NXA

XCON.COM

Contents

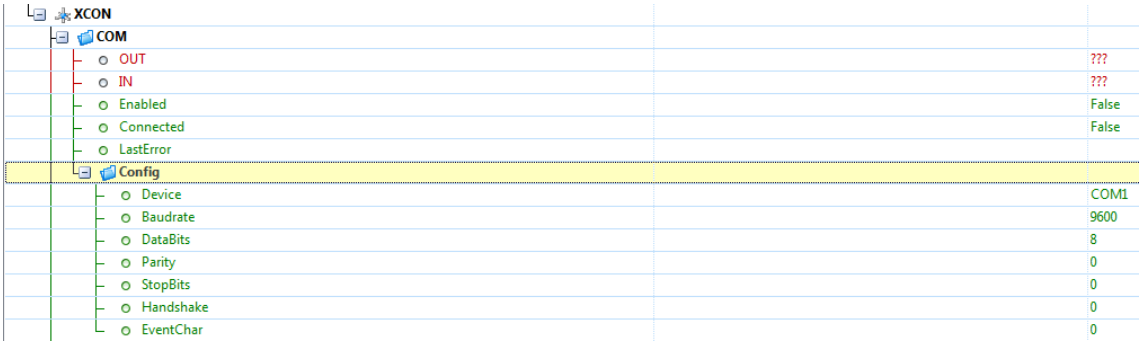
Xcon.COM Interface.

COM INTERFACE.

El COM INTERFACE es una interfaz de comunicaciones usada para establecer una comunicación a través de una interfaz serie. Se compone de 5 *entradas de comunicación* y una rama de *configuración*. Esta interfaz debe ser habilitada antes de poder ser utilizada.

Descripción de los ítems en interfaz gráfica Xcon.

La siguiente gráfica muestra cada uno de los ítems que podemos encontrar en la interfaz gráfica COM.



Se procede a especificar la funcionalidad y características de cada uno de los ítems:

PARÁMETROS DE COMUNICACIÓN

OUT

Data Type: STRING

Default value: None

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.OUT

Item ID: NETx.XCON.COM.OUT

Description: El buffer de bytes para enviar a la interfaz COM se escribe en este ítem.

IN

Data Type: STRING

Default value: None

Access Rights: Read only

Standard Path: NETx.XCON.<COM#>.IN

Item ID: NETx.XCON.COM.IN

Description: El buffer de bytes que recogemos desde la interfaz COM será escrito en este item.

ENABLE

Data Type: BOOL

Default value: 0 (False)

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Enabled

Item ID: NETx.XCON.COM.Enabled

Description: A través de este item habilitamos o deshabilitamos la interfaz COM. Este item es el encargado tanto de crear una conexión como de eliminarla.

CONNECTED

Data Type: BOOL

Default value: 0 (False)

Access Rights: Read only

Standard Path: NETx.XCON.<COM#>.Connected

Item ID: NETx.XCON.COM.Connected

Description: Muestra si la conexión se ha establecido o no.

LAST ERROR

Data Type: STRING

Default value: empty string

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.LastError

Item ID: NETx.XCON.COM.LastError

Description: Si ocurre un error, este será escrito en este item.

CONFIGURACIÓN

DEVICE

Data Type: STRING

Default value: “COM1”

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Config.Device

Item ID: NETx.XCON.COM.Config.Device

Description: Determina el Puerto COM que será direccionado en la conexión.

BAUDRATE

Data Type: INT4

Default value: 9600

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Config.Baudrate

Item ID: NETx.XCON.COM.Config.Baudrate

Description: Este item especifica el BaudRate de la interfaz de comunicaciones.

La siguiente table muesta los valores de baudrate estándar:

110	300	600
1200	2400	4800
9600	14400	19200
28800	38400	56000
57600	115200	

DATABITS

Data Type: INT4

Default value: 8

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Config.DataBits

Item ID: NETx.XCON.COM.Config.DataBits

Description: Determina los databits de cada carácter, los valores 5,7 y 8 son los mas communes mientras que 6 y 9 son raramente usados.

