Indice

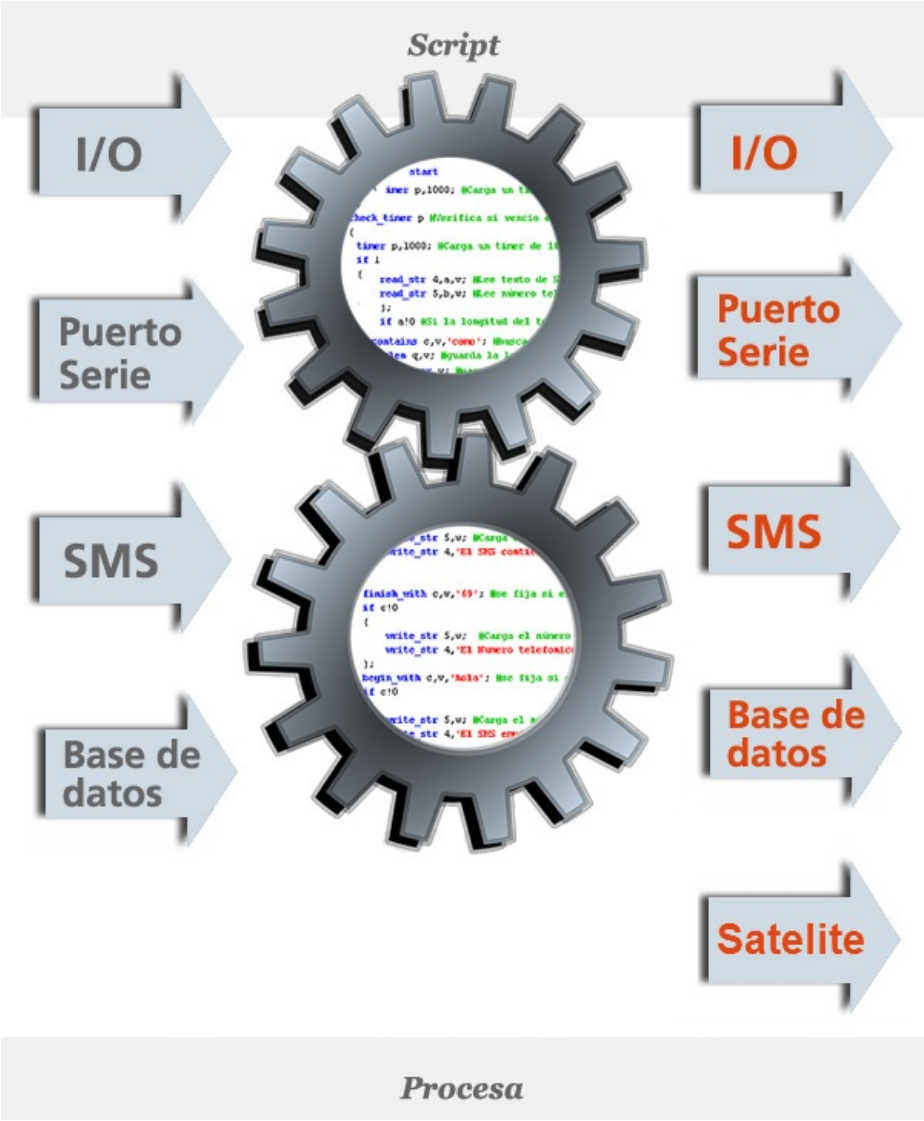
Descripción

Los GRD/cLAN permiten correr en el equipo programas escritos por el usuario, haciéndolos más flexibles y potentes.

El GRD/cLAN continuará funcionando normalmente mientras corre el script cargado en su memoria.

Operaciones que se pueden realizar desde un script

• Operaciones **Matemáticas**
• Operaciones **Lógicas**
• Operaciones de **Temporizado**
• Lectura de las entradas propias del equipo y de variables Modbus
• Encendido y Apagado de salidas digitales
• Envío y recepción de **SMS** (Solo GRD)
• Interpretación de datos del **Puerto serie**
• Envío de datos por modem **Satelital** externo



