Actuador todo/nada

CT416410

Manual de programación

ingenium

www.besknx.com

Índice

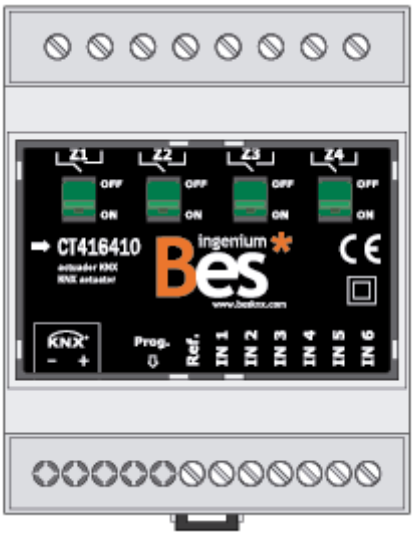
1 Descripción general

El modelo de Bes ref. CT416410 es un actuador compuesto por 4 salidas de relé libres de potencial y 6 entradas de bajo voltaje (SELV) con una referencia común interna, para la conexión de pulsadores convencionales o interruptores.

Sus 4 salidas permiten el control de 4 circuitos eléctricos On / Off o de 2 persianas (2 salidas para cada motor de persiana: fase de subida y fase de bajada). Debido a su elevada capacidad de corte, este dispositivo está también recomendado para cargas capacitivas, enchufes y aparatos eléctricos de control. Las entradas pueden trabajar en distintos modos, permitiendo el control de salidas binarias, reguladores o persianas de forma independiente o simultáneamente. Es posible configurar la respuesta del dispositivo cuando hay un flanco de subida, un flanco de bajada o una pulsación corta o larga dependiendo del modo de trabajo.

Incorpora una avanzada unidad aritmético lógica (ALU) que permite el uso de complejas operaciones lógicas, programación de temporizadores, contadores, etc. utilizando resultados internos de resultados de operaciones u otras variables externas.

La capacidad de corte de los relés es de 30A @ 230Vac (salida libre de potencial). En caso de ser necesario, inserte un contactor para controlar circuitos de elevada potencia.



Características generales:

- 6 entradas digitales de bajo voltaje (SELV).
- 4 salidas de relé libres de potencial con una capacidad de corte de 30A @ 230Vac.
- Cada salida puede trabajar independientemente o simultáneamente en distintos modos (binario, persianas, fan-coils…).
- Entradas programables para trabajar con interruptores o pulsadores.
- Intuitiva unidad aritmética lógica (ALU) con temporizadores, contadores y la posibilidad de implementar complejas operaciones aritmético lógicas.
- Ejecución de avanzados scripts que pueden ser implementados por el programador.

2 Información técnica

Alimentación	29 Vdc del bus KNX
Consumo de corriente	9 mA del bus KNX
Montaje	Carril DIN
Dimensiones	4 módulos DIN
Conexiones	Conexión a bus KNX Regleta atornillable para entradas y salidas
Entradas	6 entradas de bajo voltaje (SELV) con referencia común interna
Corriente de activación de las entradas	Mínimo: 15 mA
Longitud del cable de las entradas	30 metros máximo (desde el mecanismo a la entrada)
Salidas	Salidas a relé libres de potencial
Capacidad de corte de las salidas	30A @ 230 Vac
Rango de temperatura ambiente	Funcionamiento: -10 ºC / 55ºC Almacenamiento: -30 ºC / 60ºC Transporte: -30 ºC / 60 ºC
Regulación	De acuerdo a las directivas de compatibilidad electromagnética y bajo voltaje: EN 50090-2-2 / UNE-EN 61000-6-3:2007 / UNE-EN 61000-6-1:2007 / UNE-EN 61010-1

3 Programación

3.1 Información del catálogo ETS

Catálogo: Ingenium (fabricante) / Actuadores (nombre).

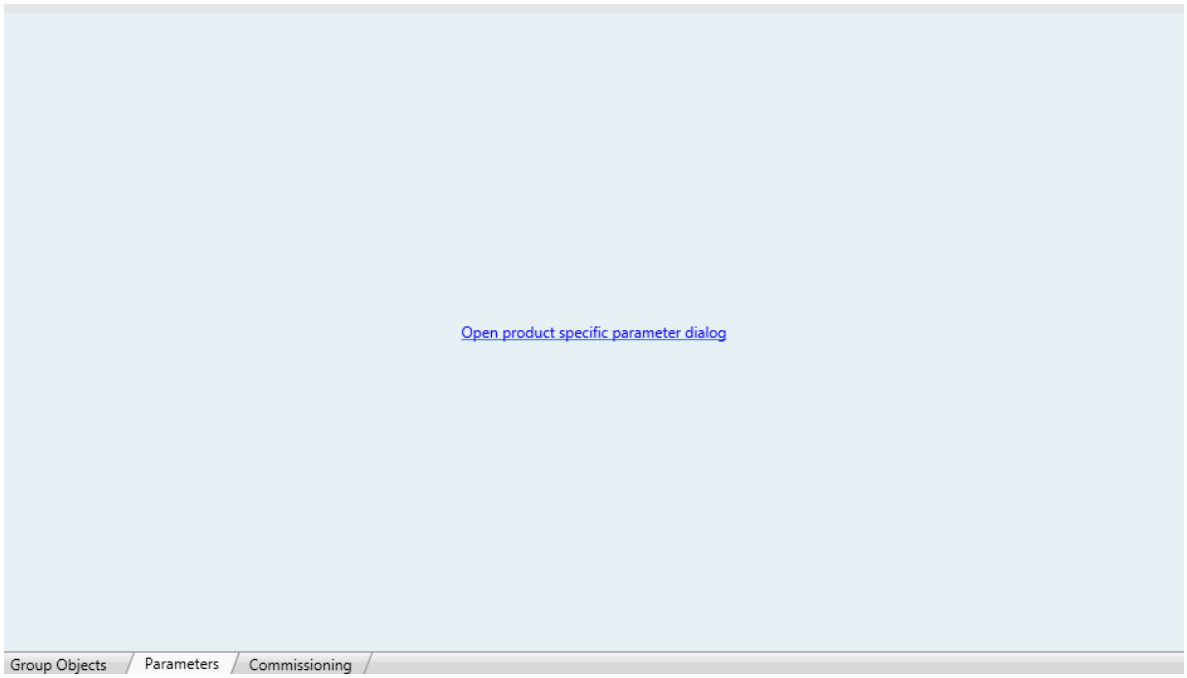
Versión del catálogo: v1.4

Número máximo de objetos de comunicación: 256

Número máximo de asignaciones: 256

Versión mínima de ETS: 4.1.8

Los parámetros del dispositivo son configurados mediante un plug-in, para lo que se deberá abrir el diálogo de parámetros específicos de producto desde el apartado de “parámetros”



3.2 Asignación de dirección individual

Este actuador dispone de un botón de programación situado en la parte frontal del dispositivo para establecer la dirección individual KNX.

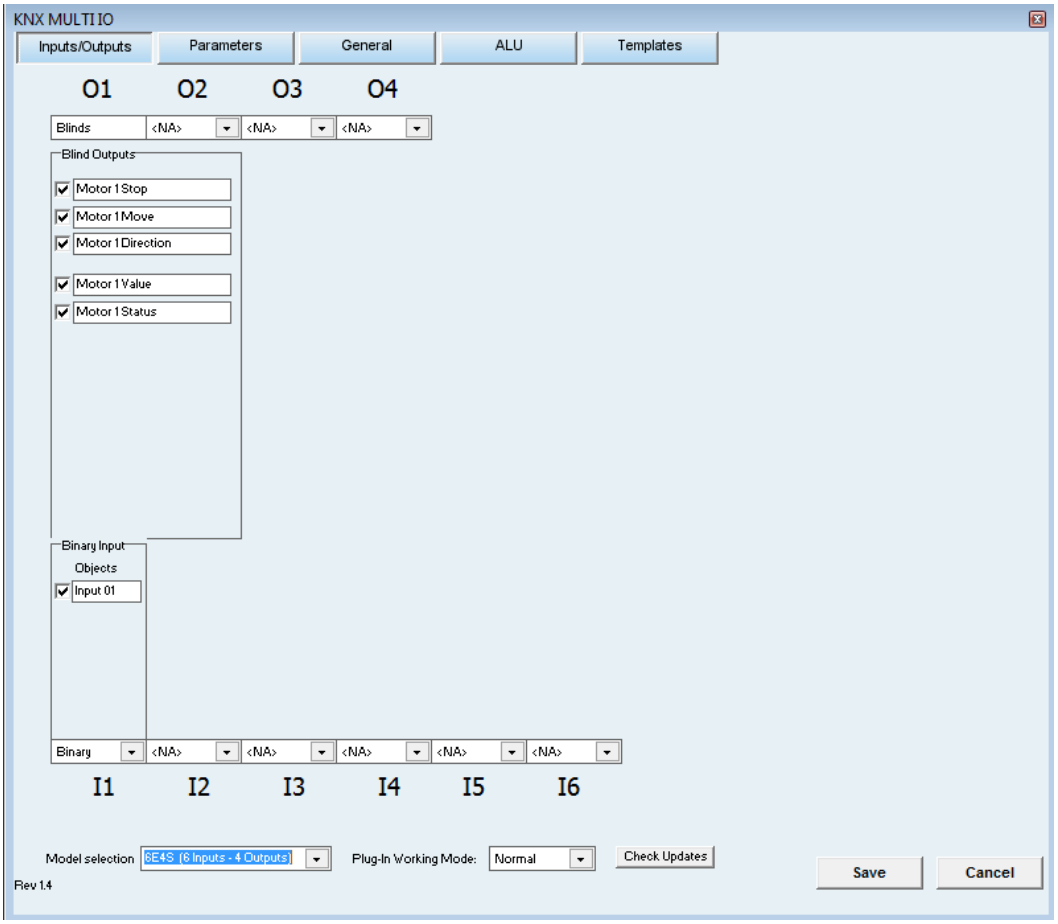
Un LED rojo próximo al botón de programación se ilumina cuando se pulsa el botón manualmente o cuando el dispositivo es forzado de forma remota a modo de programación.

El LED se apaga automáticamente si el ETS ha asignado una dirección individual correctamente o si el botón de programación es presionado manualmente de nuevo.

3.3 Tipo de dispositivo

Los parámetros del dispositivo se configuran mediante un diálogo específico de parámetros.

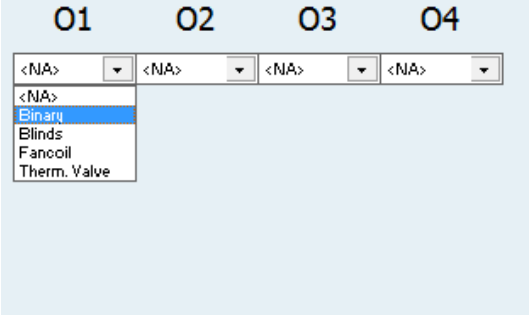
Existen diversas pestañas para configurar los distintos parámetros dependiendo del tipo de dispositivo seleccionado. En este caso, el dispositivo seleccionado debe ser de tipo “6 entradas - 4 salidas”.



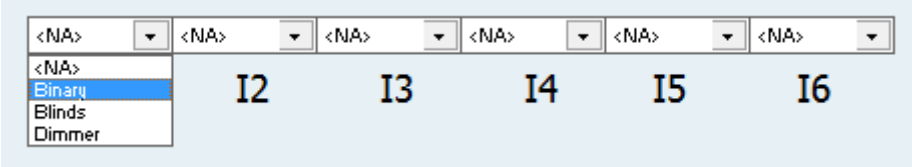
Utilice el selector en la parte inferior izquierda de la ventana principal para seleccionar el tipo de dispositivo a programar.

Después, aparecerán cierto número de entradas y salidas, en función del dispositivo seleccionado. Cada una de esas entradas y salidas puede ser configurada para trabajar en distintos modos, independientemente o de forma simultánea.

Las salidas pueden ser programadas en modo binario, persiana, fan-coil o termoválvula.

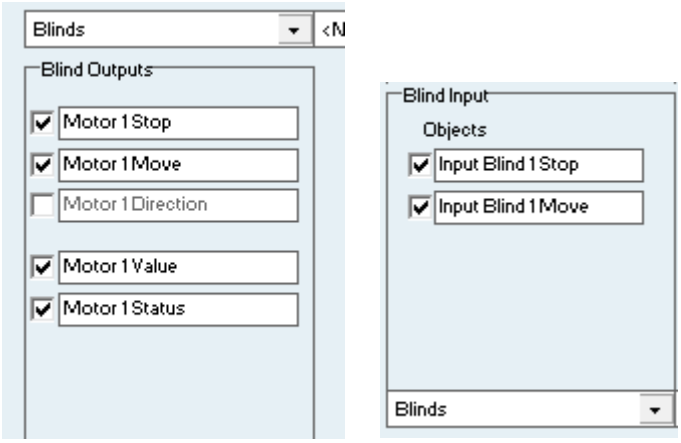


En el caso de las entradas, estas pueden ser programadas en modo binario, persiana o regulador.



Dependiendo del tipo de salida seleccionado, se puede ocupar más de un slot. Por ejemplo, cuando se selecciona el modo persiana, se reservan dos salidas (salida impar para la fase de subida, y salida par para la fase de bajada).

Una vez seleccionado los tipos de entradas y salidas, los objetos de comunicación asociados aparecerán debajo. El programador puede seleccionar si quiere utilizar o no alguno de los objetos de comunicación seleccionándolo, y modificar el nombre que aparecerá en el ETS para una identificación más sencilla del mismo.



Los objetos de comunicación por defecto serán explicados a continuación.

