.4 EJEMPLOS: 33
12.4 INT32NBITOP (OPERACIONES DE N-BIT EN VARIABLES INT32) 34
12.4.1 FUNCIONAMIENTO: 34
12.4.2 TIPS: 34
12.4.3 NOTAS: 35
12.4.4 EJEMPLOS: 35

13 COMPONENTES LOOPS 36

13.1 FORLOOP (CICLO O LAZO FOR) 36
13.1.1 FUNCIONAMIENTO: 36
13.1.2 NOTAS: 36

14 COMPONENTES PARA DISPLAY LCD (TEXTO) 37

14.1 LCDTEXTCLEAR (LIMPIAR TEXTO DEL DISPLAY) 37
14.1.1 NOTAS: 37
14.2 LCDTEXTPRINT (IMPRIMIR TEXTO EN DISPLAY) 38
14.2.1 FUNCIONAMIENTO: 38
14.2.2 CONSEJOS: 38
14.3 LCDTEXTPRINTF (IMPRIMIR TEXTO CON FORMATO EN DISPLAY) 39
14.3.1 ENTRADAS: 39

14.3.2 SALIDAS: 39
14.3.3 FORMATO: 40
14.3.4 CÓDIGOS DE FORMATOS DISPONIBLES: 40
14.3.5 ALINEACIÓN Y RELLENO: 41
14.3.6 PRECISIÓN: 41
14.3.7 NOTAS / CONSEJOS: 41

15 COMPONENTES NETHMI (NETWORK HUNMAN MACHINE INTERFACE) 43

15.1 NETHMISETADDRESS (ESTABLECER DIRECCIÓN IP DEL PANEL HMI) 43
15.1.1 FUNCIONAMIENTO: 43
15.1.2 IMPORTANTE: 43
15.1.3 NOTAS: 44
15.1.4 CONSEJOS: 44
15.2 NETHMILCDPRINTF (IMPRIMIR TEXTO CON FORMATO EN DISPLAY) 45
15.2.1 ENTRADAS: 45
15.2.2 SALIDAS: 45
15.2.3 FORMATO: 46
15.2.4 CÓDIGOS DE FORMATOS DISPONIBLES: 46
15.2.5 ALINEACIÓN Y RELLENO: 47
15.2.6 PRECISIÓN: 47
15.2.7 NOTAS / CONSEJOS: 47
15.3 NETHMICLEAR (LIMPIAR DISPLAY) 49
15.3.1 NOTAS / CONSEJOS: 49
15.4 NETHMISETEVENT (ACTIVAR EVENTO NETHMI) 50
15.4.1 ENTRADAS: 50
15.4.2 SALIDAS: 50
15.4.3 NOTAS: 50
15.4.4 CONSEJOS: 50
15.5 NETHMICLREVENT (DESACTIVAR EVENTO NETHMI) 51
15.6 NETHMIDATAPROCESS (PROCESAR DATOS NETHMI) 52
15.6.1 ENTRADAS: 52
15.6.2 SALIDAS: 52
15.6.3 NOTAS / CONSEJOS: 53
15.7 NETHMICMDCHECK (COMPROBAR COMANDO NETHMI) 54
15.7.1 ENTRADAS: 54
15.7.2 SALIDAS: 54
15.7.3 EJEMPLOS: 55
15.7.4 NOTAS / CONSEJOS: 55

16 COMPONENTES TEMPORIZADORES LADDER 56

16.1 TIMERTON (TEMPORIZADOR CON RETARDO A LA CONEXIÓN) 56
16.1.1 FUNCIONAMIENTO GENERAL: 56
16.1.2 ENTRADAS: 57
16.1.3 SALIDAS: 57
16.1.4 NOTAS: 57
16.1.5 CONSEJOS: 57