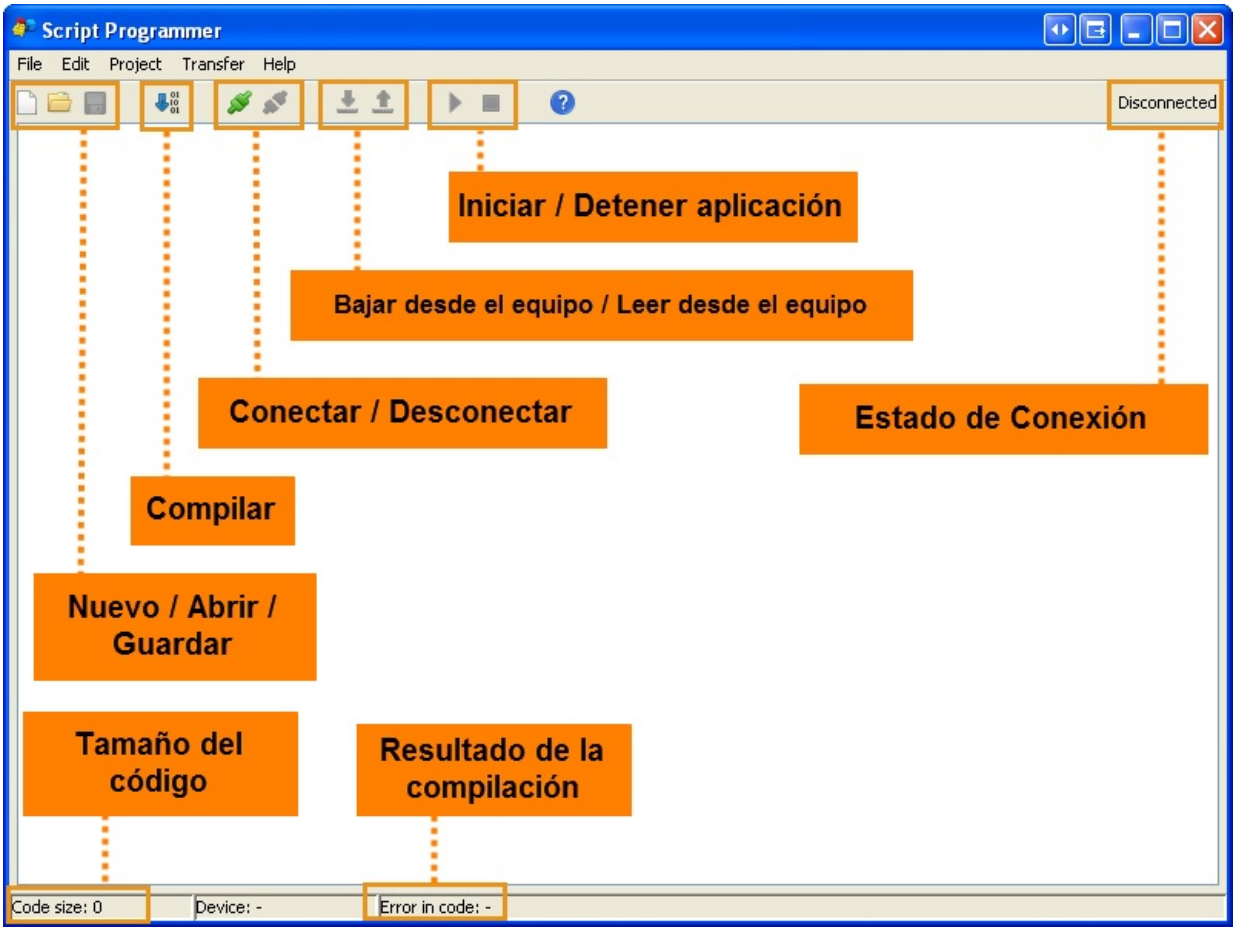
2017-03-27

Introducción

Este programa permite desarrollar, compilar y transferir el script al GRD/cLAN. Para utilizarlo, primero tenemos que asegurarnos de que el "GRDconfig" funciona correctamente y podemos comunicarnos con el GRD/cLAN.

Descripción del Software

A continuación se ve una descripción de las funciones de la pantalla principal.