3.4 Objetos de salida

3.4.1 Tabla de salidas binarias

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
0	Output 1 | Output 1 switching	1 bit	1.001	●		●		●
1	Output 1 status | Output 1 switching feedback	1 bit	1.001	●	●		●	
2	Output 2 | Output 2 switching	1 bit	1.001	●		●		●
3	Output 2 status | Output 2 switching feedback	1 bit	1.001	●	●		●	
4	Output 3 | Output 3 switching	1 bit	1.001	●		●		●
5	Output 3 status | Output 3 switching feedback	1 bit	1.001	●	●		●	
6	Output 4 | Output 4 switching	1 bit	1.001	●		●		●
7	Output 4 status | Output 4 switching feedback	1 bit	1.001	●	●		●	
8	Output 1 bit script | Script recall	1 bit	1.001	●		●	●	●
9	Output 2 bit script | Script recall	1 bit	1.001	●		●	●	●
10	Output 3 bit script | Script recall	1 bit	1.001	●		●	●	●
11	Output 4 bit script | Script recall	1 bit	1.001	●		●	●	●
128	Output 1 1-byte script | Script recall	1 byte	5.010	●		●	●	●
129	Output 2 1-byte script | Script recall	1 byte	5.010	●		●	●	●
130	Output 3 1-byte script | Script recall	1 byte	5.010	●		●	●	●
131	Output 4 1-byte script | Script recall	1 byte	5.010	●		●	●	●
239	Output 1 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
240	Output 2 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
241	Output 3 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
242	Output 4 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●

3.4.2 Descripción de salidas binarias

Nombre	Object X: Output X | Output X switching
Función	Objeto de comunicación de 1 bit para encender y apagar una salida.
Descripción	Cuando se recibe un "1" a través de este objeto la salida es activada. Cuando se recibe un "0" la salida es desactivada. Este es el funcionamiento del modo "normalmente abierto" (ver la sección *3.6.1 Parámetros de salidas binarias* en la página 16). El modo "normalmente cerrado" es el opuesto. Por defecto, el estado de una salida se memoriza cuando hay un fallo en la tensión de alimentación. Si es necesario definir un estado específico, utilice un "script de inicio" (ver la sección *3.12 Modo programador* en la página 36).
Nombre	**Object X: Output X | Output X switching feedback**
Función	Objeto de comunicación de 1 bit para notificación sobre el estado de la salida.
Descripción	Cuando la salida está apagada y recibe un telegrama de encendido, envía un "1" a través de este objeto. Cuando la salida está encendida y recibe un telegrama de apagado, envía un "0" a través de este objeto.
Nombre	**Object X: Output X | Output X script recall**
Función	Objeto de comunicación de 1-bit/1-byte/2-bytes para la ejecución de scripts.
Función	Programación avanzada de scripts (ver la sección *3.12 Modo programador* en la página 36).

3.4.3 Tabla de salidas de tipo persiana

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
0	Motor 1 move | Blind 1 move 0=up;1=down	1 bit	1.001	●		●		●
1	Motor 1 stop | Blind 1 stop 0=1=stop	1 bit	1.001	●		●		●
2	Motor 1 direction | current direction feedback	1 bit	1.001	●	●		●	
128	Motor 1 value | Blind 1 direct positioning	1 byte	5.001	●		●		●
129	Motor 1 status | Blind 1 position feedback	1 byte	5.001	●	●		●	
3	Motor 2 move | Blind 2 move 0=up;1=down	1 bit	1.001	●		●		●
4	Motor 2 stop | Blind 2 stop 0=1=stop	1 bit	1.001	●		●		●
5	Motor 2 direction | current direction feedback	1 bit	1.001	●	●		●	
130	Motor 2 value | Blind 2 direct positioning	1 byte	5.001	●		●		●
131	Motor 2 status | Blind 2 position feedback	1 byte	5.001	●	●		●	

6	Motor 1 bit script | Script recall	1 bit	1.001	●		●	●	●
7	Motor 2 bit script | Script recall	1 bit	1.001	●		●	●	●
132	Motor 1 1-byte script | Script recall	1 byte	5.010	●		●	●	●
133	Motor 2 1-byte script | Script recall	1 byte	5.010	●		●	●	●
239	Motor 1 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
240	Motor 2 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●

3.4.4 Descripción de salidas de tipo persiana

Nombre	**Object X: Motor X move | Blind 1 move (0=up;1=down)**
Función	Objeto de comunicación de 1 bit para mover arriba o abajo la persiana.
Descripción	Cuando se recibe un “1” a través de este objeto el motor de la persiana se mueve hacia arriba. Cuando se recibe un “0”, el motor de la persiana se mueve hacia abajo. Las salidas impares (Z1 y Z3) se deben conectar a las fases de subida, mientras que las pares (Z2 y Z4) se deben conectar a las fases de bajada. Este orden no puede ser alterado.
Nombre	**Object X: Motor X stop | Blind 1 stop (0=1=stop)**
Función	Objeto de comunicación de 1 bit para parar el movimiento de la persiana.
Descripción	Cuando se recibe cualquier valor a través de este objeto de comunicación el motor de la persiana deja de moverse.
Nombre	**Object X: Motor X direction | current direction feedback**
Función	Objeto de comunicación de 1 bit para notificación de la dirección de movimiento actual.
Descripción	Cuando el motor de la persiana empieza a moverse hacia arriba se envía un telegrama de valor “0” a través de este objeto. Cuando el motor de la persiana empieza a moverse hacia abajo se envía un telegrama de valor “1” a través de este objeto. Si se lee este objeto mientras el motor no se está moviendo responde con un telegrama de valor “0”.
Nombre	**Object X: Motor X value | Blind 1 direct positioning**
Función	Objeto de comunicación de 1 byte para posicionar la persiana a un valor directamente.
Descripción	Cuando se envía un valor a través de este objeto la persiana se mueve hasta la posición recibida, siendo 0 = completamente cerrada y 255 = completamente abierta.

Nombre	Objeto X: Motor X status | Blind X position feedback
Función	Objeto de comunicación de 1 byte para notificación de la posición de la persiana.
Descripción	Cuando el motor de la persiana se para envía una notificación a través de este objeto con la posición actual de la persiana, siendo 0 = completamente cerrada y 255 = completamente abierta. Por defecto, la posición de la persiana se envía cada segundo mientras se está moviendo. Si se desactiva el parámetro "notificación periódica de la persiana", la notificación se envía solamente cuando el motor se para (ver la sección *3.6.2 Parámetros de salida tipo persiana* en la página 17).
Nombre	**Objeto X: Output X | Output X script recall**
Función	Objeto de comunicación de 1-bit/1-byte/2-bytes para la ejecución de scripts.
Descripción	Programación avanzada de scripts (ver la sección *3.12 Modo programador* en la página 36).

3.4.5 Tabla de salidas de tipo fan-coil

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
0	Fan 1 speed 1 | Fan-Coil speed 1 setting	1 bit	1.001	●		●		●
1	Fan 1 speed 1 status | Fan-Coil speed 1 feedback	1 bit	1.001	●	●		●	
2	Fan 1 speed 2 | Fan-Coil speed 2 setting	1 bit	1.001	●		●		●
3	Fan 1 speed 2 status | Fan-Coil speed 2 feedback	1 bit	1.001	●	●		●	
4	Fan 1 speed 3 | Fan-Coil speed 3 setting	1 bit	1.001	●		●		●
5	Fan 1 speed 3 status | Fan-Coil speed 3 feedback	1 bit	1.001	●	●		●	
128	Fan 1 mode | Fan-Coil mode setting	1 byte	5.001	●		●		●
129	Fan 1 mode status | Fan-Coil mode feedback	1 byte	5.001	●	●		●	
6	Fan 1 bit script | Script recall	1 bit	1.001	●		●	●	●
130	Fan 1 byte script | Script recall	1 byte	5.010	●		●	●	●
239	Fan 1 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●

3.4.6 Descripción de salidas de tipo fan-coil

Nombre	Objeto X: Fan 1 speed X | Fan-Coil speed X setting
Función	Objeto de comunicación de 1 bit para cambiar el fan-coil a la velocidad correspondiente.
Descripción	Cuando se recibe un "1" a través de este objeto, el fan-coil cambia a la velocidad correspondiente. Las otras velocidades son desactivadas y se envía un "0" a través de sus objetos de comunicación de notificación. Las velocidades del fan-coil deben ser conectadas a las salidas como sigue: Z1=velocidad 1, Z2=velocidad 2 y Z3=velocidad 3. En caso de que sea necesario cambiar esta configuración, utilice un "fan-coil personalizado" (vea la sección *3.11.5 Custom fan-coil* en la página 34).
Nombre	**Objeto X: Fan 1 speed X status | Fan-Coil speed X feedback**
Función	Objeto de comunicación de 1 bit para notificación de la velocidad actual.
Descripción	Cuando se selecciona una velocidad, el estado se envía a través de este objeto. Se envía un telegrama de valor "1" en el caso de la velocidad seleccionada y un "0" en el resto.
Nombre	**Objeto X: Fan 1 mode | Fan-Coil mode setting**
Función	Objeto de comunicación de 1 byte para seleccionar una velocidad directamente.
Descripción	Cuando se recibe un valor a través de este objeto el control de fan-coil lo compara con el nivel umbral configurado y activa la velocidad correspondiente (ver la sección *3.6.3 Parámetros de salida de tipo* fan-coil en la página 17).
Nombre	**Objeto X: Fan 1 mode status | Fan-Coil mode feedback**
Función	Objeto de comunicación de 1 byte para notificación del estado del fan-coil.
Descripción	Con cada cambio, se envía a través de este objeto la velocidad actual del fan-coil.
Nombre	**Objeto X: Fan X | Fan X script recall**
Función	Objeto de comunicación de 1-bit/1-byte/2-bytes para la ejecución de scripts.
Descripción	Programación avanzada de scripts (ver la sección *3.12 Modo programador* en la página 36).

3.4.7 Tabla de salidas tipo termoválvula

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
128	Valve 1 pwm| Valve 1 pwm control	1 byte	5.010	●	●	●	●	●
0	Valve 1 status | Valve 1 switching feedback	1 bit	1.001	●	●		●	
129	Valve 2 pwm| Valve 2 pwm control	1 byte	5.010	●	●	●	●	●
1	Valve 2 status | Valve 2 switching feedback	1 bit	1.001	●	●		●	
130	Valve 3 pwm| Valve 3 pwm control	1 byte	5.010	●	●	●	●	●
2	Valve 3 status | Valve 3 switching feedback	1 bit	1.001	●	●		●	

131	Valve 4 pwm| Valve 4 pwm control	1 byte	5.010	●	●	●	●	●
3	Valve 4 status | Valve 4 switching feedback	1 bit	1.001	●	●		●	
4	Valve 1 bit script | Script recall	1 bit	1.001	●		●	●	●
5	Valve 2 bit script | Script recall	1 bit	1.001	●		●	●	●
6	Valve 3 bit script | Script recall	1 bit	1.001	●		●	●	●
7	Valve 4 bit script | Script recall	1 bit	1.001	●		●	●	●
132	Valve 1 1-byte script | Script recall	1 byte	5.010	●		●	●	●
133	Valve 2 1-byte script | Script recall	1 byte	5.010	●		●	●	●
134	Valve 3 1-byte script | Script recall	1 byte	5.010	●		●	●	●
135	Valve 4 1-byte script | Script recall	1 byte	5.010	●		●	●	●
239	Valve 1 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
240	Valve 2 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
241	Valve 3 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●
242	Valve 4 2-bytes script | Script recall	2 bytes	7.001	●		●	●	●

● = Configuración por defecto
o = Opcional

3.4.8 Descripción de salidas de tipo termoválvula

Nombre	Objeto X: Valve X pwm| Valve X pwm control
Función	Objeto de comunicación de 1 byte para establecer el ciclo de trabajo de la señal pwm de la termoválvula.
Descripción	El ciclo de trabajo de la señal pwm que controla la salida de la termoválvula es establecido mediante el envío de un valor a través de este objeto (ver la sección *3.6.4 Parámetros de las salidas de tipo termoválvula* en la página 18).
Nombre	**Objeto X: Valve X status | Valve X switching feedback**
Función	Objeto de comunicación de 1 byte para notificación de estado.
Descripción	Con cada cambio se envía automáticamente el estado de la termoválvula a través de este objeto.
Nombre	**Objeto X: Valve X | Valve X script recall**
Función	Objeto de comunicación de 1-bit/1-byte/2-bytes para la ejecución de scripts.
Descripción	Programación avanzada de scripts (ver la sección *3.12 Modo programador* en la página 36).