16.2 TIMERTOFF (TEMPORIZADOR CON RETARDO A LA DESCONEXIÓN) 58
16.2.1 FUNCIONAMIENTO GENERAL: 58
16.2.2 ENTRADAS: 58
16.2.3 SALIDAS: 59
16.2.4 NOTAS: 59
16.2.5 CONSEJOS: 59
16.3 TIMERTDELAY (TEMPORIZADOR DE RETARDOS) 60
16.3.1 FUNCIONAMIENTO GENERAL: 60
16.3.2 ENTRADAS: 61
16.3.3 SALIDAS: 61
16.3.4 IMPORTANTE: 61
16.3.5 NOTAS: 61
16.3.6 CONSEJOS: 61
16.4 TIMERTP (TEMPORIZADOR POR IMPULSOS) 62
16.4.1 FUNCIONAMIENTO GENERAL: 62
16.4.2 ENTRADAS: 62
16.4.3 SALIDAS: 63
16.4.4 NOTAS: 63
16.4.5 CONSEJOS: 63
16.5 TIMERGETET (OBTENER ELAPSED TIME) 64
16.5.1 ENTRADAS: 64
16.5.2 SALIDAS: 64
16.5.3 NOTAS: 64
16.6 TIMERRESET (RESETEAR TIMER) 65
16.6.1 ENTRADAS: 65
16.6.2 NOTAS: 65
16.7 TIMERISRUNNING (COMPROBAR SI TEMPORIZADOR ESTA CORRIENDO) 66
16.7.1 ENTRADAS: 66
16.7.2 SALIDAS: 66
16.7.3 IMPORTANTE: 66
16.7.4 NOTAS: 66
16.8 TIMERISTIMEOUT (COMPROBAR SI TEMPORIZADOR HA FINALIZADO) 67
16.8.1 ENTRADAS: 67
16.8.2 SALIDAS: 67
16.8.3 IMPORTANTE: 67
16.8.4 NOTAS: 67

17 COMPONENTES DE TEMPORIZADORES POR EVENTO 68

17.1 ONTIMER1SETEVENT (ACTIVAR EVENTO ONTIMER1) 68
17.1.1 ENTRADAS: 68
17.1.2 SALIDAS: 68
17.1.3 IMPORTANTE: 68
17.1.4 NOTAS: 68
17.2 ONTIMER1CLREVENT (DESACTIVAR EVENTO ONTIMER1) 69
17.2.1 IMPORTANTE: 69
17.3 ONTIMER2SETEVENT (ACTIVAR EVENTO ONTIMER2) 70
17.3.1 ENTRADAS: 70
17.3.2 SALIDAS: 70

17.3.3 NOTAS: 70
17.4 ONTIMER2CLREVENT (DESACTIVAR EVENTO ONTIMER2) 71

18 COMPONENTES PARA TEMPORIZADORES TIMEOUT 72

18.1 TIMEOUTSETEVENT (ACTIVA UN TIMEOUT PARA EL EVENTO ONTIMEOUT) 72
18.1.1 ENTRADAS: 72
18.1.2 SALIDAS: 72
18.1.3 NOTAS: 72
18.2 TIMEOUTRELOAD (RECARGAR UN TEMPORIZADOR TIMEOUT) 73
18.2.1 ENTRADAS: 73
18.2.2 SALIDAS: 73
18.2.3 NOTAS: 73
18.3 TIMEOUTCHECK (COMPROBAR TIMEOUT DEL TEMPORIZADOR) 74
18.3.1 ENTRADAS: 74
18.3.2 SALIDAS: 74
18.3.3 NOTAS: 74
18.4 TIMEOUTCLREVENT (DESACTIVAR UN TEMPORIZADOR DEL TIPO TIMEOUT) 75
18.4.1 ENTRADAS: 75
18.4.2 SALIDAS: 75
18.4.3 NOTAS: 75

19 COMPONENTES DEL RTC (RELOJ DE TIEMPO REAL) 76

19.1 RTCSETALL (ESTABLECER HORA Y FECHA EN RTC) 76
19.1.1 ENTRADAS: 76
19.1.2 NOTAS / CONSEJOS: 77
19.2 RTCGETDATE (OBTENER FECHA DEL RTC) 78
19.2.1 ENTRADAS: 78
19.2.2 SALIDAS: 78
19.2.3 NOTAS: 78
19.3 RTCGETTIME (OBTENER HORA DE RTC) 79
19.3.1 ENTRADAS: 79
19.3.2 SALIDAS: 79
19.3.3 NOTAS: 79
19.4 RTCGETDOW (OBTENER DÍA DE LA SEMANA DEL RTC) 80
19.4.1 ENTRADAS: 80
19.4.2 SALIDAS: 80
19.4.3 NOTAS: 80
19.5 RTCGETTIMESTAMP (OBTENER TIMESTAMP DEL RTC) 81
19.5.1 ENTRADAS: 81
19.5.2 SALIDAS: 81
19.5.3 INFORMACIÓN: 81
19.5.4 NOTAS: 81
19.6 RTCONSECONDSETEVENT (ACTIVAR EVENTO RTCONSECOND) 82
19.6.1 ENTRADAS: 82
19.6.2 SALIDAS: 82
19.6.3 NOTAS: 82