PARITY

Data Type: INT4

Default value: 0 (False)

Access Rights: Read only

Standard Path: NETx.XCON.<COM#>.Config.Parity

Item ID: NETx.XCON.COM.Config.Parity

Description: Especifica la paridad del mensaje. Si paridad es distinto de cero, se añadirá un bit de paridad con la comprobación especificada por ese número; es decir, si parity=1 se añadirá un último bit de paridad impar y se comprobará que es correcta con el fin de establecer mecanismos de seguridad y comprobación del envío y recepción correcta de mensajes.

La siguiente tabla muestra los distintos métodos de paridad disponibles:

Value	Meaning
0	No parity
1	Odd parity
2	Even parity
3	Mark parity
4	Space parity

STOPBITS

Data Type: INT4

Default value: 0

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Config.StopBits

Item ID: NETx.XCON.COM.Config.StopBits

Description: Determina el número de bits de parade al final de cada carácter, es usado para detector el final de cada carácter para recibir y resincronizar. Los dispositivos electromecánicos lentos pueden demander hasta dos bits de parade.

La siguiente tabla muestra los bits de parade permitidos:

Value	Meaning
0	1.0 stop bits
1	1.5 stop bits
2	2.0 stop bits

HANDSHAKE

Data Type: INT4

Default value: 0

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Config.Handshake

Item ID: NETx.XCON.COM.Config.Handshake

Description: Control de flujo por handshake. Valor a 1 para activarlo.

EVENTCHAR

Data Type: INT4

Default value: 0

Access Rights: Read and Write

Standard Path: NETx.XCON.<COM#>.Config.EventChar

Item ID: NETx.XCON.COM.Config.EventChar

Description: Determina el carácter a usar para indicar el final de una línea (p.ej. CHR(10) para Linefeed).

Ejemplo de prueba.

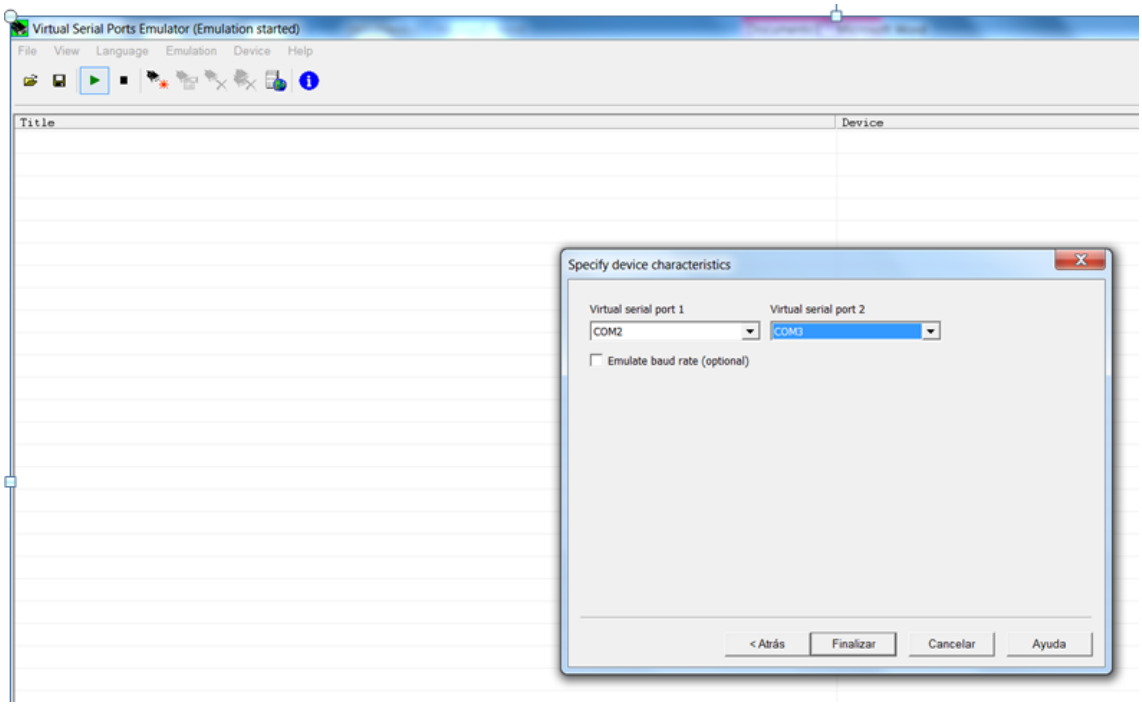
Para mostrar la funcionalidad del sistema, a continuación se desarrollará un ejemplo a través de una serie de pasos en el que mostraremos la potencialidad del COM INTERFACE.