3.5 Objetos de las entradas

3.5.1 Tabla de entradas binarias

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
0	Input 1 | Input 1 switch on/off telegram	1 bit	1.001	●	o	●	●	●
1	Input 2 | Input 2 switch on/off telegram	1 bit	1.001	●	o	●	●	●
2	Input 3 | Input 3 switch on/off telegram	1 bit	1.001	●	o	●	●	●
3	Input 4 | Input 4 switch on/off telegram	1 bit	1.001	●	o	●	●	●
4	Input 5 | Input 5 switch on/off telegram	1 bit	1.001	●	o	●	●	●
5	Input 6 | Input 6 switch on/off telegram	1 bit	1.001	●	o	●	●	●

3.5.2 Tabla de entradas de tipo persiana

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
0	Input motor 1 stop | Short press: Stop/step	1 bit	1.001	●			●	●
1	Input motor 1 move | Long press: Move up/down	1 bit	1.001	●			●	●
2	Input motor 2 stop | Short press: Stop/step	1 bit	1.001	●			●	●
3	Input motor 2 move | Long press: Move up/down	1 bit	1.001	●			●	●
4	Input motor 3 stop | Short press: Stop/step	1 bit	1.001	●			●	●
5	Input motor 3 move | Long press: Move up/down	1 bit	1.001	●			●	●

3.5.3 Tabla de entradas de tipo regulación

Objeto	Nombre | Función	Longitud	DPT	Flags				
				C	R	W	T	U
0	Input 01 | Input 1 switch on/off telegram	1 bit	1.001	●		●	●	●
199	Input 11 Dimming | Input 1 dimming telegram	4 bits	3.007	●		o	●	●
1	Input 02 | Input 2 switch on/off telegram	1 bit	1.001	●		●	●	●
200	Input 12 Dimming | Input 2 dimming telegram	4 bits	3.007	●		o	●	●
2	Input 03 | Input 3 switch on/off telegram	1 bit	1.001	●		●	●	●
201	Input 13 Dimming | Input 3 dimming telegram	4 bits	3.007	●		o	●	●

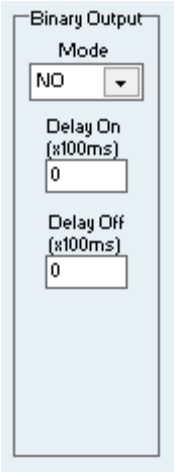
3	Input 04 | Input 4 switch on/off telegram	1 bit	1.001	●		●	●	●
202	Input 14 Dimming | Input 4 dimming telegram	4 bits	3.007	●		o	●	●
4	Input 05 | Input 5 switch on/off telegram	1 bit	1.001	●		●	●	●
203	Input 15 Dimming | Input 5 dimming telegram	4 bits	3.007	●		o	●	●
5	Input 06 | Input 6 switch on/off telegram	1 bit	1.001	●		●	●	●
204	Input 16 Dimming | Input 6 dimming telegram	4 bits	3.007	●		o	●	●

● = Configuración por defecto
o = Opcional

3.6 Parámetros de las salidas

3.6.1 Parámetros de salidas binarias

Cuando una salida se configura como una salida individual binaria se pueden configurar los siguientes parámetros:



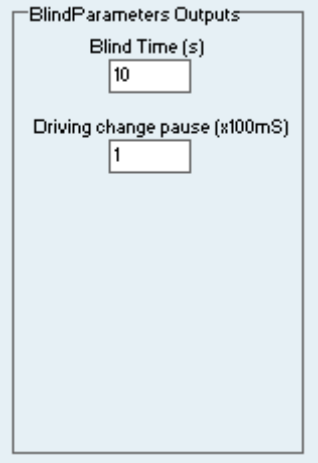
Mode: Modo. Puede ser normalmente abierto o normalmente cerrado. En el modo “normalmente abierto” el relé de la salida es controlado siguiendo la lógica estándar: 1 = cerrado, 0 = abierto. En el modo “normalmente cerrado” el relé se controla con la lógica inversa: 1 = abierto, 0 = cerrado.

Delay on (x100ms): Retardo de activación. Factor de tiempo (en base a 100 milisegundos) que el dispositivo espera antes de activar la salida (abrir o cerrar contacto dependiendo del modo de trabajo seleccionado).

Delay off (x100ms): Retardo de desactivación. Factor de tiempo (en base a 100 milisegundos) que el dispositivo espera antes de desactivar la salida (abrir o cerrar contacto dependiendo del modo de trabajo seleccionado).

3.6.2 Parámetros de salida tipo persiana

Cuando la salida es configurada como de tipo persiana se pueden configurar los siguientes parámetros:

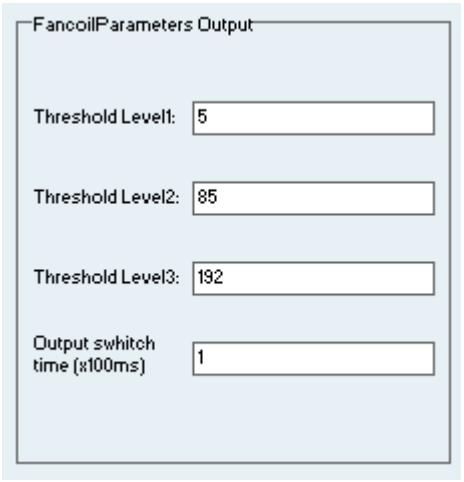


Blind time (s): Tiempo de persiana. En este parámetro se debe configurar el tiempo medido en segundos que tarda la persiana en subirse completamente.

Driving change pause: Pausa de cambio de dirección. Este parámetro es un factor (en base a 100 milisegundos) para un tiempo límite que debe esperar el actuador antes de cambiar de dirección cuando la persiana se está moviendo.

3.6.3 Parámetros de salida de tipo fan-coil

Cuando la salida es configurada como de tipo fan-coil se pueden configurar los siguientes parámetros:



El valor recibido a través del objeto de comunicación de control de fan-coil <<Fan X mode [1 byte]>> es comparado con estos niveles umbral por el actuador (ver la sección *3.4.6 Descripción de salidas de tipo fan-coil* en la página 13).

Threshold level 1: Nivel umbral 1. (De 0 a 255). Si el valor de control de fan-coil es menor que este valor umbral las salidas de fan-coil se desactivan. Si el valor de control es mayor, se activa la salida 1 (O1).

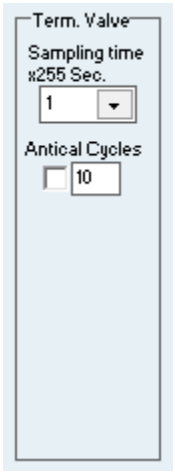
Threshold level 2: Nivel umbral 2. (De 0 a 255). Si el valor de control de fan-coil es menor que este valor umbral se activa la salida 1 (O1). Si el valor de control es mayor, se desactiva la salida 1 (O1) y se activa la salida 2 (O2).

Threshold level 3: Nivel umbral 3. (De 0 a 255). Si el valor de control de fan-coil es menor que este valor umbral se activa la salida 2 (O2). Si el valor de control es mayor, se desactiva la salida 2 (O2) y se activa la salida 3 (O3).

Output switch time (x100 ms): Tiempo de activación de la salida. Este parámetro es un factor (en base a 100 milisegundos) que indica el tiempo que el actuador debe esperar antes de cambiar de una velocidad a otra.

3.6.4 Parámetros de las salidas de tipo termoválvula

Cuando la salida es configurada como de tipo termoválvula se pueden configurar los siguientes parámetros:



Sampling time: Tiempo de muestreo. Factor de tiempo (en base a 255 segundos) para establecer el ciclo de trabajo de la señal PWM generada por la salida.

Anti-lime cycles: Ciclos anti cal. Activación o desactivación de la función anti cal. Cuando esta función está activada, el dispositivo cierra automáticamente la salida durante 5 segundos, de acuerdo al tiempo de muestreo multiplicado por el número de ciclos definido.

3.7 Parámetros de las entradas

3.7.1 Parámetros de las entradas binarias

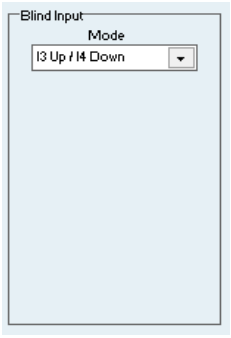
Cuando se define el modo de trabajo de una entrada como binario se pueden configurar los siguientes parámetros:

Rising edge: Flanco ascendente. Valor enviado cuando tiene lugar un flanco de subida en la entrada (generado cuando la entrada se conecta a referencia). Puede ser configurado para enviar siempre un "1" lógico, un "0" lógico, conmutar entre "0" y "1" o no realizar ninguna acción.

Falling edge: Flanco descendente. Valor enviado cuando tiene lugar un flanco de bajada en la entrada (generado cuando la entrada se desconecta de referencia). Puede ser configurado para enviar siempre un "1" lógico, un "0" lógico, conmutar entre "0" y "1" o no realizar ninguna acción.

3.7.2 Parámetros de entradas de tipo persiana

Cuando se define el modo de trabajo de una entrada como de tipo persiana se pueden configurar los siguientes parámetros:



I1 Up/ I2 Down: Modo estándar de 2 pulsadores. Entrada impar para subir la persiana y entrada par para bajarla.

Para la primera entrada, el comportamiento para la pulsación larga es mover la persiana hacia arriba enviando un "0" a través del objeto <<Input motor X move>>. El comportamiento de la pulsación corta es stop/step, enviando un "0" a través del objeto <<Input motor X stop>>.

Para la segunda entrada, el comportamiento para la pulsación larga es mover la persiana hacia abajo enviando un "1" a través del objeto <<Input motor X move>>. El comportamiento para la pulsación corta es stop/step, enviando un "1" a través del objeto <<Input motor X stop>>.

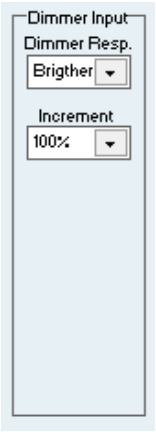
I1 Up-Down / I2 Not used: Modo de funcionamiento con un solo pulsador. Permite subir y bajar la persiana desde la misma entrada. En este caso la entrada par correspondiente no se utiliza.

El comportamiento de la entrada para pulsación corta es stop/step enviando de forma alterna "0" o "1" a través del objeto <<Input motor X stop>>. El comportamiento para la pulsación larga es subir o bajar la persiana enviando de forma alterna "0" o "1" a través del objeto <<Input motor X move>>.

El parámetro que define el tiempo necesario para que sea considerada una pulsación larga se establece desde la pestaña de parámetros generales (ver la sección *3.8 Parámetros generales* en la página 21).

3.7.3 Parámetros de las entradas de tipo regulador

Cuando se define el modo de trabajo de una entrada como de tipo regulador se pueden configurar los siguientes parámetros:



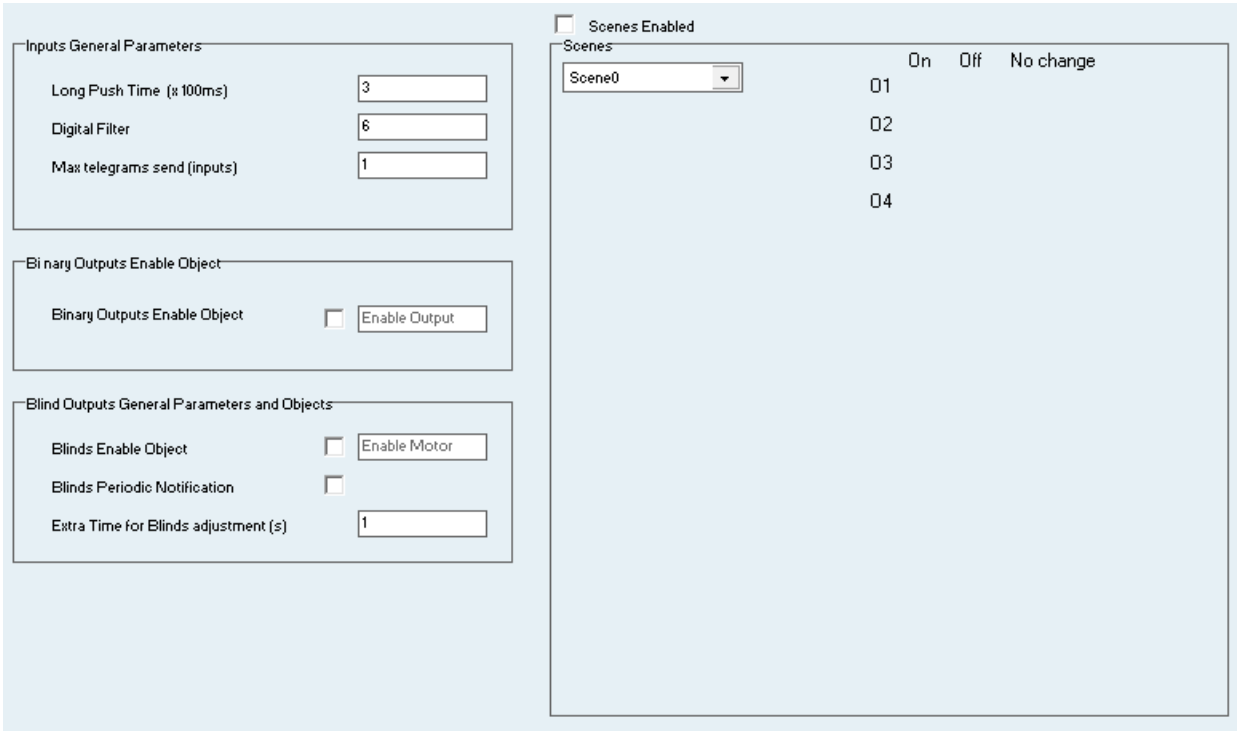
Dimmer response: Respuesta del regulador. Define la respuesta de la entrada a la hora de regular. Para pulsación larga el comportamiento puede ser regular siempre hacia arriba (más luminosidad), regular siempre hacia abajo (menos luminosidad) o regular de forma alterna hacia arriba y hacia abajo. El comportamiento para pulsación corta está predefinido y es siempre conmutar encendido/apagado.

El parámetro que define el tiempo necesario para que sea considerada una pulsación larga se establece desde la pestaña de parámetros generales (ver la sección *3.8 Parámetros generales* en la página 21).

Increment: Intervalo de regulación enviado con cada pulsación larga.

3.8 Parámetros generales

En esta pestaña se pueden configurar diversos parámetros generales del dispositivo:



3.8.1 Entradas

Input long push time: Es el tiempo que el dispositivo usa para diferenciar entre pulsación corta y pulsación larga, utilizada en entradas de tipo regulación y de tipo persiana.

Input digital filter: Es un parámetro utilizado para filtrar rebotes en las entradas y evitar falsas pulsaciones. Este valor se mide en ciclos de microcontrolador. El valor recomendado es 6 y puede variar de 0 a 10.

Max telegrams send (inputs): Este parámetro permite al usuario definir el número de direcciones de grupo utilizadas para enviar telegramas cuando hay una acción en alguna entrada. Por defecto, cuando se asocia más de una dirección de grupo al objeto de comunicación de una entrada, solo la primera envía telegramas. Es posible utilizar este parámetro para enviar telegramas a través de más de una dirección de grupo.

3.8.2 Salidas binarias

Binary outputs enable object: Selecciona si utilizar o no un objeto de comunicación extra que permita habilitar o deshabilitar el control de salidas binarias desde comandos KNX a través del bus (no se habilitan o deshabilitan las entradas).

Polaridad: 1 = control habilitado / 0 = control deshabilitado.

3.8.3 Salidas de tipo persiana

Blinds outputs enable object: Selecciona si utilizar o no un objeto de comunicación extra que permita controlar las persianas desde comandos KNX a través del bus (no se habilitan o deshabilitan las entradas).