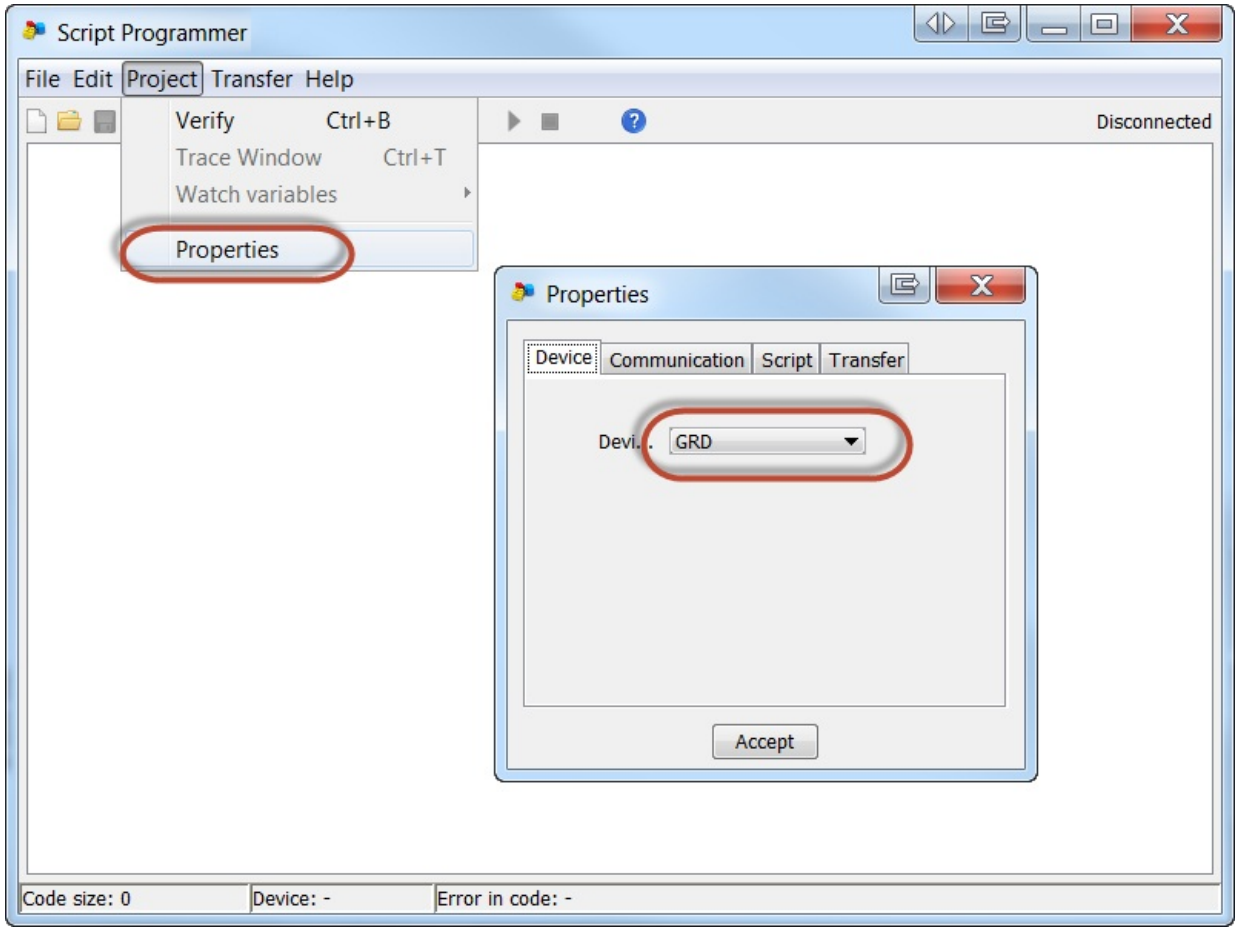
Conexión con el Equipo

Existen dos formas de conectarse, de manera local (USB en el GRD, LAN/Ethernet en el cLAN) y remota (a través del Middleware)

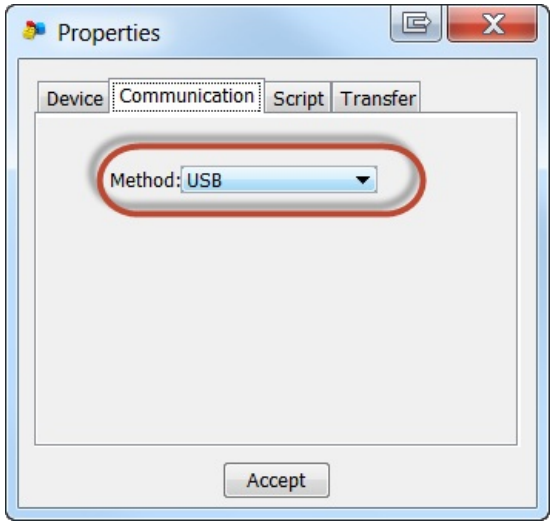
Conexión con el Equipo - GRD por USB

Primero debe conectar el GRD a la PC y asegurarse de que este desconectado del ***"GRDconfig"***