Procedimiento:

Configuración de las comunicaciones

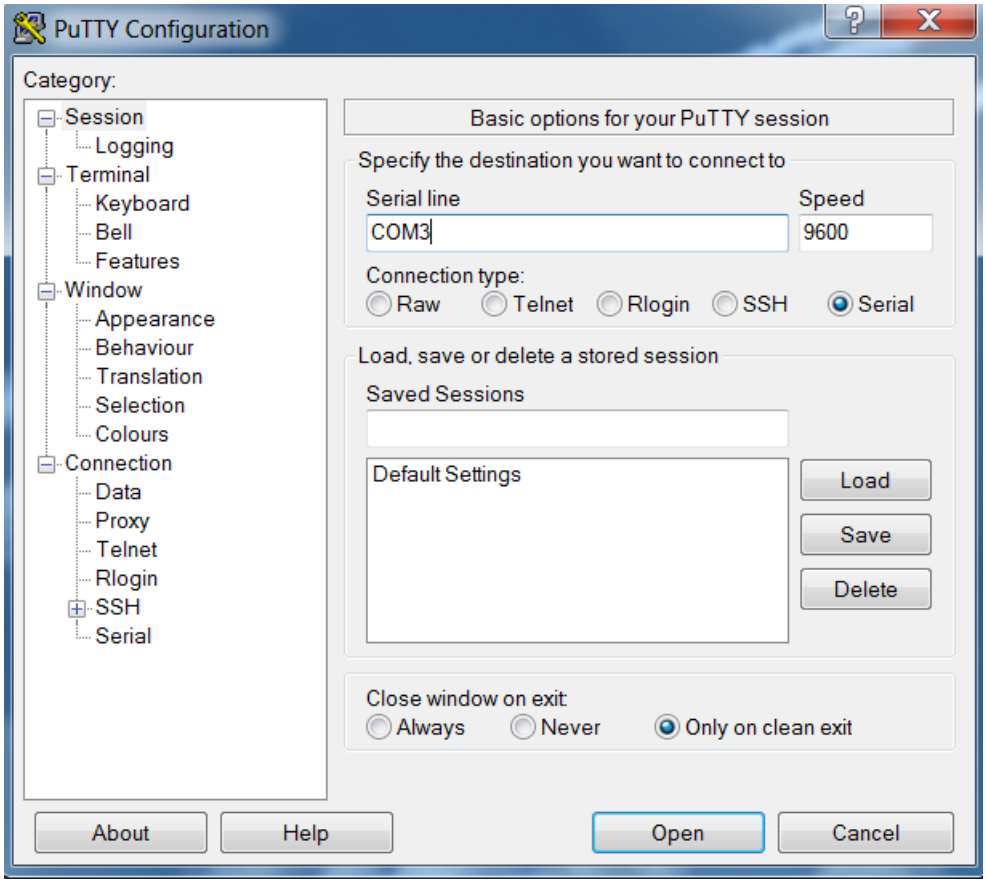
Para emparejar dos puertos de comunicación serie dentro del mismo equipo utilizaremos un emulador de puertos virtuales como es el Virtual Serial Ports Emulator (VSPE). De este modo conseguimos una conexión virtual entre dos puertos serie dentro de un mismo equipo.