Polaridad: 1 = control habilitado / 0 = control deshabilitado.

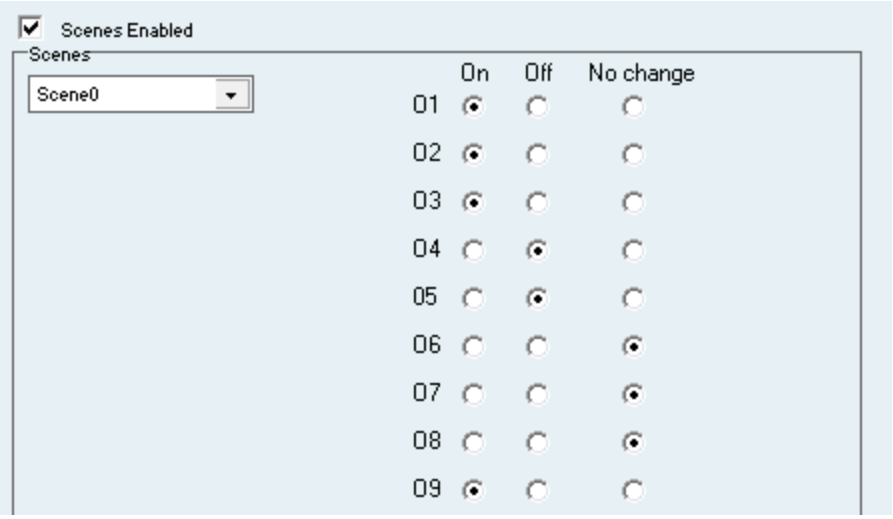
Blinds periodic notification: Activa o desactiva la notificación periódica del estado de la posición de las persianas cuando se están moviendo (cada segundo).

Extra time for blind adjustment: Define un tiempo adicional en segundos para el ajuste completo de la posición de la persiana cuando alcanza el límite máximo o mínimo. La salida correspondiente permanece cerrada un tiempo extra medido en segundos.

3.8.4 Escenas

Scenes enabled: Activa o desactiva los objetos de comunicación para ejecutar las escenas.

Primero seleccione el valor de escena desde el menú desplegable de la izquierda. Este es el valor que el objeto de comunicación correspondiente deberá recibir para ser ejecutada. Después, determine el comportamiento de cada salida: Activar (On), desactivar (Off) o no realizar ninguna acción (No change). (El estado final de la salida dependerá del modo de trabajo, normalmente abierto o normalmente cerrado).

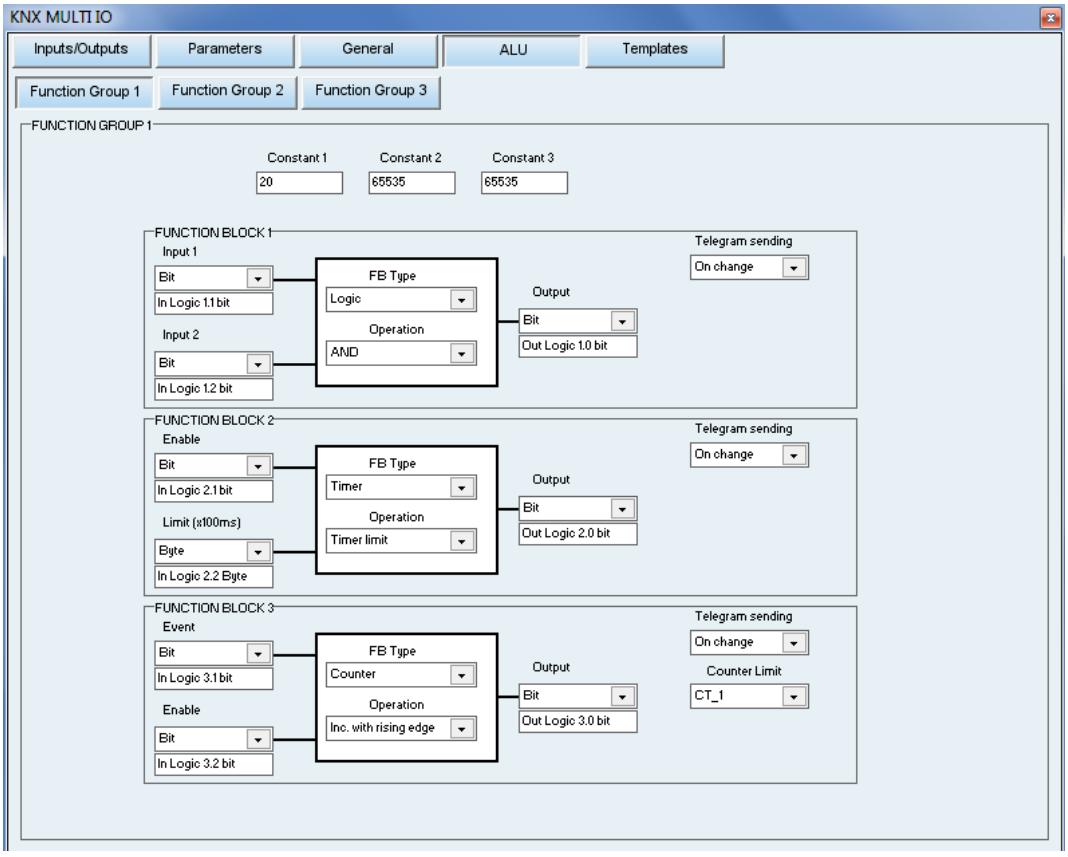


3.9 Unidad aritmético lógica

3.9.1 Introducción

Este dispositivo incorpora una avanzada unidad aritmético lógica (ALU) que permite la implementación de complejas operaciones lógicas, la programación de temporizadores, contadores, etc. utilizando variables internas o externas y una interfaz intuitiva de programación.

La unidad aritmético lógica está compuesta por 3 grupos de funciones, y 3 bloques de función en cada grupo. Un bloque de función tiene 2 entradas y 1 salida, con 3 objetos de comunicación cada uno referido a un tipo de dato distinto (bit, byte, 2 bytes).



La nomenclatura utilizada para los objetos de comunicación es la siguiente:

[in/out] [logic X.Y] [size]

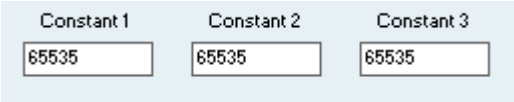
In/Out: Indica si el objeto es una entrada o una salida del bloque de función.

Logic X.Y: Siendo X el número del bloque de función (de 1 a 9) y Y el número de entrada (1, 2) o salida (0).

Size: Indica el tipo de dato del objeto de comunicación.

Debajo de cada entrada o salida hay una casilla que indica el nombre del objeto de comunicación. Este nombre puede ser editado por el programador.

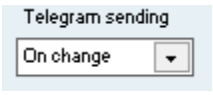
Un bloque de función puede operar con valores recibidos desde el bus a través de los objetos de entrada o con parámetros fijos, que son configurados en las casillas de "Constantes" en la parte superior (hay 3 para cada grupo de función).



Otra posibilidad es el uso de variables intermedias, llamadas "VARX". La ventaja de este tipo de variables es evitar el envío de telegramas al bus cuando es necesario enlazar el resultado de una operación resultante de un bloque con la entrada de otro bloque de función en la misma aplicación. Hay hasta 4 variables intermedias disponibles para cada grupo de función. Una variable intermedia de un grupo de función no puede ser utilizada en otro bloque.

Una característica importante de esta aplicación es que un único dispositivo puede implementar múltiples aplicaciones ALU con hasta 9 bloques de función, de modo que es posible la realización de complejas operaciones enlazando salidas y entradas con variables intermedias u objetos de comunicación.

La salida de un bloque de función puede funcionar en modo de envío pasivo o activo. Utilice el selector que aparece a continuación para decidir si el resultado de la operación es enviado en cada cambio (activo) o es de sólo lectura (pasivo).



Para programar una operación, primero selecciona el tipo de bloque y la operación que será implementada. Después, seleccione el tipo de dato de las entradas (constantes, variables intermedias u objetos de comunicación) y finalmente el tipo de dato de salida.

3.9.2 Tipo de bloques

Es posible configurar cada uno de los 3 bloques a partir de una de estas 3 posibles categorías:

Funciones aritmético lógicas:

- Operaciones lógicas: operaciones AND, OR, XOR, NAND, NOR y NXOR.
- Comparaciones: igual, distinto, mayor, mayor o igual, menor, menor o igual.
- Operaciones aritméticas: suma, resta, multiplicación y división.

Temporizadores

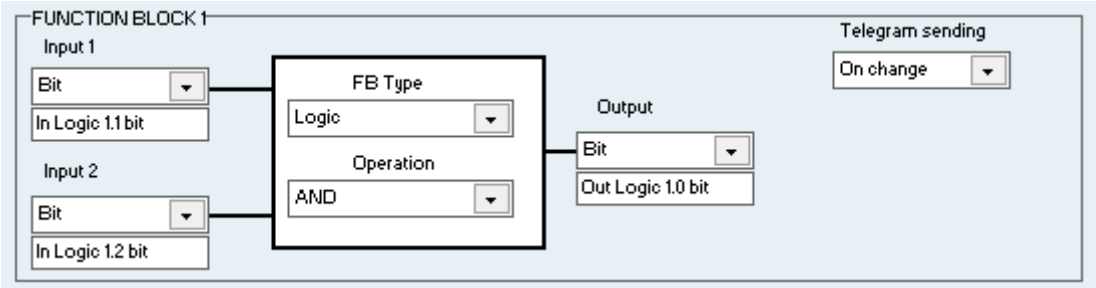
- Con límite
- PWM
- Cíclicos

Contadores

- Flanco ascendente
- Flanco descendente
- Contador de eventos "1" o distintos de "0"
- Contador de eventos "0"

3.9.3 Funciones aritmético lógicas

Ve al selector de tipo de bloque de función y seleccione “Logic” y la operación deseada.



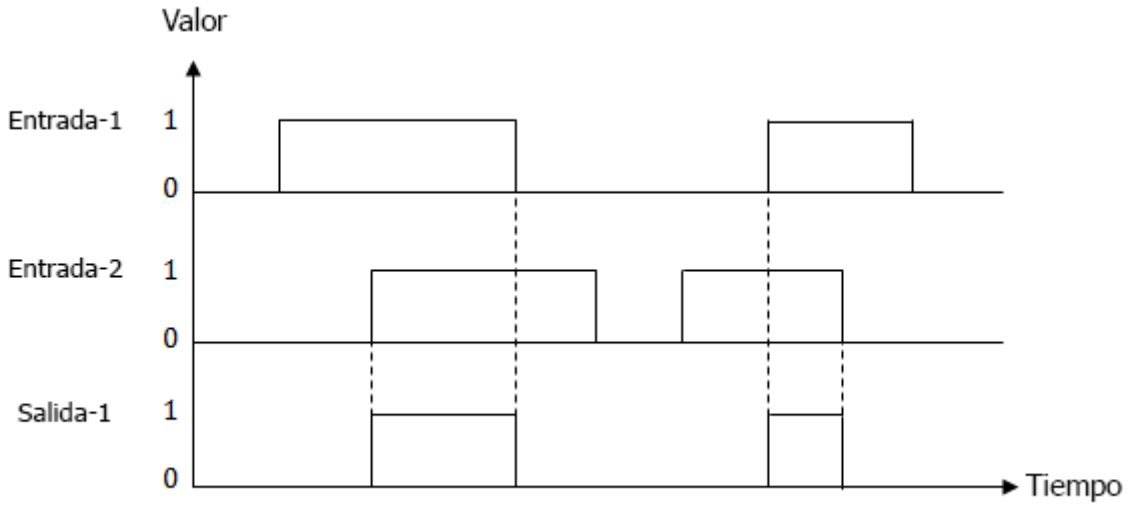
Input 1 / Input 2 - Seleccione desde dónde se han obtenido los valores de entrada: Pueden ser obtenidos desde el bus seleccionando los objetos de comunicación en los menús desplegables de las entradas (input 1, input 2) de acuerdo al tipo de dato requerido (bit, byte, 2 bytes), o puede tratarse de una constante (CTEX) o de una variable intermedia (VARX).

Output - Del mismo modo que en las entradas, en las salidas puede seleccionarse el objeto de comunicación en función del tipo de dato requerido (bit, byte, 2 bytes) y además el resultado puede ser guardado en una de las variables intermedias disponibles (VARX). El valor de salida se actualiza cada vez que los objetos de entrada reciben un telegrama.

Operation - Esta opción permite seleccionar el comportamiento del bloque. En caso de operaciones de comparación (=, ≠, >, ≥, <, ≤), el valor enviado será “1” en caso de ser cierto y “0” si es falso. Es importante tener en cuenta que el tipo de dato de salida se truncará si el tamaño es menor que el tamaño de los datos de entrada.

Una vez programado el bloque de función, cada vez que una entrada reciba un telegrama la operación será ejecutada, enviando o no el resultado al objeto de comunicación correspondiente en función de si ha sido programado para envío continuo o de solo lectura.

Por ejemplo: La siguiente figura muestra cómo programar una operación lógica “AND” con 2 bits de entrada. Cuando se recibe un valor a través del objeto de comunicación de entrada (In Logic 1.1 bit, In Logic 1.2 bit) el bloque de función calcula la operación y el resultado s envía a través del objeto de salida (Out Logic 1.0 bit).