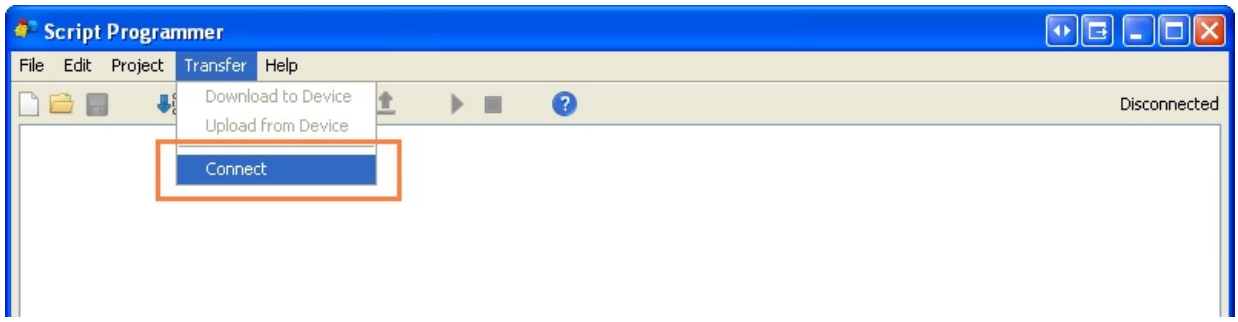
Luego ir al menú ***"Project"***, opción ***"Properties"*** y elegir "GRD" en la pestaña "Device"



Elegir "USB" en la pestaña ***"Communication"***



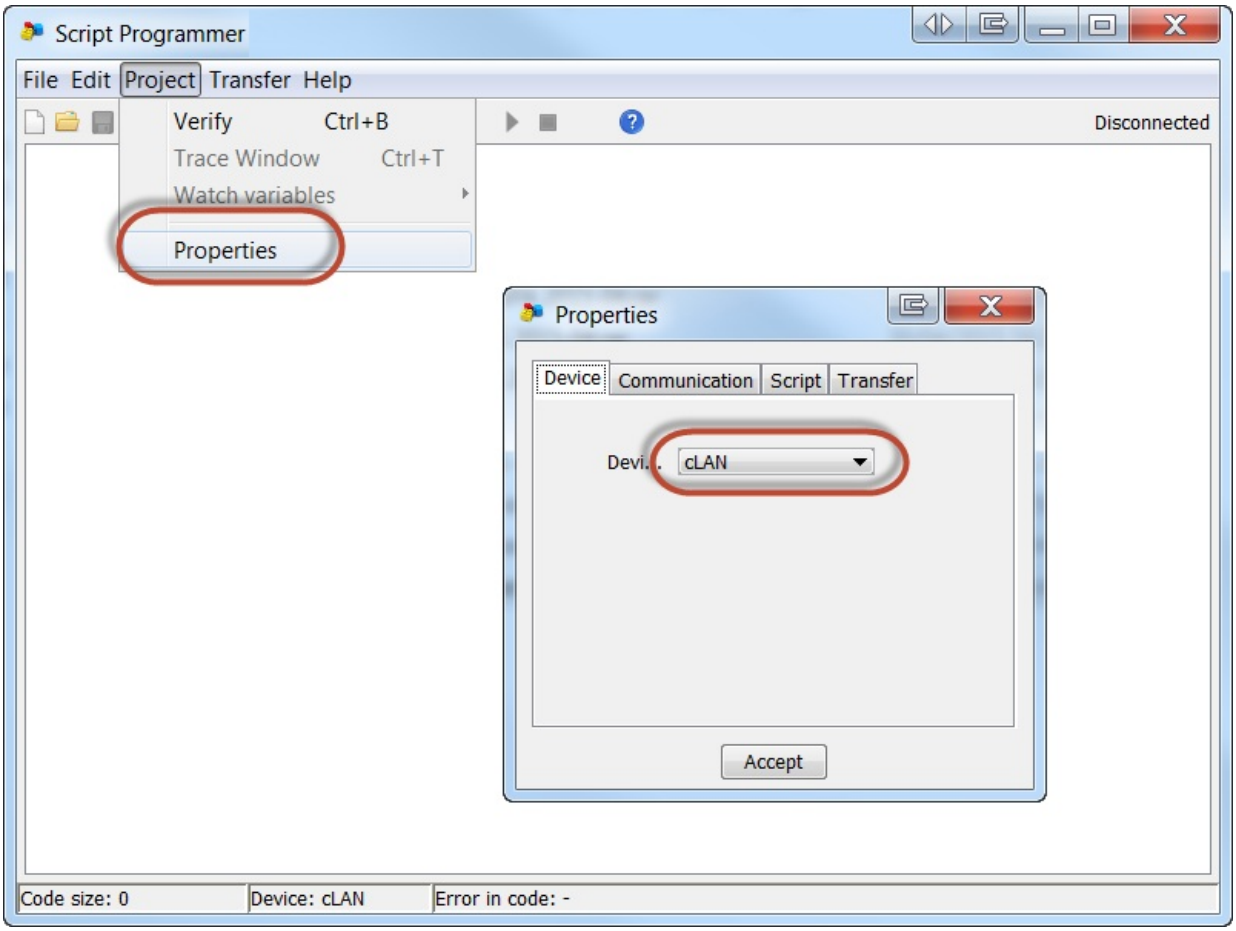
Ir al menú ***"Transfer"*** y hacer clic en ***"Connect"***.