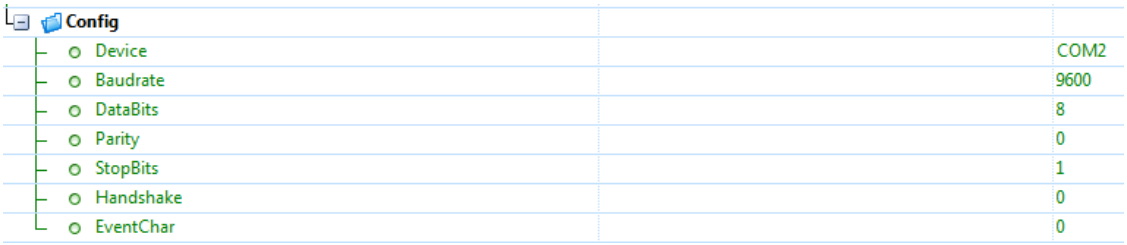
Con el VSPE emparejaremos los puertos COM2 y COM3 de nuestro equipo de forma que se establezca un enlace virtual entre los dos puertos actuando como si estuvieran vinculados físicamente.



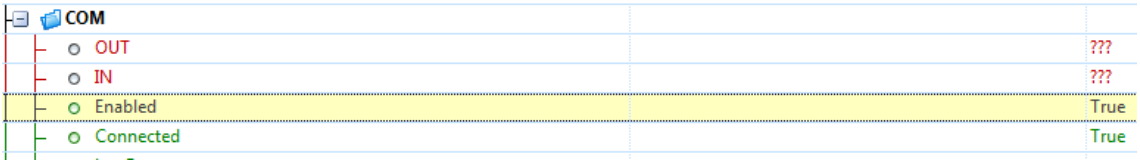
Como cliente serie utilizaremos un software cliente de comunicación como puede ser Putty.

En este punto del ejemplo enlazaremos el puerto COM2 a COM de BMS Server y COM3 al cliente Putty. Los parámetros de comunicación son los siguientes: 9600,8,N,1 (Baudrate, Databits, Parity, StopBits).





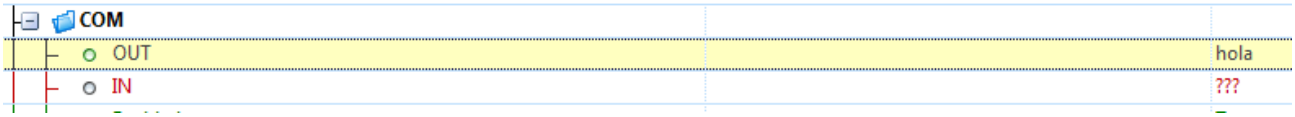
Una vez están enlazados los puertos de comunicación y configurados los parámetros de comunicación procederemos a realizar un sencillo envío de mensajes entre dichos puertos para ilustrar la funcionalidad del COM INTERFACE.