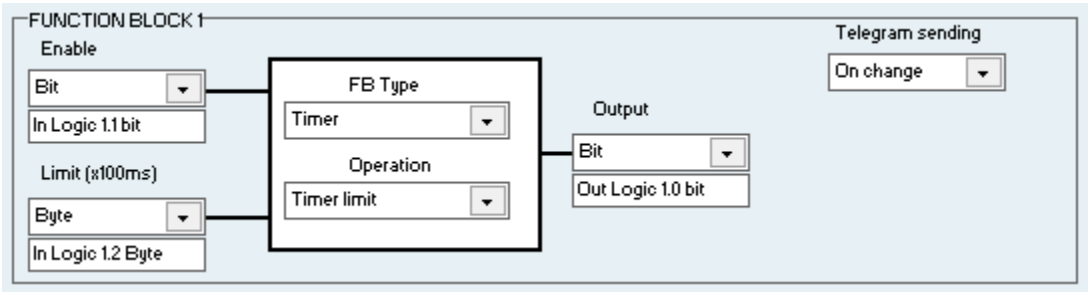


3.9.4 Temporizadores

Se dispone de tres tipos de temporizadores que se pueden seleccionar en la lista “Tipo de temporizador”. El comportamiento de cada tipo se explicará a continuación.

LÍMITE DEL TEMPORIZADOR

Envía un telegrama al bus o una variable intermedia cuando se excede un valor límite.

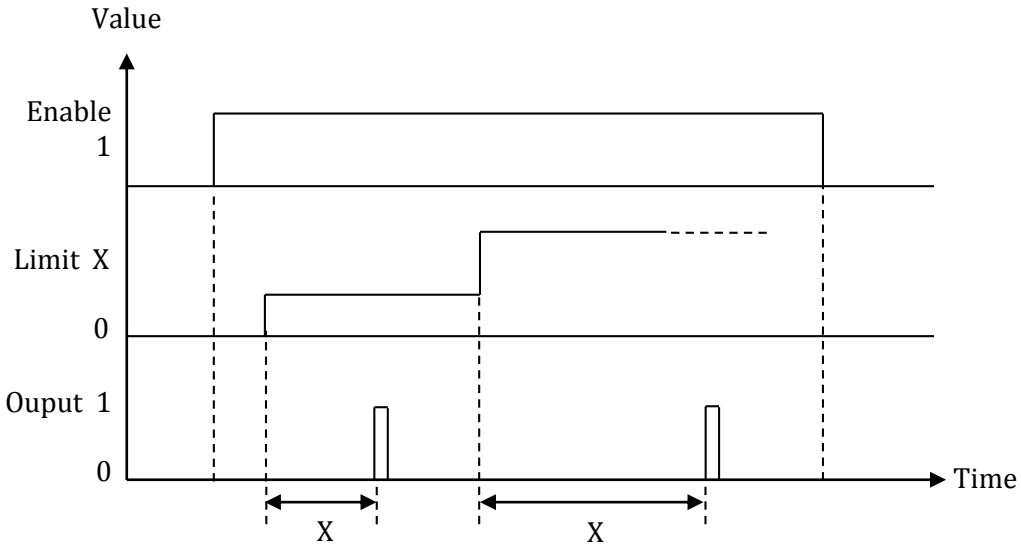


Enable - Habilita/deshabilita la función de temporizador con un valor de 0/1. Cuando se envía un “1” a esta entrada el temporizador es activado y la cuenta atrás se inicia con un valor distinto de 0 recibido en la entrada “Limit”. Cuando se envía un “0”, el temporizador es desactivado y la salida envía un 0.

Limit - El temporizador inicia la cuenta atrás desde su valor límite, que puede ser tomado de una constante o de un objeto de comunicación de bus. Cuando finaliza, se envía el telegrama de salida y no se inicia de nuevo hasta la recepción de un nuevo telegrama de activación. El valor límite puede ser de 1 o 2 bytes cuando es enviado a través de bus. Puede ser obtenido también de una constante. El temporizador también puede pausar la cuenta atrás cuando recibe un valor de “0” y prosigue con cualquier otro valor distinto de 0.

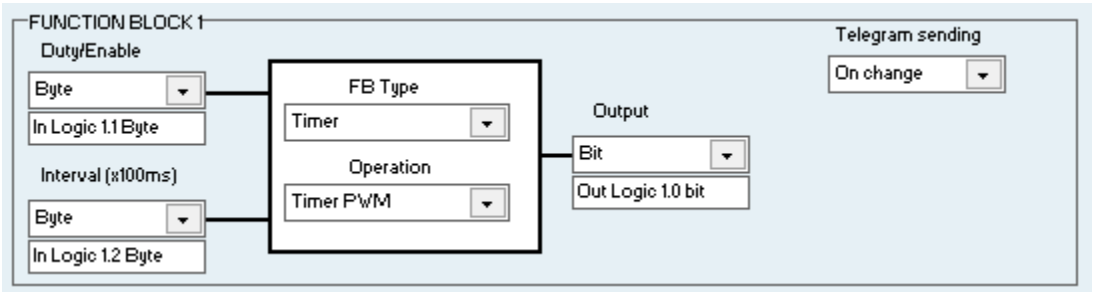
El valor del temporizador debe ser introducido en una escala de 1000 ms, por ejemplo, para un valor de 3 segundos el valor a introducir debe ser 30.

Output - La salida puede tener un tamaño de 1 bit, 1 byte o 2 bytes, siendo el valor a enviar “1” en cualquiera de los casos. También es posible enviar el evento a una variable intermedia, que puede ser la entrada de otro bloque de función.



TEMPORIZADOR PWM

Este tipo de temporizador envía telegramas de “1” y “0” alternativamente de forma cíclica durante el intervalo programado. El tiempo entre “1” y “0” depende del valor de ciclo de trabajo (de 1 a 10). Un valor de 0 desactiva el temporizador.

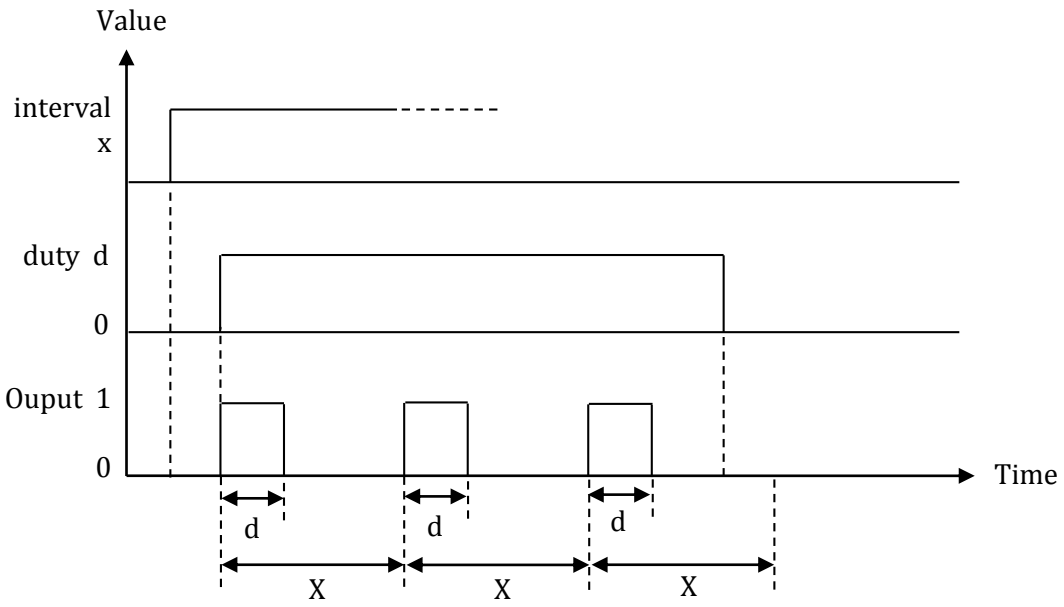


Duty/Enable - El ciclo de trabajo admite valores de 1 byte o 2 bytes de 1 a 10. También permite desactivar el temporizador cuando el valor es “0”. Si el valor es 10 significa que la salida estará activada el 100% del tiempo.

Interval - Es el periodo de señal. Este valor puede ser obtenido del bus (1 byte o 2 bytes), de un valor constante o de una variable intermedia.

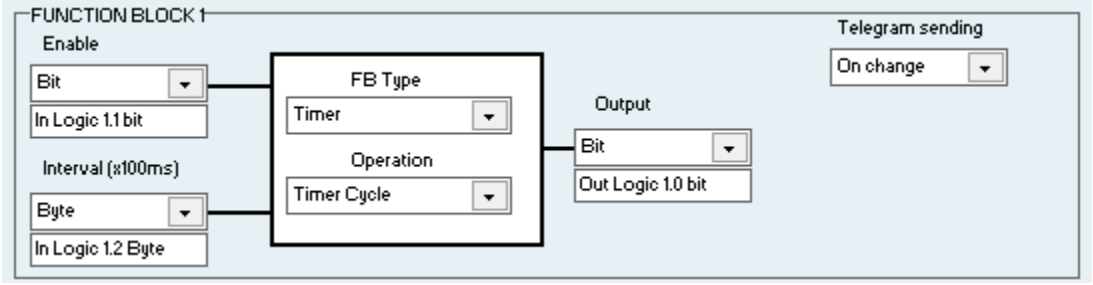
El valor del temporizador debe ser introducido en una escala de 100 ms, es decir, un intervalo de 3 segundos equivaldría a un valor de 30.

Output - La salida puede tener un tamaño de 1 bit, 1 byte o 2 bytes, siendo el valor enviado “1” en cualquier caso. También es posible enviar el evento a una variable intermedia, que podría ser a su vez la entrada de otro bloque de función.



TEMPORIZADOR CÍCLICO

Este tipo de temporizador envía un telegrama de valor "1" cíclicamente cuando se excede el tiempo definido en el intervalo.

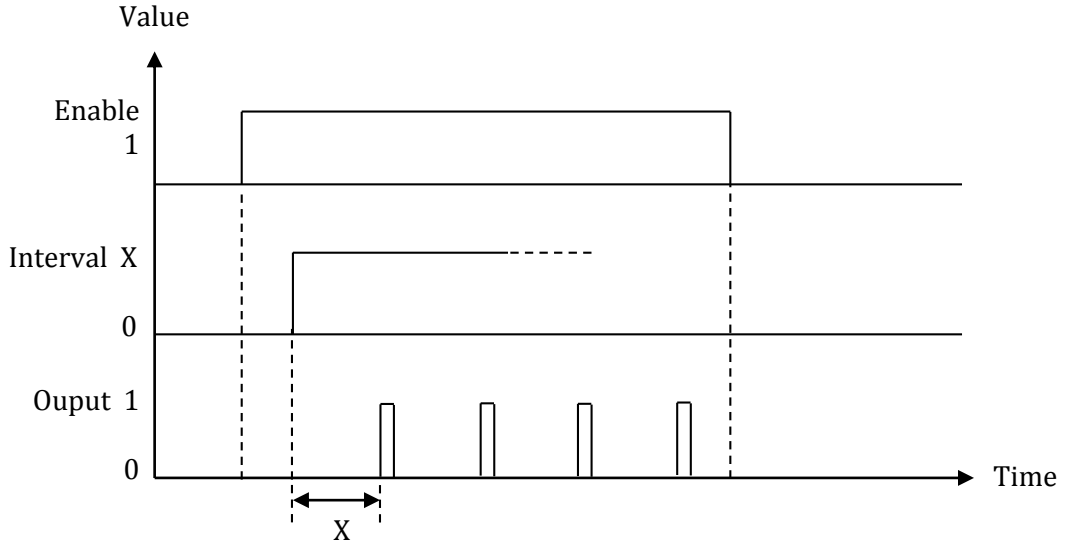


Enable - Permite activar o desactivar el temporizador. Esta entrada puede ser asociada a un objeto de comunicación de bus (1 bit, 1 byte o 2 bytes) o una variable intermedia.

Interval - Es el periodo de la señal. Este valor puede ser obtenido del bus (1 byte o 2 bytes), de un valor constante o de una variable intermedia.

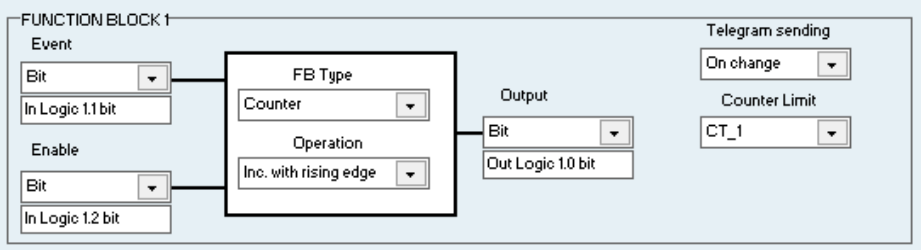
El valor del temporizador debe ser introducido en una escala de 100 ms, es decir, un intervalo de 3 segundos equivaldría a un valor de 30.

Output - La salida puede tener un tamaño de 1 bit, 1 byte o 2 bytes, siendo el valor enviado "1" en cualquier caso. También es posible enviar el evento a una variable intermedia, que podría ser a su vez la entrada a otro bloque de función.



3.9.5 Contadores

Hay cuatro tipos de contadores que pueden ser seleccionados en la lista “Tipo de contador”, que serán explicados a continuación.



Counter Limit – Permite establecer el número de eventos sobre los que el contador envía el telegrama de final. Este valor puede obtenerse de un objeto de comunicación (1 byte o 2 bytes), de un valor constante o de una variable intermedia. Un valor de 10 desactiva el contador.

Event – Es la entrada del contador (indica qué eventos se contarán). Puede ser un objeto de comunicación de 1 bit, 1 byte o 2 bytes o una variable intermedia.

Output – Cuando el valor del contador excede el límite, se envía un telegrama de valor “1”. Puede ser un objeto de comunicación de 1 bit, 1 byte o 2 bytes o una variable intermedia.

Enable – Sobrecarga en la entrada 2. El contador puede ser habilitado o deshabilitado con un bit en la entrada 2. Al mismo tiempo, el valor límite puede ser seleccionado con objetos de comunicación de 1 byte o 2 bytes.

FLANCO ASCENDENTE

Cuando la entrada detecta un flanco de subida (cambio de 0 a 1) el contador incrementa su valor interno. Cuando el contador alcanza el límite envía un telegrama al bus de valor “1”. Después vuelve al estado deshabilitado inicial.

FLANCO DESCENDENTE

Cuando la entrada detecta un flanco de bajada (cambio de 1 a 0) el contador incrementa su valor interno. Cuando el contador alcanza el límite envía un telegrama al bus de valor “1”. Después vuelve al estado deshabilitado inicial.