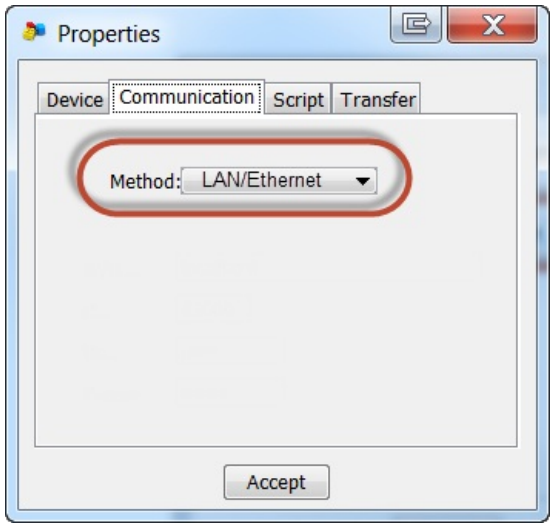
Conexión con el Equipo - cLAN por LAN/Ethernet

Primero debe conectar el cLAN a la misma red que la PC y asegurarse de que tenga configurada una dirección IP válida como se indica en su manual del usuario.

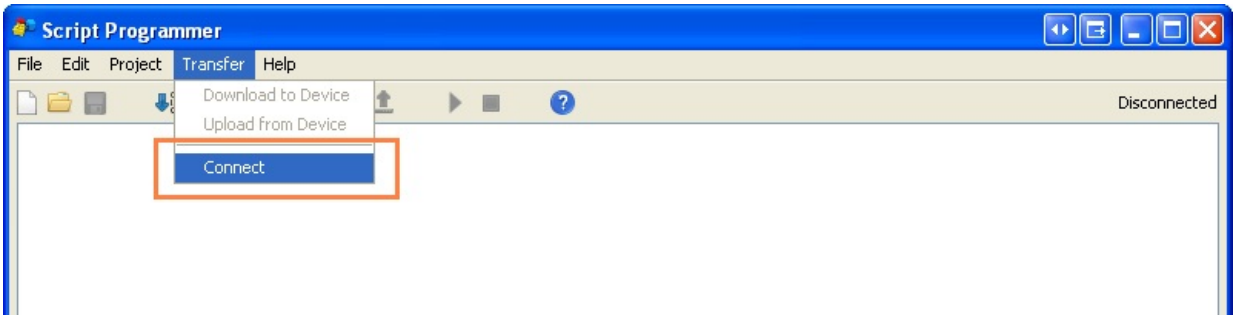
Luego ir al menú ***"Project"***, opción ***"Properties"*** y elegir "cLAN" en la pestaña "Device"



Elegir "LAN/Ethernet" en la pestaña ***"Communication"***

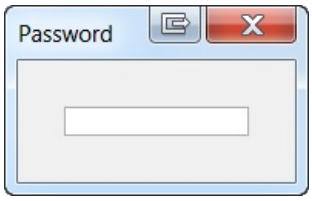


Ir al menú ***"Transfer"*** y hacer clic en ***"Connect"***.



Al hacerlo deberá ver el/los cLAN conectados a su red. Seleccione el que quiera configurar.

Para conectarse deberá escribir la contraseña del cLAN. Es la misma que usa para conectarse el MW.