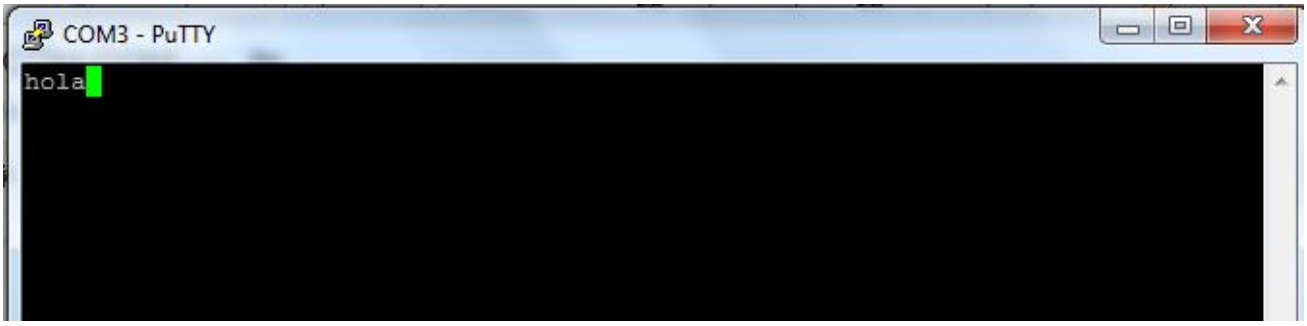


Envío de mensaje desde BMS Server.

Antes de comenzar a enviar mensajes, será necesario habilitar el item Enable escribiendo 1 sobre el ítem correspondiente.

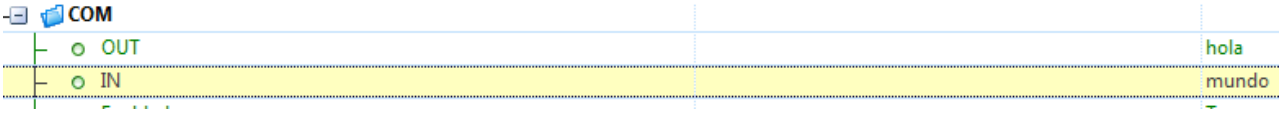
Para el envío de mensajes, basta con escribir sobre el campo OUT un valor de salida, de modo que si escribimos “hola”, en el terminal del cliente nos aparecerá el mensaje.





Envío de mensaje desde terminal cliente (PuTTY).

El envío de mensajes de cliente se realiza tan sólo con introducir el mensaje por el terminal, de forma que si escribimos “mundo” por el terminal, en la casilla “IN” de COM deberá aparecer el mensaje.



Manejo de funciones y creación de scripts en LUA.

Descripción:

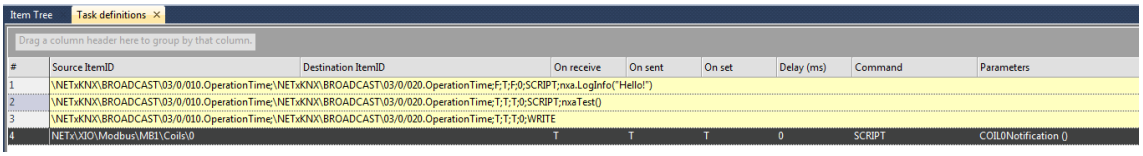
El objetivo del uso de scripts en BMS es el de asignar a una tarea o evento una serie de acciones pudiendo ser más o menos complejas. De esta forma se puede aprovechar todas sus funcionalidades de un modo más personalizado.

En este apartado, se va a asociar un script a una tarea determinada, a modo de ejemplo. Para ello imaginemos una situación en la que tenemos un dispositivo modbus cuya funcionalidad es la de una alarma en una puerta de una máquina frigorífica.

El dispositivo tendría un registro de configuración (no se explicará al detalle en este ejemplo) donde se podrían definir parámetros de temporización y temperatura, de forma que si se deja la puerta abierta y pasa el tiempo programado, se disparará la alarma, cambiando el estado de un coil determinado.

Procedimiento:

En primer lugar, debemos definir la tarea específica, en nuestro caso se reproduce el script cuando se modifica el valor del coil tanto en envío como en recepción, la función que se ejecuta es COIL0Notification().