INCREMENTO CON “1”

Cuando la entrada detecta un telegrama de valor “1” el contador incrementa su valor interno. Cuando el contador alcanza el límite envía un telegrama al bus de valor “1”. Después vuelve al estado deshabilitado inicial.

INCREMENTO CON “0”

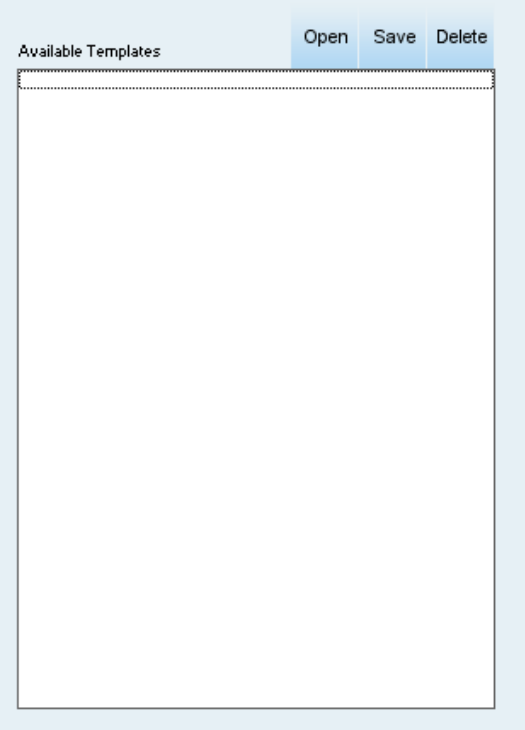
Cuando la entrada detecta un telegrama de valor “0” el contador incrementa su valor interno. Cuando el contador alcanza el límite envía un telegrama al bus de valor “1”. Después vuelve al estado deshabilitado inicial.

3.10 Plantillas

La aplicación de plug-in permite guardar o cargar cualquier configuración realizada por el programador.

La funcionalidad de copiar/pegar o transferir parámetros incluida en el ETS4 no está soportada cuando el dispositivo se programa con una aplicación externa, pero el plug-in permite salvar la parametrización completa de cualquier dispositivo con el objetivo de ser usado en cualquier otro dispositivo o incluso en cualquier otro proyecto.

Haz click en la ventana “templates” para ver las plantillas disponibles almacenadas.



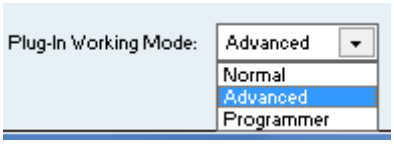
Haciendo click sobre el botón “save” serán guardados en un fichero todos los parámetros configurados, nombres editados, objetos de comunicación y toda la información de la programación actual.

Haz click sobre cualquier plantilla y presiona “open” para cargarla o “delete” para borrarla de la base de datos.

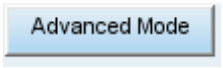
Las funcionalidades de copiar/pegar y transferir parámetros incluidas en el ETS4 no están soportadas cuando el dispositivo es programado desde una aplicación externa. Para copiar una configuración de un actuador a otro dispositivo utilice una plantilla.

3.11 Modo avanzado

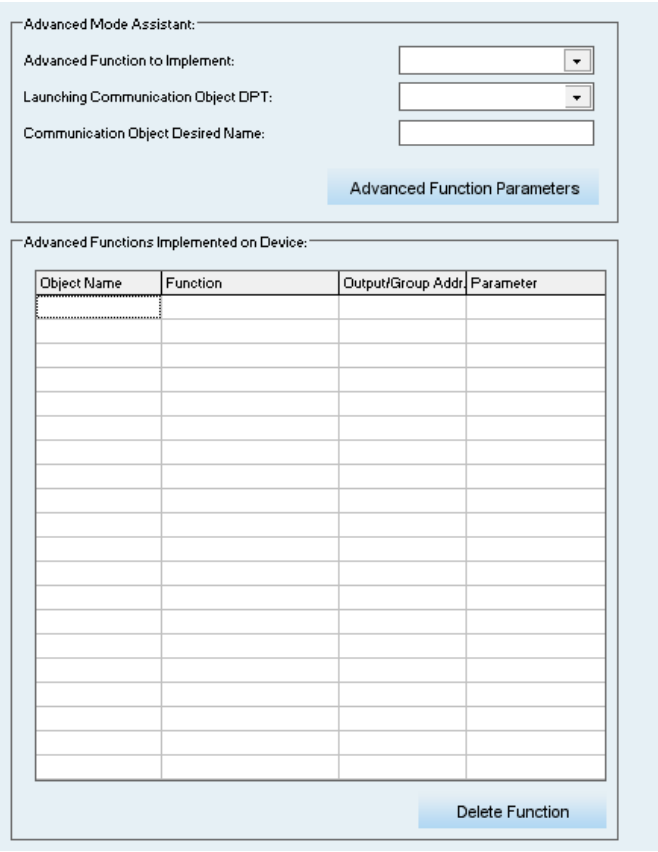
Este actuador incorpora la posibilidad de implementar funciones avanzadas y complejas operaciones permitiendo al programador ejecutar una simple escena o desarrollar su propia ejecución avanzada que puede ser lanzada desde bus, enviar telegramas o recibir parámetros, contar, realizar operaciones, etc.



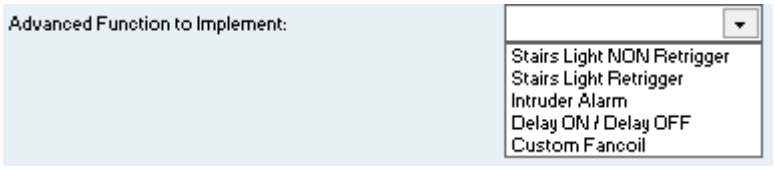
Los modos de programación avanzada son accesibles solamente seleccionando “advanced mode” o “programmer mode” en el menú desplegable en la parte inferior de la pantalla.



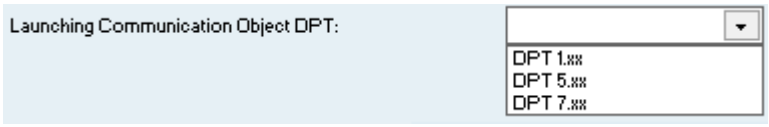
El modo avanzado muestra una pestaña extra en la parte superior de la pantalla donde el programador puede crear funciones avanzadas utilizando un asistente. Las funciones creadas estarán disponibles para ser ejecutadas a través de objetos de comunicación, como será explicado a continuación.



Para crear una función, primero utiliza el selector para decidir qué tipo de función implementar de las funciones disponibles:

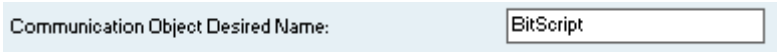


Después selecciona el tipo de dato de la función:



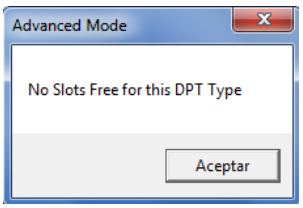
Este es el tamaño del objeto de comunicación (dpt 1.*, dpt 5.*, dpt 7.*.) asociado a la función avanzada. Un valor enviado a este objeto de comunicación ejecutará la función avanzada.

Tras esto, el programador puede editar el nombre del objeto para una identificación más sencilla del mismo en el ETS.



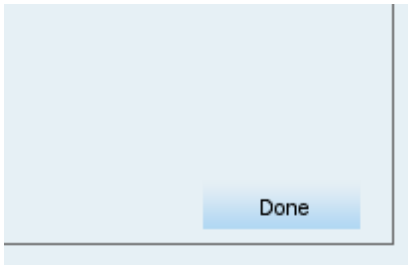
When the type of advanced function, the size of its object and the name is configured, do click on the “advanced function parameters” button to continue with the settings of the specific parameters of the function.

Hay un límite en el número de funciones avanzadas que pueden ser implementadas en el dispositivo, y también depende del número de salidas configurado. Hay una función de 1 bit, una de 1 byte y una de 2 bytes disponibles por cada salida.



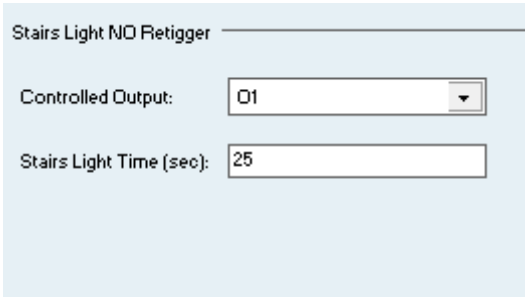
Un mensaje de “no slots free” indicará la necesidad de configurar más salidas, o si se ha excedido el número máximo de funciones avanzadas.

Cuando ya se han configurado los parámetros específicos de la función avanzada, presione el botón “Done”.



3.11.1 Stairs light non retrigger

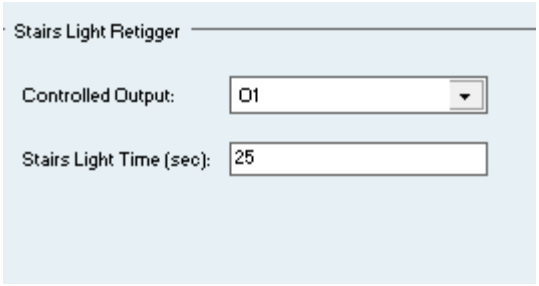
Luz de escalera sin redisparo. Esta función avanzada permite programar una luz de escalera decidiendo que salida se controla y durante cuánto tiempo está activada.



Enviando un "1" al objeto de comunicación asociado a la función se activa la salida. Después, la salida es desactivada automáticamente tras el tiempo configurado. Si se recibe otro "1" antes de que se desactive la cuenta atrás no empieza de nuevo.

3.11.2 Stairs light retrigger

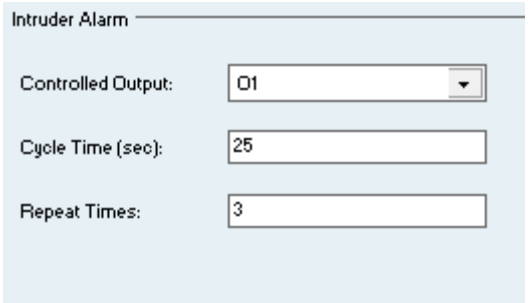
Luz de escalera redisparable. Esta función avanzada permite programar una luz de escalera decidiendo que salida se controla y durante cuánto tiempo está activada.



Enviando un "1" al objeto de comunicación asociado a la función se activa la salida. Después, la salida es desactivada automáticamente tras el tiempo configurado. Si se recibe otro "1" antes de que se desactive la cuenta atrás se reinicializa.

3.11.3 Intruder alarm

Alarma de intrusión. Esta función avanzada permite la programación de una alarma básica de intrusión, decidiendo qué salida es controlada, el tiempo que permanece activada y cuántos ciclos se realizan.

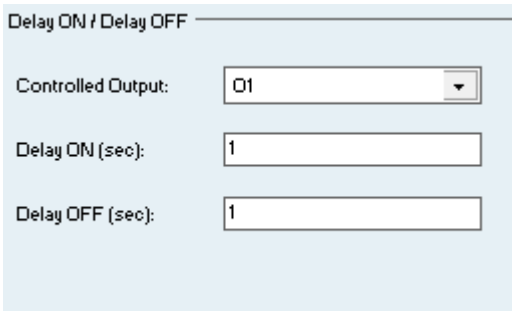


Esta función está pensada para el control de una sirena o cualquier otro dispositivo de alarma. Mediante el envío de un telegrama de valor "1" al objeto de comunicación asociado a la función la salida es activada el tiempo de ciclo definido.

Después, la salida es automáticamente desactivada durante otro tiempo de ciclo y el proceso se repite el número de veces que haya configurado.

3.11.4 Delay on / delay off

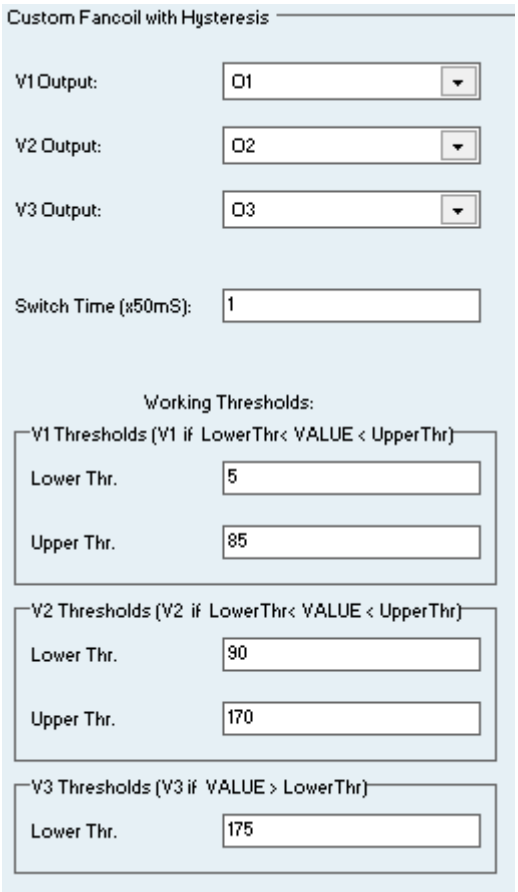
Retardo al encendido / apagado. Esta función avanzada permite la activación o desactivación de retardos decidiendo qué salida se controla y el tiempo de cada retardo.



Mediante el envío de un telegrama de valor “1” al objeto de comunicación asociado a la función la salida es activada tras el tiempo configurado en “Delay ON”. Cuando el telegrama es de valor “0” la salida es desactivada tras el tiempo configurado en “Delay OFF”.