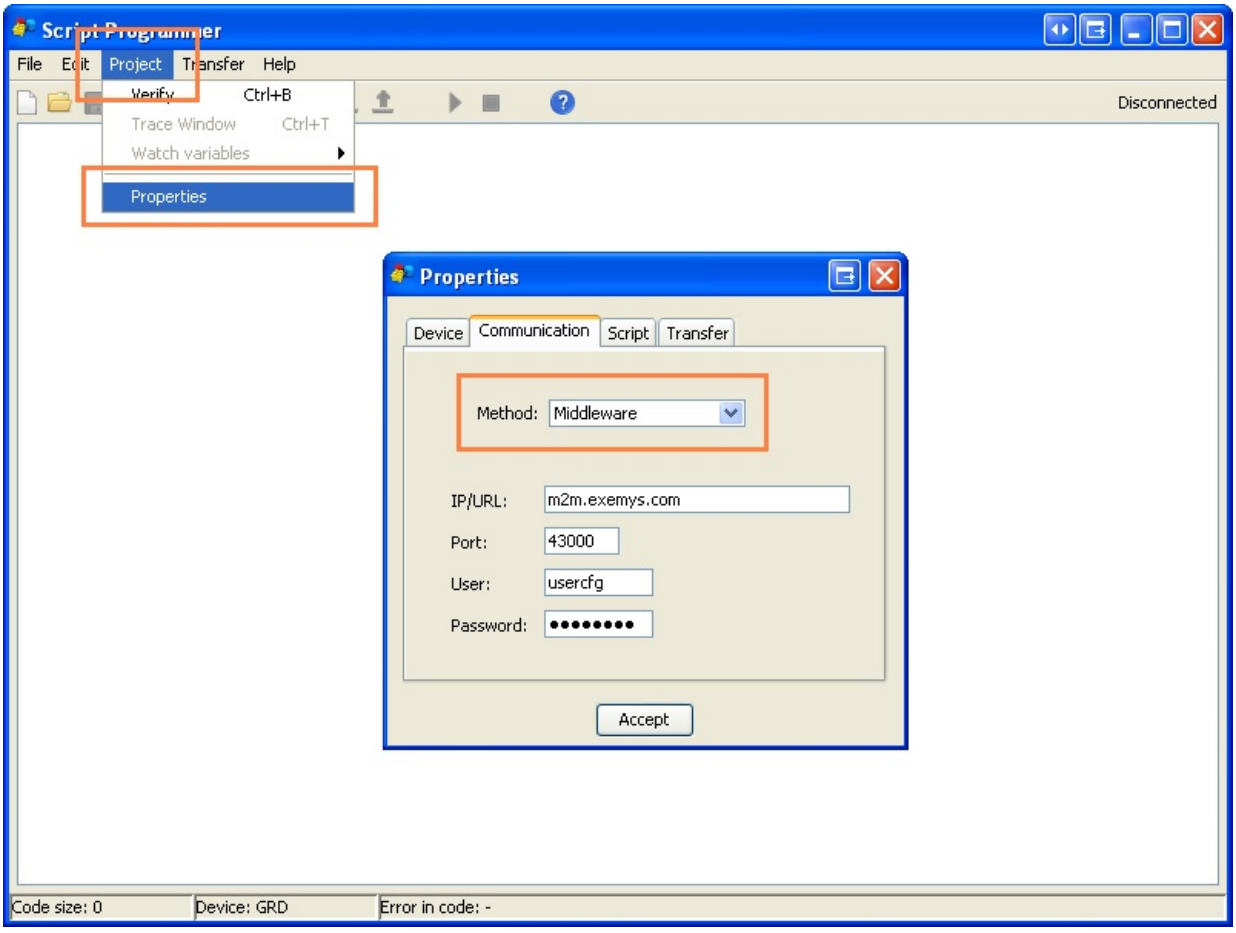


Conexión con el Equipo - cLAN-XF/GRD-XF de modo remoto (solo modelos XF)

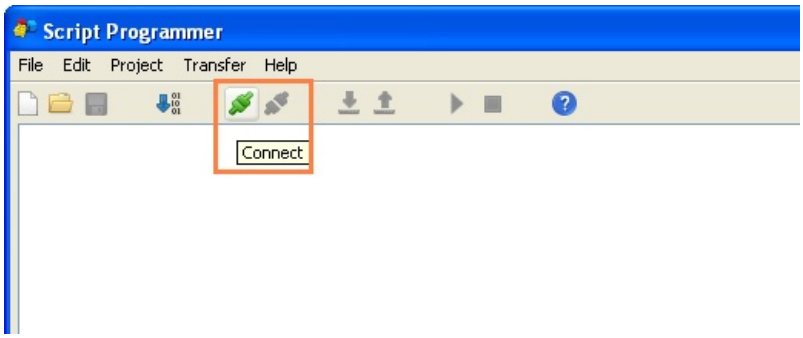
El equipo debe estar conectado al Middleware para poder configuralo en forma remota.

La versión del Middleware debe ser 4.2.0 o superior para soportar carga/descarga de scripts remota.