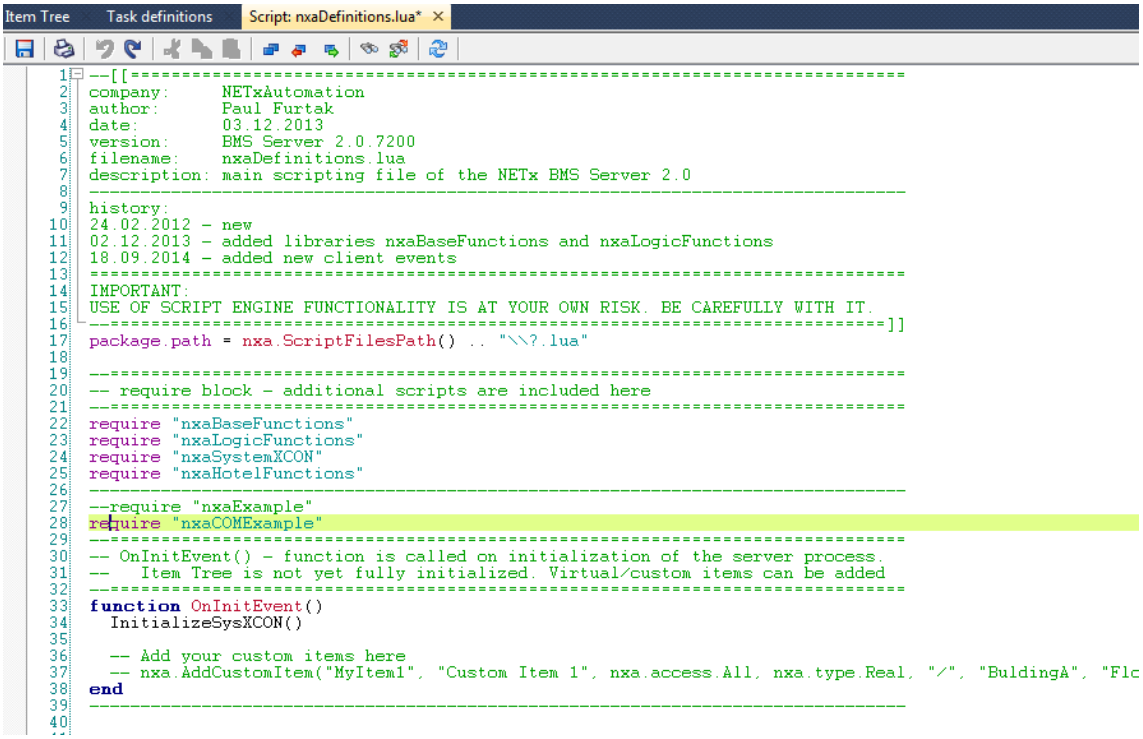


A continuación, debemos crear la función COIL0Notification(), la cual será la encargada de enviar la notificación a través del puerto serie de comunicaciones.

Para ello, podemos crear la función en un archivo ya existente o crear un archivo .lua nuevo en la misma ruta que los scripts del sistema que por defecto se encuentran en *C:\Program Files (x86)\NETxAutomation\NETx.BMS.Server.2.0\Workspaces\Workspace\ScriptFiles*

En nuestro caso se ha creado un archivo .lua nuevo llamado *nxaCOMExample.lua*

**Importante: Al crear un nuevo archivo de script, debemos añadirlo en *nxaDefinitions.lua* de la siguiente forma: *require "nxaCOMExample"* La siguiente imagen muestra cómo debería estar escrito y su localización en el archivo de definiciones.

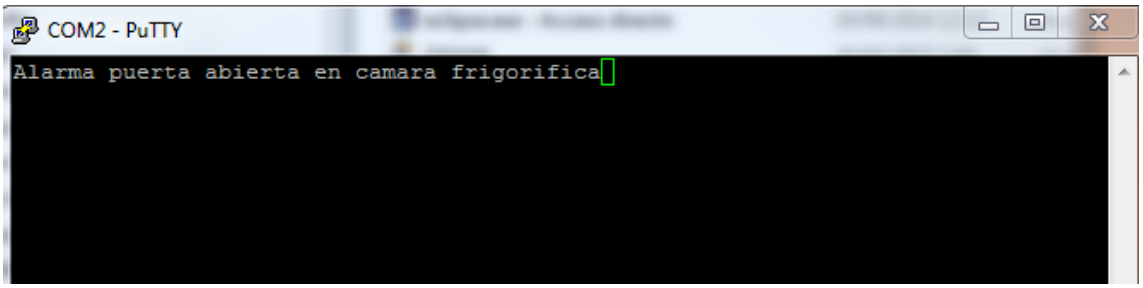


Una vez hecho esto, procedemos a crear la función. Aquí se configura el puerto cada vez que se produce el evento, de esta forma nos aseguramos que la transmisión se realizará correctamente.