19.7 RTCONSECONDCLREVENT (DESACTIVAR EVENTO RTCONSECOND) 83
19.8 RTCONALARMSETEVENT (ACTIVAR EVENTO RTCONALARM) 84
19.8.1 ENTRADAS: 84
19.8.2 NOTAS: 85
19.9 RTCONALARMCLREVENT (DESACTIVAR EVENTO RTCONALARM) 86

20 COMPONENTES PARA ALARMAS DE TIEMPO 87

20.1 TIMEALARMSETEVENT (ACTIVAR UN EVENTO DE ALARMA ONTIMEALARMX) 87
20.1.1 ENTRADAS: 88
20.1.2 SALIDAS: 88
20.1.3 MODOS DE ALARMA ADMITIDOS: 88
20.1.4 EJEMPLOS 89
20.1.5 NOTAS 89
20.2 TIMEALARMCLREVENT (DESACTIVAR UN EVENTO DE ALARMA ONTIMEALARMX) 90

21 COMPONENTES PARA CONTADORES LADDER 91

21.1 COUNTERUP (CONTADOR ASCENDENTE CUP) 91
21.1.1 FUNCIONAMIENTO: 91
21.1.2 ENTRADAS: 91
21.1.3 SALIDAS: 91
21.1.4 NOTAS: 92
21.2 COUNTERDOWN (CONTADOR DESCENDENTE CDOWN) 93
21.2.1 FUNCIONAMIENTO: 93
21.2.2 ENTRADAS: 93
21.2.3 SALIDAS: 93
21.2.4 NOTAS: 93
21.3 COUNTERUPDOWN (CONTADOR ASCENDENTE/DESCENDENTE CUD) 94
21.3.1 FUNCIONAMIENTO: 94
21.3.2 ENTRADAS: 94
21.3.3 SALIDAS: 95
21.3.4 NOTAS: 95
21.4 COUNTERGETCV (OBTENER COUNT VALUE DEL CONTADOR) 96
21.4.1 ENTRADAS: 96
21.4.2 SALIDAS: 96
21.4.3 NOTAS: 96

22 COMPONENTES PARA CONTADORES EXTERNOS 97

22.1 EXTCOUNTSETUP (CONFIGURAR CONTADOR EXTERNO) 97
22.1.1 ENTRADAS: 97
22.1.2 SALIDAS: 97
22.1.3 IMPORTANTE: 97
22.1.4 TIPOS DE FLANCOS ADMITIDOS: 98
22.1.5 NOTAS: 98

22.2 EXTCOUNTDISABLE (DESACTIVAR CONTADOR EXTERNO) 99
22.2.1 ENTRADAS: 99
22.2.2 SALIDAS: 99
22.3 EXTCOUNTRESET (RESETEAR CONTADOR EXTERNO) 100
22.3.1 ENTRADAS: 100
22.3.2 SALIDAS: 100
22.4 EXTCOUNTCHANGEEDGE (CAMBIAR FLANCO DE CUENTA DEL CONTADOR EXTERNO) 101
22.4.1 ENTRADAS: 101
22.4.2 SALIDAS: 101
22.4.3 TIPOS DE FLANCOS ADMITIDOS: 101
22.5 EXTCOUNTGETEVENTCNT (OBTENER CUENTA DEL CONTADOR EXTERNO) 102
22.5.1 ENTRADAS: 102
22.5.2 SALIDAS: 102
22.5.3 INFORMACIÓN: 102
22.5.4 NOTAS: 102
22.6 EXTCOUNTGETEVENTPERIOD (OBTENER PERIODO ENTRE EVENTOS) 103
22.6.1 ENTRADAS: 103
22.6.2 SALIDAS: 103
22.6.3 NOTAS: 103
22.6.4 INFORMACIÓN: 104
22.7 EXTCOUNTGETEVENTFREQ (OBTENER FRECUENCIA ENTRE EVENTOS) 105
22.7.1 ENTRADAS: 105
22.7.2 SALIDAS: 105
22.7.3 NOTAS: 105
22.7.4 INFORMACIÓN: 106
22.8 EXTCOUNTGETEVENTRPM (OBTENER RPM ENTRE EVENTOS) 107
22.8.1 ENTRADAS: 107
22.8.2 SALIDAS: 107
22.8.3 IMPORTANTE: 107
22.8.4 NOTAS: 107
22.8.5 INFORMACIÓN: 108
22.9 EXTCOUNTSETMATCHEVENT (ACTIVA EVENTO ONCOUNTXMATCH) 109
22.9.1 ENTRADAS: 109
22.9.2 SALIDAS: 109
22.9.3 NOTAS: 109
22.9.4 EJEMPLO: 110
22.10 EXTCOUNTCLRMATCHEVENT (DESACTIVAR EVENTO ONCOUNTXMATCH) 111
22.10.1 ENTRADAS: 111
22.10.2 SALIDAS: 111
22.10.3 NOTAS: 111