Para utilizar la transferencia vía Middleware debemos configurar la IP/URL, puerto, usuario y contraseña de configuración remota del mismo, para realizarlo debemos ir al menú ***"Project"***, luego la opción ***"Properties"***, seleccionar la sección ***"Communication"*** y seleccionar el ***"Method"*** Middleware



Luego de seleccionar el método de comunicación puede conectarse en cualquier momento presiona el botón de la barra de herramientas



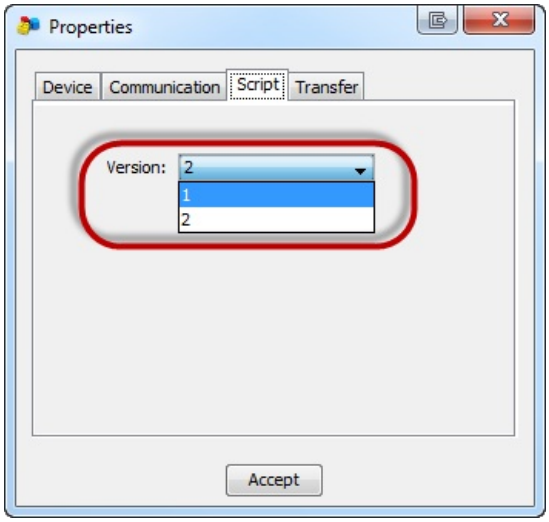
Al intentar conectarse se verá esta ventana donde deberá seleccionar el equipo.

Versiones de Script 1 y 2

En el menú ***"Project"***, opción ***"Properties"***, pestaña ***"Script"*** puede seleccionar entre la versión 1 y la 2 del script.

Solo los GRD-XF-2G y cLAN V1.x funcionan con versión 1. El resto funciona con versión 2.

La versión 2 se diferencia de la 1 porque permite el doble de variables ya que pueden ir en minúscula o mayúscula.



La versión de script será usada por el Script Programmer en dos situaciones, cuando **verifique** un script o cuando trate de **enviarlo** al equipo.

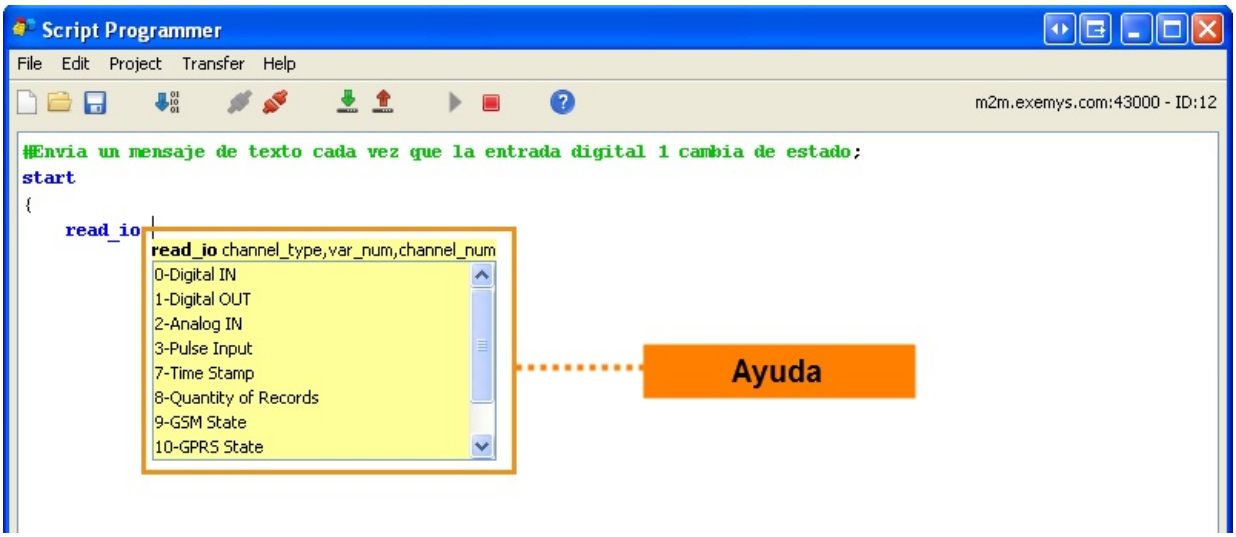
Si al enviar un script la versión seleccionada no es compatible con el equipo destino verá un mensaje indicando ese error

Carga y descarga de scripts

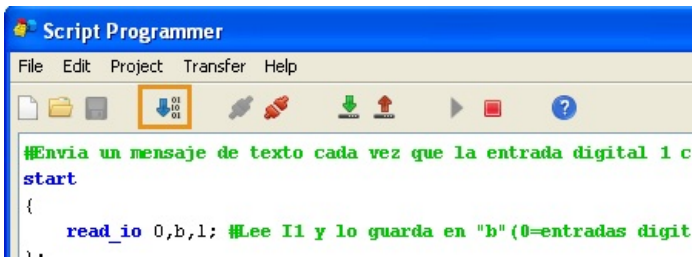
Una vez conectados al equipo podremos transferir y descargar los scripts.También de podra detener y poner en marcha.