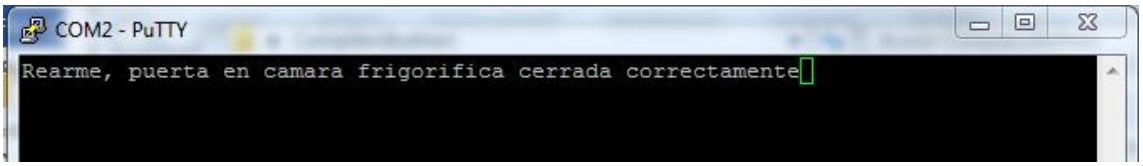
Item Tree | Task definitions | Script: nxaDefinitions.lua* | Script: nxaCOMExample.lua

```
--nxa.SetValue(itemID, value [,delay_in_ms])
--nxa.GetValue(itemID [,defaultvalue]) As value
--nxa.WriteValue(itemID, value [,delay_in_ms])
--nxa.ReadValue(itemID [,delay in ms])


function COIL0Notification ()
--nxa.SetValue("NETx.XCON.<COM3>.OUT","MUNDO")
--configura comunicaciones

nxa.SetValue("NETx.XCON.COM.Config.Device","COM3")
nxa.SetValue("NETx.XCON.COM.Config.Baudrate","9600")
nxa.SetValue("NETx.XCON.COM.Config.DataBits","8")
nxa.SetValue("NETx.XCON.COM.Config.Parity","0")
nxa.SetValue("NETx.XCON.COM.Config.StopBits","1")
--resto de parámetros por defecto

--Habilita comunicaciones
nxa.SetValue("NETx.XCON.COM.Enabled","1")

--Envía mensaje COIL1

if (nxa.GetValue("NETx\\XIO\\Modbus\\MB1\\Coils\\0")==true) then--si se activa la alarma...
    nxa.SetValue("NETx.XCON.COM.OUT","Alarma puerta abierta en camara frigorifica")--COIL0=1
else nxa.SetValue("NETx.XCON.COM.OUT","Rearme, puerta en camara frigorifica cerrada correctamente")--COIL0=0
    end
end
```

Con una configuración como la del ejemplo anterior, los mensajes que se mostrarían serían como sigue:



Si se cierra la puerta, se vuelve a cambiar el estado del coil, con lo cual se envía lo siguiente:



SCRIPT

---

function COIL0Notification ()

--nxa.SetValue(Item ID,Valor)

--nxa.GetValue(Item ID)

--configura comunicaciones

nxa.SetValue("NETx.XCON.COM.Config.Device","COM3")

```
nxa.SetValue("NETx.XCON.COM.Config.Baudrate","9600")

nxa.SetValue("NETx.XCON.COM.Config.DataBits","8")

nxa.SetValue("NETx.XCON.COM.Config.Parity","0")

nxa.SetValue("NETx.XCON.COM.Config.StopBits","1")

--resto de parámetros por defecto


--Habilita comunicaciones

nxa.SetValue("NETx.XCON.COM.Enabled","1")


--Envía mensaje COIL1


if (nxa.GetValue("NETx\\XIO\\Modbus\\MB1\\Coils\\0")==true) then--si se activa la alarma...

        nxa.SetValue("NETx.XCON.COM.OUT","Alarma puerta abierta en camara frigorifica")--
COIL0=1

else nxa.SetValue("NETx.XCON.COM.OUT","Rearme, puerta en camara frigorifica cerrada
correctamente")--COIL0=0

        end

end
```