23 COMPONENTES PARA FUNCIONES LADDER 112

23.1 FUNCCALL (LLAMAR A FUNCIÓN LADDER) 112
23.1.1 ENTRADAS: 112
23.1.2 SALIDAS: 112
23.1.3 NOTAS: 112
23.2 FUNCRETURN (RETORNAR DE FUNCION LADDER) 114
23.2.1 ENTRADAS: 114

23.2.2 IMPORTANTE: 114
23.2.3 INFORMACIÓN: 114
23.2.4 NOTAS: 114

24 COMPONENTES PAWN 115

24.1 PAWNINSERT (INSERTAR CÓDIGO PAWN) 115
24.1.1 INFORMACIÓN: 115
24.1.2 EJEMPLO: 115
24.1.3 CONSEJOS: 116
24.2 PAWNFUNC (LLAMAR A FUNCIÓN PAWN) 117
24.2.1 ENTRADAS: 117
24.2.2 SALIDAS: 117
24.2.3 INFORMACIÓN: 118
24.2.4 EJEMPLO: 118
24.2.5 CONSEJOS: 118

25 COMPONENTES PARA SALIDAS PWM 119

25.1 PWMFREQ (ESTABLECER FRECUENCIA DEL PWM) 119
25.1.1 ENTRADAS: 119
25.1.2 SALIDAS: 119
25.1.3 ATENCIÓN: 119
25.1.4 FORMA DE ONDA DEL PWM: 120
25.1.5 NOTAS: 120
25.2 PWMDC (ESTABLECER DUTY-CYCLE EN SALIDA PWM) 121
25.2.1 ENTRADAS: 121
25.2.2 SALIDAS: 121
25.2.3 EJEMPLOS: 121
25.2.4 FORMA DE ONDA DEL PWM: 122
25.2.5 NOTAS: 122

26 COMPONENTES PARA ENTRADAS ANALÓGICAS 123

26.1 VINREAD (LEER ENTRADA) 123
26.1.1 ENTRADAS: 123
26.1.2 SALIDAS: 123
26.1.3 RESOLUCIÓN: 123
26.1.4 CONSEJOS: 124
26.1.5 NOTAS: 124
26.2 VIN12FILTER (FILTRO PARA ENTRADAS VIN1 Y VIN2) 125
26.2.1 ENTRADAS: 125
26.2.2 SALIDAS: 125
26.2.3 IMPORTANTE: 126
26.2.4 NOTAS: 126
26.2.5 CONSEJOS: 126
26.2.6 EJEMPLO: 126