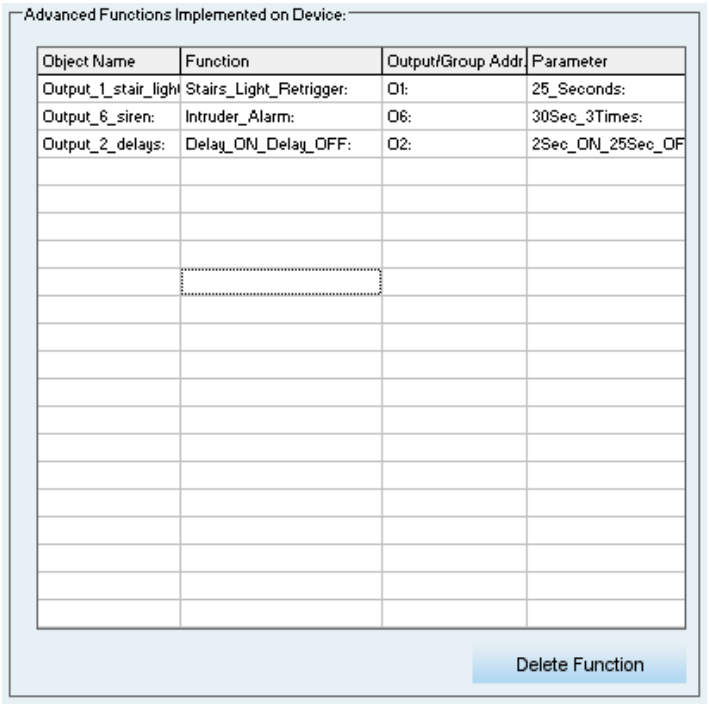
3.11.5 Custom fan-coil

Esta función avanzada permite la programación de un control de fan-coil personalizado decidiendo qué salida corresponde a cada velocidad y definiendo los valores umbral de trabajo.



Mediante el envío de valores de 1 byte o 2 bytes al objeto de comunicación asociado a la función el dispositivo decide qué salida (velocidad) debe ser activada de acuerdo a los valores umbral configurados.

Las funciones avanzadas que ya han sido programadas se muestran en la lista de la izquierda. Cualquier función puede ser eliminada seleccionándola y presionando el botón “delete function” en la parte inferior de la lista.



3.12 Modo programador

3.12.1 Descripción de scripts

Este dispositivo incorpora un método de programación avanzada con su propio lenguaje de programación, similar a otros lenguajes como C. Los scripts permiten al programador ejecutar una simple escena o desarrollar su propia ejecución avanzada de instrucciones que puede ser lanzada desde bus, enviando telegramas o recibiendo parámetros, contando, operando, etc.

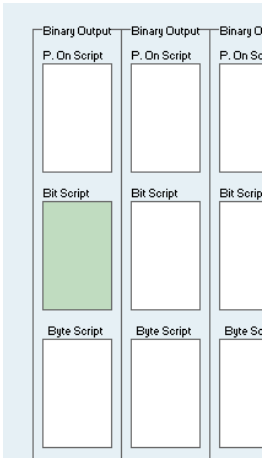
Hay 3 tipos de scripts, de acuerdo a los parámetros que pueden recibir del bus (a través de una dirección de grupo asociada): Scripts de 1 bit, scripts de 1 byte y scripts de 2 bytes. Hay un cuarto tipo de script, llamado “power-on script”, que se ejecuta cada vez que el dispositivo recupera la tensión de alimentación.

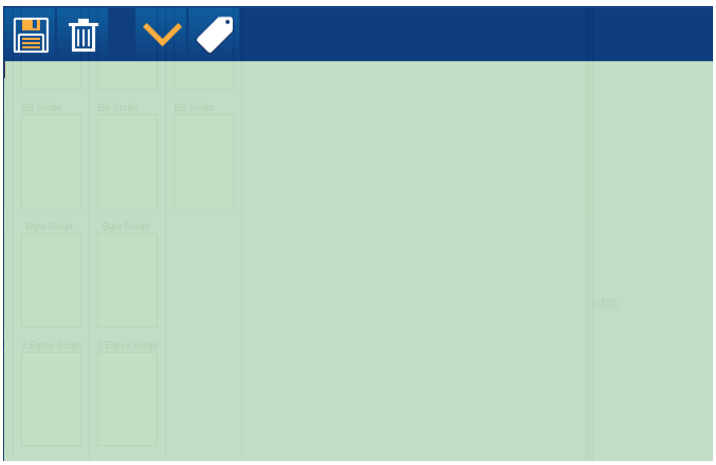
Los scripts pueden ser activados o desactivados desde la pestaña de entradas y salidas.

Cada script puede ocupar hasta un máximo de 20 Kb. Cuando se programa un script , se muestra un mensaje de “script size out of bounds” (tamaño del script por encima del límite) para indicar que se ha sobrepasado el número de líneas / comandos.

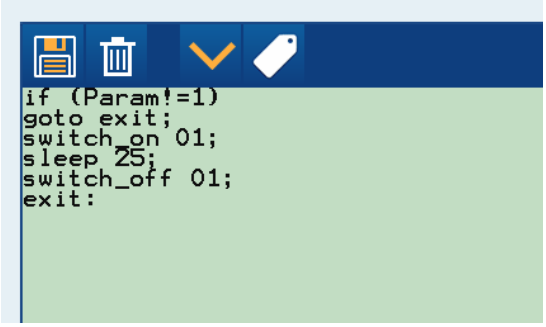
3.12.2 Editor

Al seleccionar un script aparecerá su ventana de programación. El script que se está modificando aparece resaltado en verde.



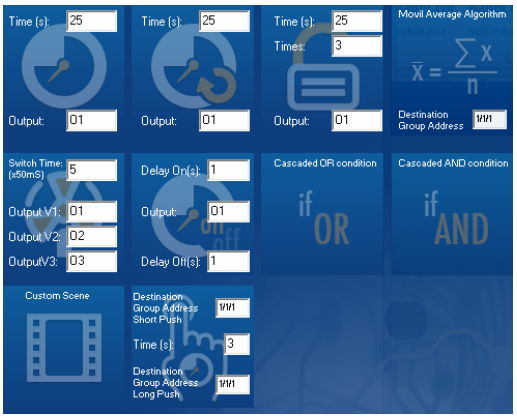


La programación del script puede ser realizada de forma manual de acuerdo al lenguaje que será explicado más adelante o haciendo uso del asistente. Cuando se haya finalizado la edición del script presione el icono de guardado para guardar el script o haga click sobre el icono de la papelera para descartar cambios.



Utilice el botón mostrado a continuación para comprobar si ha habido algún error de sintaxis. Si no se muestra ningún mensaje significa que el código se ha escrito correctamente. De lo contrario se notificará el error mediante una ventana emergente.

El asistente incluye algunos ejemplos comunes de funciones avanzadas que pueden ser programadas mediante script. Cada una de esas funciones puede tener varios parámetros que pueden ser configurados desde el asistente.



3.12.3 Lenguaje de programación

El lenguaje de programación utilizado en los scripts es similar a otros lenguajes. Por ejemplo, un punto y coma indica el final de cada instrucción. Para una mejor comprensión del código cada instrucción puede ser escrita en una nueva línea y hacer uso de tabulaciones. Existen diversas funciones y palabras reservadas que serán explicadas a continuación.

PARÁMETROS RECIBIDOS POR EL BUS

param

Este es el valor recibido por el bus KNX a través de la dirección de grupo asociada al script. Puede ser de 1 bit, 1 byte o 2 bytes dependiendo del tipo de script.

Puede ser usado en operaciones, funciones y otras instrucciones, incluso con operadores de distinto tamaño, por ejemplo:

```
...
var1=(param*2)+234;
...
```

También es posible asignarle un nuevo valor directamente o como resultado de una operación durante la ejecución del script, por ejemplo:

```
...
var1=(param*2)+234;
param=0;
...
```

El parámetro mantendrá el valor recibido o asignado hasta el final de la ejecución del script. Cuando se recibe un valor de 2 bytes de coma flotante, el valor real tomado por "param" es el binario. No están soportadas las operaciones de coma flotante.

INTERNAL VARIABLES

varX

Hay hasta 10 variables internas en cada script que pueden ser usadas en cualquier operación, función u otras instrucciones, incluso con operadores de otro tamaño. No es necesario declararlas o inicializarlas porque con cada ejecución del script toman el valor "0".

```
var1=(param*2)+234;
var2=var2+var1;
```

Del mismo modo que el parámetro "param", es posible asignarles un nuevo valor directamente o como resultado de una operación durante la ejecución del script. Las variables mantienen el valor asignado hasta el final de la ejecución del script.

Las variables internas del script pueden ser usadas solo en ese script, es decir, si necesitamos enviar una variable de un script a otro deberemos utilizar el parámetro de bus "param".

ARITHMETIC AND COMPARISON OPERATORS

Hay dos tipos de operadores: aritméticos y de comparación.

Aritméticos: = , + , - , * , / (igualación, suma, resta, multiplicación y división).

Estos operadores aritméticos pueden ser usados en cualquier instrucción con variables, valores constantes y también el parámetro "param". El resultado de una operación aritmética puede ser asignado al parámetro o a una variable interna mediante el operador "=". Los paréntesis pueden ser utilizados siguiendo las mismas reglas que una operación matemática común.

```
var1=5;
var2=(param/2)+(var1*2);
var3=var3+var2;
```

Todas la operaciones aritméticas deben realizarse con enteros, las operaciones de coma flotante no están soportadas.

De comparación: == , >= , <= , > , < , != (igual, mayor o igual, menor o igual, mayor, menor, distinto).

Los operadores de comparación son utilizados en la función condicional "if", como se explica a continuación.

FUNCTIONS: GOTO AND IF

Goto X; Ir a una línea del código.

La función "goto" provoca que la ejecución del código prosiga desde una línea determinada, identificada con una etiqueta "X" elegida por el programador.

Una etiqueta se define con una combinación de letras y/o números, pero el primer caracter debe ser una letra siempre, y debe finalizar con dos puntos ":". Por ejemplo:

```
...
goto next;
        var1=0;
        ...
        Var10=0;
next:
        var1=param+1;
...
```

El programa continúa desde la etiqueta "next" y las instrucciones tras la función "goto next" no se ejecutan

If (A X B) Función condicional "if"

La función condicional "if" permite al programador que se ejecuten o no una o más instrucciones del script dependiendo del resultado de una operación de comparación.

A y B pueden ser variables internas, valores constantes o el parámetro "param", mientras que X es un operador de comparación.

Si la operación de comparación es cierta, la siguiente línea a la función "if" se ejecuta. Si la operación de comparación es falsa, el programa salta la instrucción. Por ejemplo:

```
If(param>100)
        var1=param;
If(param<=100)
        var1=0;
```

Si el parámetro no es mayor que 100 la siguiente instrucción no será ejecutada

En el ejemplo anterior, var1 tomará el valor del parámetro recibido solamente si el valor es mayor que 100. En caso contrario var1 toma el valor 0.

Cuando el programador necesita que se ejecute más de una instrucción dependiendo de la condición puede repetir la misma función "if", pero resulta mejor opción la utilización de la función "if" opuesta en combinación con la función "goto" y una etiqueta. Es decir:

```
If(param>100)
        var1=param;
If(param>100)
         var2=param*2;
If(param>100)
         var3=param*3;
If(param<=100)
        var1=0;
If(param<=100)
        var2=0;
If(param<=100)
        var3=0;
```

El script anterior es el mismo que el mostrado a continuación utilizando la condición opuesta:

```
If(param<=100)
        goto next;
                var1=param;
                var2=param*2;
                var3=param*3;
next;
If(param>100)
        goto next2;
                var1=0;
                var2=0;
                var3=0;
next2;
```

Si el parámetro no es mayor de 100 la función "goto" es ejecutada y las siguientes funciones no

El programador puede utilizar funciones "if" anidadas con el objetivo de implementar condiciones compuestas. En el ejemplo siguiente el script comprueba si el valor del parámetro se encuentra entre 5 y 10. Si la condición es cierta el parámetro es guardado en la variable 1. Si no, la variable toma el valor 0.

```
If(param<5)
        If(param>10)
                var1=param;
var1=0;
```

Si el parámetro no es menor que 5, el programa salta a la instrucción var1 = 0

Cuando se utilizan funciones "if" anidadas, se puede pensar que si la primera condición es falsa, la siguiente instrucción no será ejecutada, y por tanto el segundo "if" no es evaluado, pero no funciona así.

En el ejemplo anterior, la instrucción que sigue a la primera función "if" termina en el primer punto y coma ";". Esto significa que si el parámetro no es menor que 5 el programa continúa en la instrucción "var1=0". Esto se puede entender mejor reescribiendo el script de la siguiente manera:

```
If(param<5)
        If(param>10) var1=param;
var1=0;
```

Las tabulaciones y los saltos de línea no son necesarios. Se utilizan para una mejor comprensión del código.

DIRECT COMMANDS

Existen varias instrucciones directas que son ejecutadas con palabras reservadas, como se explica a continuación.

Switch_on X; Siendo X el nombre de la salida, de O1 a O16, activa el relé correspondiente.
Switch_off X; Siendo X el nombre de la salida, de O1 a O16, desactiva el relé correspondiente.

Set X Y; Siendo X el nombre de la persiana y Y el porcentaje de apertura.

Sleep X; Siendo X el tiempo en segundos de 0 a 65535. Este comando pausa la ejecución del script durante el tiempo configurado.

ENVÍO DE TELEGRAMAS AL BUS

Send_telegram(X/X/X,dpt,value,command);

X/X/X – Es la dirección de grupo de 0/0/1 a 31/7/255.
dpt – Indica el tipo de dato enviado: dpt1, dpt5, dpt7 and dpt9.
value – Es el valor enviado al bus.
command – El comando del telegrama, que puede ser: escritura o lectura.

```
...
send_telegram(1/1/10,dpt5,150,write);
send_telegram(1/2/15,dpt9,21.50,write);
...
```

3.12.4 Scripts de ejemplo

RETARDO SIMPLE AL APAGAR

Descripción: Se necesita encender una luz y automáticamente apagarla tras un tiempo determinado. Se realizará un script de bit que se ejecutará al recibir "1" o "0" a través de la dirección de grupo asociada. El script siguiente es la forma más simple de programarlo.