26.3 VIN34FILTER (FILTRO PARA ENTRADAS VIN3 Y VIN4) 127
26.3.1 ENTRADAS: 127
26.3.2 SALIDAS: 127
26.3.3 IMPORTANTE: 128
26.3.4 NOTAS: 128
26.3.5 CONSEJOS: 128
26.3.6 EJEMPLO: 128
26.4 VIN12SAMPLERON (ACTIVAR SAMPLER 1 / 2) 129
26.4.1 ENTRADAS: 129
26.4.2 SALIDAS: 130
26.4.3 IMPORTANTE: 130
26.4.4 NOTAS: 130
26.4.5 CONSEJOS: 130
26.5 VIN12SAMPLEROFF (DESACTIVAR SAMPLER 1 / 2) 131
26.5.1 ENTRADAS: 131
26.5.2 SALIDAS: 131
26.5.3 CONSEJOS: 131
26.6 VIN34SAMPLERON (ACTIVAR SAMPLER 3 / 4) 132
26.6.1 ENTRADAS: 132
26.6.2 SALIDAS: 133
26.6.3 IMPORTANTE: 133
26.6.4 NOTAS: 133
26.6.5 CONSEJOS: 133
26.7 VIN34SAMPLEROFF (DESACTIVAR SAMPLER 3 / 4) 134
26.7.1 ENTRADAS: 134
26.7.2 SALIDAS: 134
26.7.3 CONSEJOS: 134
26.8 VINSAMPLERGETBUF (OBTENER BUFFER DE MUESTRAS DE UN SAMPLER) 135
26.8.1 ENTRADAS: 135
26.8.2 SALIDAS: 135
26.8.3 NOTAS: 136
26.8.4 CONSEJOS: 136

27 COMPONENTES PARA SALIDAS ANALÓGICAS 137

27.1 VOUTWRITE (ESCRIBIR EN SALIDA ANALÓGICA) 137
27.1.1 ENTRADAS: 137
27.1.2 SALIDAS: 137
27.1.3 RESOLUCIÓN: 137
27.1.4 NOTAS: 137
27.2 VOUTSINEGENSETUP (CONFIGURAR GENERADOR SENOIDAL) 138
27.2.1 ENTRADAS: 138
27.2.2 SALIDAS: 138
27.2.3 NOTAS: 138
27.2.4 APLICACIONES: 138
27.3 VOUTSINEGENFREQ (MODIFICAR FRECUENCIA DEL GENERADOR SENOIDAL) 139
27.3.1 ENTRADAS: 139
27.3.2 SALIDAS: 139
27.3.3 IMPORTANTE: 139

27.3.4 NOTAS: 139
27.3.5 APLICACIONES: 139

28 COMPONENTES PARA CONEXIÓN ETHERNET / UDP 140

28.1 UDPSEND (ENVIAR DATOS VÍA UDP) 140
28.1.1 ENTRADAS: 140
28.1.2 SALIDAS: 141
28.1.3 EJEMPLO: 141
28.1.4 DATOS EMPAQUETADOS (PACKED) 141
28.1.5 NOTAS: 141
28.2 UDPONRXSETEVENT (ACTIVAR EVENTO ONUDPRX) 142
28.2.1 ENTRADAS: 142
28.2.2 SALIDAS: 142
28.2.3 NOTAS: 142
28.2.4 CONSEJOS: 142
28.3 UDPRXDATAREAD (OBTENER DATOS RECIBIDOS POR UDP) 143
28.3.1 ENTRADAS: 143
28.3.2 SALIDAS: 143
28.3.3 EJEMPLO: 144
28.3.4 DATOS EMPAQUETADOS (PACKED) 144
28.3.5 CONSEJOS: 144
28.3.6 NOTAS: 144

29 COMPONENTES PARA MEMORIA EEPROM 145

29.1 EEPROMWRITECELLS (ESCRIBIR ARRAY INT32 EN MEMORIA) 145
29.1.1 ENTRADAS: 145
29.1.2 SALIDAS: 145
29.1.3 EJEMPLO: 146
29.1.4 NOTAS: 146
29.2 EEPROMREADCELLS (LEER ARRAY INT32 DE MEMORIA) 147
29.2.1 ENTRADAS: 147
29.2.2 SALIDAS: 147
29.2.3 EJEMPLO: 148
29.2.4 NOTAS: 148

30 ABREVIACIONES Y TÉRMINOS EMPLEADOS 149

31 HISTORIAL DE REVISIONES 150

32 REFERENCIAS 151

33 INFORMACIÓN LEGAL 151

33.1 AVISO DE EXENCIÓN DE RESPONSABILIDAD 151

34 INFORMACIÓN DE CONTACTO 152

35 CONTENIDO 153

Copyright Slicetex Electronics 2012

www.slicetex.com