Bit Script:

```
Switch_on OX;
Sleep Y;
Switch_off OX;
```

Notas: OX es el nombre interno de la salida (de O1 a O16 dependiendo del tipo de actuador). Si se requiere encender o apagar la salida de otro dispositivo la instrucción deberá ser "send_telegram()". Y es el tiempo medido en segundos que la luz está encendida.

RETARDO REDISPARABLE AL APAGAR

Descripción: El mismo ejemplo que el caso anterior pero ahora se requiere reinicializar la cuenta atrás (retardo) cada vez que se reciba un "1" a través de la dirección de grupo asociada.

Bit Script:

```
if(param==0)
        goto exit;
switch_on OX;
wait:
         if(param==1)
                  var1=0;
         param=0;
         if(var1>=Y)
                  goto exit;
         var1=var1+1;
        sleep 1;
goto wait;
exit:
        switch_off OX;
```

Notas: OX es el nombre interno de la salida (de O1 a O16 dependiendo del tipo de actuador). Si se requiere encender o apagar la salida de otro dispositivo la instrucción deberá ser "send_telegram()". Y es el tiempo medido en segundos que la luz está encendida. Las tabulaciones no son necesarias, se han utilizado para una presentación más clara del código.

RETARDO AL APAGAR CON EL TIEMPO COMO PARÁMETRO

Descripción: El mismo ejemplo que el anterior pero en este caso el tiempo que permanece la luz encendida es enviado por el bus a través de la dirección de grupo asociada. De forma que ahora será necesario un script de tipo byte.

Byte script:

```
if(param==0)
        goto exit;
switch_on OX;
wait:
        if(var1>=param)
                goto exit;
        var1=var1+1;
        sleep 1;
goto wait;
exit:
        switch_off OX;
```

Notas: OX es el nombre interno de la salida (de O1 a O16 dependiendo del tipo de actuador). Si se requiere encender o apagar la salida de otro dispositivo la instrucción deberá ser "send_telegram()". El parámetro recibido a través de la dirección de grupo es guardado en "param" y es el tiempo medido en segundos que la luz está encendida. Las tabulaciones no son necesarias, se han utilizado para una presentación más clara del código.

RETARDO AL ENCENDER CON EL TIEMPO COMO PARÁMETRO

Descripción: Se trata del mismo ejemplo que el anterior pero en este caso a la hora de encender una luz.

Byte script:

```
if(param==0)
        goto exit;
wait:
        if(var1>=param)
                goto exit;
        var1=var1+1;
        sleep 1;
goto wait;
exit:
        switch_on OX;
```

Notas: OX es el nombre interno de la salida (de O1 a O16 dependiendo del tipo de actuador). Si se requiere encender o apagar la salida de otro dispositivo la instrucción deberá ser "send_telegram()". El parámetro recibido a través de la dirección de grupo es guardado en "param" y es el tiempo medido en segundos que la luz está apagada. Las tabulaciones no son necesarias, se han utilizado para una presentación más clara del código.

CUENTA ATRÁS CON EL TIEMPO COMO PARÁMETRO

Descripción: Se utilizará un script de tipo byte para recibir un valor de tipo byte e iniciar una cuenta atrás enviando un telegrama al bus cada segundo.

Byte script:

```
if(param==0)
	goto exit;
count:
	param=param-1;
	sleep 1;
	send_telegram(X/X/X,dpt7,param,write);
	if(param<=0)
		goto exit;
goto count;
exit:
```

Notas: X/X/X es la dirección de grupo a través de la cual se desea enviar la cuenta atrás. El parámetro recibido a través de la dirección de grupo se guarda en el parámetro "param" y es el tiempo en segundos tomado para la cuenta atrás. Las tabulaciones no son necesarias, se han utilizado para una presentación más clara del código.

OPERACIÓN LÓGICA "OR" CON 3 VARIABLES

Descripción: Se necesita programar una operación lógica OR de 3 bits que serán activados con el valor "1" o desactivados con el valor "0", dependiendo del parámetro recibido.

Byte script:

```
param=65535;
wait:
	if(param==65535)
		goto wait;
	if(param==0)
		var1=0;
	if(param==1)
		var1=1;
	if(param==2)
		var2=0;
	if(param==3)
		var2=1;
	if(param==4)
		var3=0;
	if(param==5)
		var3=1;
	var4=var1+var2+var3;
	send_telegram(X/X/X,dpt1,var4,write);
	param=65535;
goto wait;
```

Notas: X/X/X es la dirección de grupo a través de la cual deseamos enviar el resultado. El parámetro recibido a través de la dirección de grupo se guarda en el parámetro "param" y es procesado. El resultado es formateado a tipo bit (dpt1) donde cualquier valor distinto de "0" es "1". Las tabulaciones no son necesarias, han sido utilizadas para una presentación más clara del texto.

RESTA EN VALOR ABSOLUTO

Descripción: Se desea recibir dos parámetros y calcular la resta en valor absoluto.

Byte script:

```
var1=param;
wait:
	if(param!=var1)
		goto wait;
	var2=param;
	if(var1>=var2)
		var3=var1-var2;
	if(var1<var2)
		var3=var2-var1;
	send_telegram(X/X/X,dpt7,var3,write);
```

Notas: X/X/X es la dirección de grupo a través de la cual se desea enviar el resultado. El primer parámetro recibido a través de la dirección de grupo es guardado en "var1" y después se espera por el segundo parámetro, que se guardará en "var2". Las tabulaciones no son necesarias, han sido utilizadas para una presentación más clara del código.

"AND" LÓGICA CON MÁS DE UNA INSTRUCCIÓN EN FUNCIÓN IF

Descripción: Se requiere la activación o desactivación de las salidas internas, dependiendo del valor de byte recibido, de 0 a 255. La primera salida se activa si la condición "0 >= param < 64" es cierta, la segunda salida si "64 <= param < 128", la tercera salida si "128 <= param < 192" y finalmente la cuarta salida si "192 <= param < 255". Además, cuando se activa una salida deben desactivarse el resto. Esto significa que necesitamos ejecutar más de una instrucción dentro de las funciones "if", así que la forma de implementarlo es escribir la condición opuesta y hacer uso de la función "goto".

Script:

```
if(param<0)
	goto if1;
if(param>=64)
	goto if1;
	switch_off O2;
	switch_off O3;
	switch_off O4;
	switch_on O1;
if1:

if(param<64)
	goto if2;
if(param>=128)
```

```
        goto if2;
        switch_off O1;
        switch_off O3;
        switch_off O4;
        switch_on O2;
if2:

if(param<128)
        goto if3;
if(param>=192)
        goto if3;
        switch_off O1;
        switch_off O2;
        switch_off O4;
        switch_on O3;
if3:

if(param<192)
        goto if4;
if(param>255)
        goto if4;
        switch_off O1;
        switch_off O2;
        switch_off O3;
        switch_on O4;
if4:
```

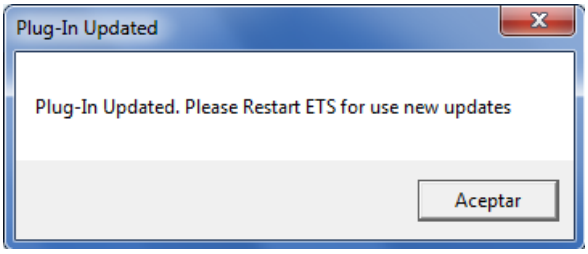
Notas: OX es el nombre interno de la salida (de O1 a O16 dependiendo del tipo de actuador). Si se requiere activar o desactivar la salida de otro dispositivo la instrucción deberá ser “send_telegram()”. El parámetro recibido a través de la dirección de grupo es guardado en “param” y es procesado. Las tabulaciones no son necesarias, se han utilizado para una presentación más clara del código.

3.13 Actualización del plug-in

El plug-in permite la comprobación de actualizaciones online y su instalación de forma automática. Para ello, basta con pulsar el botón que se muestra a continuación “Check Updates”.

No es necesario re-importar el catálogo del dispositivo en el ETS4 o instalar software adicional, simplemente basta con pulsar el botón antes mencionado, situado en la parte inferior de la pantalla principal, y el plug-in comenzará a descargar e instalar la nueva versión desde el servidor, si hay alguna disponible.

Tras la descarga e instalación del plug-in, se debe reiniciar el ETS4 con el objetivo de aplicar los cambios.



Si no hay ninguna versión nueva, se mostrará la ventana “no updates”.

4 Ejemplos de aplicación

4.1 Control individual de salidas con dos escenas

4.1.1 Dispositivos

Ref. 416410: Actuador de 6 entradas y 4 salidas.

Pulsadores convencionales KNX.

4.1.2 Descripción

El actuador controla 4 circuitos de luz de la habitación, que están conectados a las salidas de O1 a O4. Estas luces deberán ser controladas desde pulsadores conectados a las entradas del actuador.

Las luces deben poder ser controladas también a través de cualquier otro pulsador KNX conectado al bus en cualquier punto del mismo.

Dos pulsadores adicionales podrán llamar dos escenas del actuador para un encendido general “All On” y un apagado general “All Off” de las 4 salidas simultáneamente.

4.1.3 Enlaces de objetos

Ref. 416410 – Objeto 0 / O1 -> Objeto 8 / I1 – Ref. 416410

Ref. 416410 – Objeto 2 / O2 -> Objeto 9 / I2 – Ref. 416410

Ref. 416410 – Objeto 4 / O3 -> Objeto 10 / I3 – Ref. 416410

Ref. 416410 – Objeto 6 / O4 -> Objeto 11 / I4 – Ref. 416410

Ref. 416410 – Objeto 14 / Bit script 1 -> Objeto 12 / I5 – Ref. 416410

Ref. 416410 – Objeto 15 / Bit script 2 -> Objeto 13 / I6 – Ref. 416410

4.1.4 Configuración de parámetros

La siguiente configuración de los parámetros es la recomendada para este caso. La configuración ideal puede cambiar dependiendo de la aplicación o la instalación.

Parámetro		**Configuración**
General	Device type	6 inputs / 4 outputs
Salida 1	Output type	Binary
	Object: O1	☑
	Object: O1 Status	☑
	Object: Bit script 1	☑
Salida 2	Output type	Binary
	Object: O2	☑
	Object: O2 Status	☑
	Object: Bit script 1	☑
Salida 3/4	Output type	Binary
	Objects: O3,O4	☑
	Objects: O3,O4 Status	☑
Salida 1 a 4	Input type	Binary
	Objects: I1,I2,I3,I4	☑
	Rising edge	Switch (0/1)
	Falling edge	No action
Salida 5/6	Input type	Binary
	Objects: I5,I6	☑
	Rising edge	Send 1
	Falling edge	No action

4.1.5 Scripts

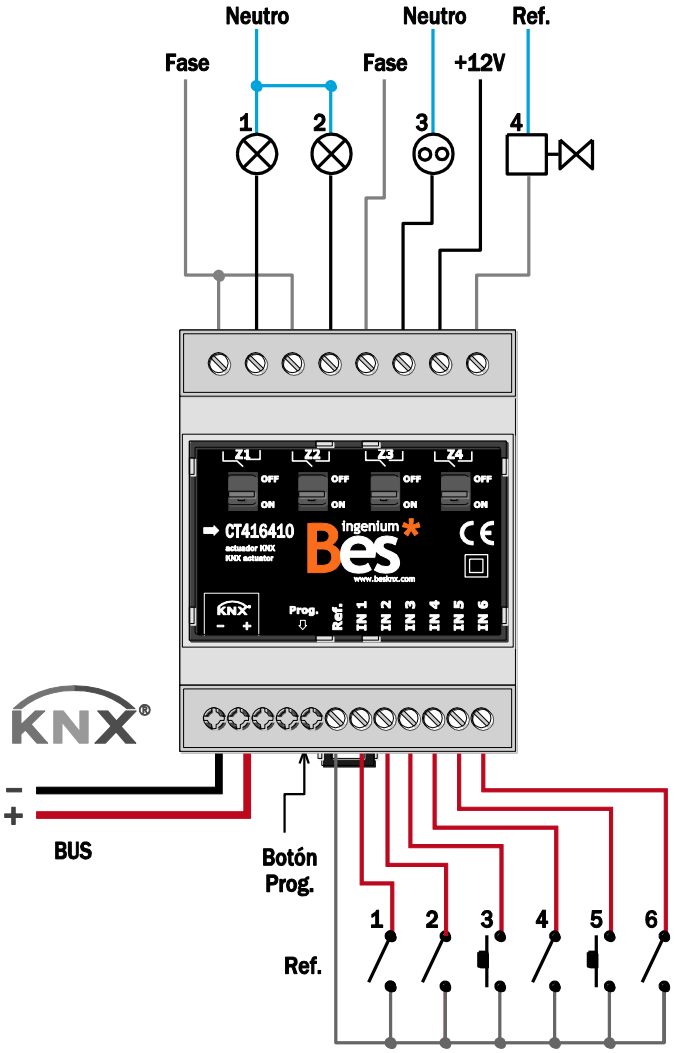
Bit script 1 (objeto 14)

```
Switch_on O1;
Switch_on O2;
Switch_on O3;
Switch_on O4;
```

Bit script 2 (objeto 15)

```
Switch_off O1;
Switch_off O2;
Switch_off O3;
Switch_off O4;
```

5 Instalación



Alimente las líneas de bajo voltaje (bus y entradas) en conductos *separados de la alimentación a 230 V y las salidas, con el objetivo de asegurar que existe el suficiente aislamiento y evitar así interferencias.*

No conecte el voltaje principal de 230 V o cualquier otro voltaje externo a ningún punto del bus ni a las entradas.

KNX products by ingenium

ingenium Bes

www.besknx.com

Ingenium, Ingeniería y Domótica S.L.

Parque Tecnológico de Asturias, Parcela 50

33428 Llanera, Asturias, Spain

T (+34) 985 757 195

tec@besknx.com

www.besknx.com

www.ingeniumsl.com

Versión del manual: v1.1