Edición de scripts

Para desarrollar un programa solo debemos escribir el código en el panel de edición, el entorno cuenta con ayuda en la escritura de las funciones y las resalta indicando su correcta escritura.

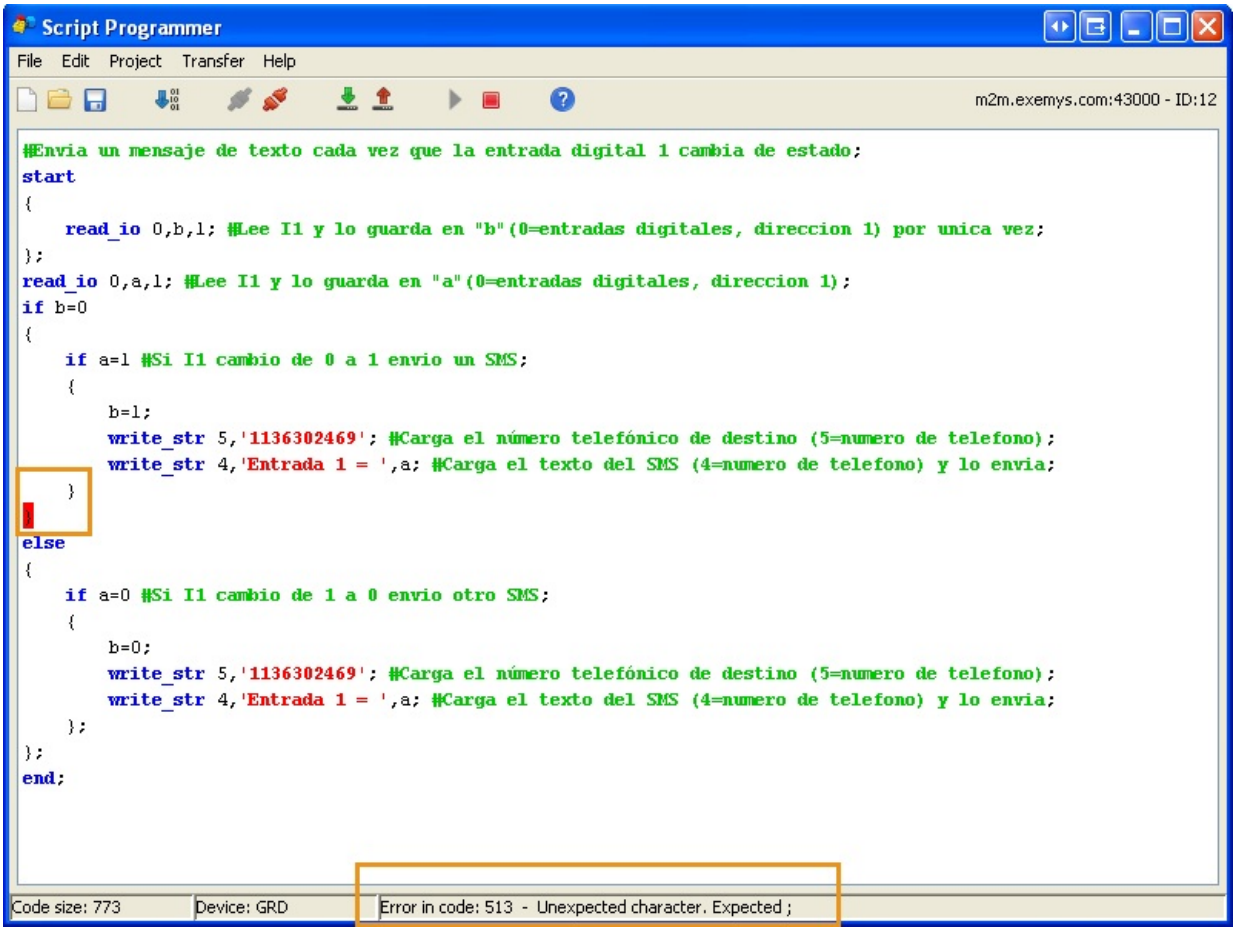


Una vez que terminamos de escribir el código con el botón ***"Verify"*** compilamos el programa y así podremos ver si este tiene algún error de sintaxis.



Al momento de compilar en la parte inferior nos indicará si el mismo tiene o no errores, si hay un error se marcará en rojo la línea en donde se encuentra y en la casilla ***"Error in Code:"*** nos dirá la línea, si no hay aparecerá un cartel indicándolo y en ***"Error in Code: NONE"***.

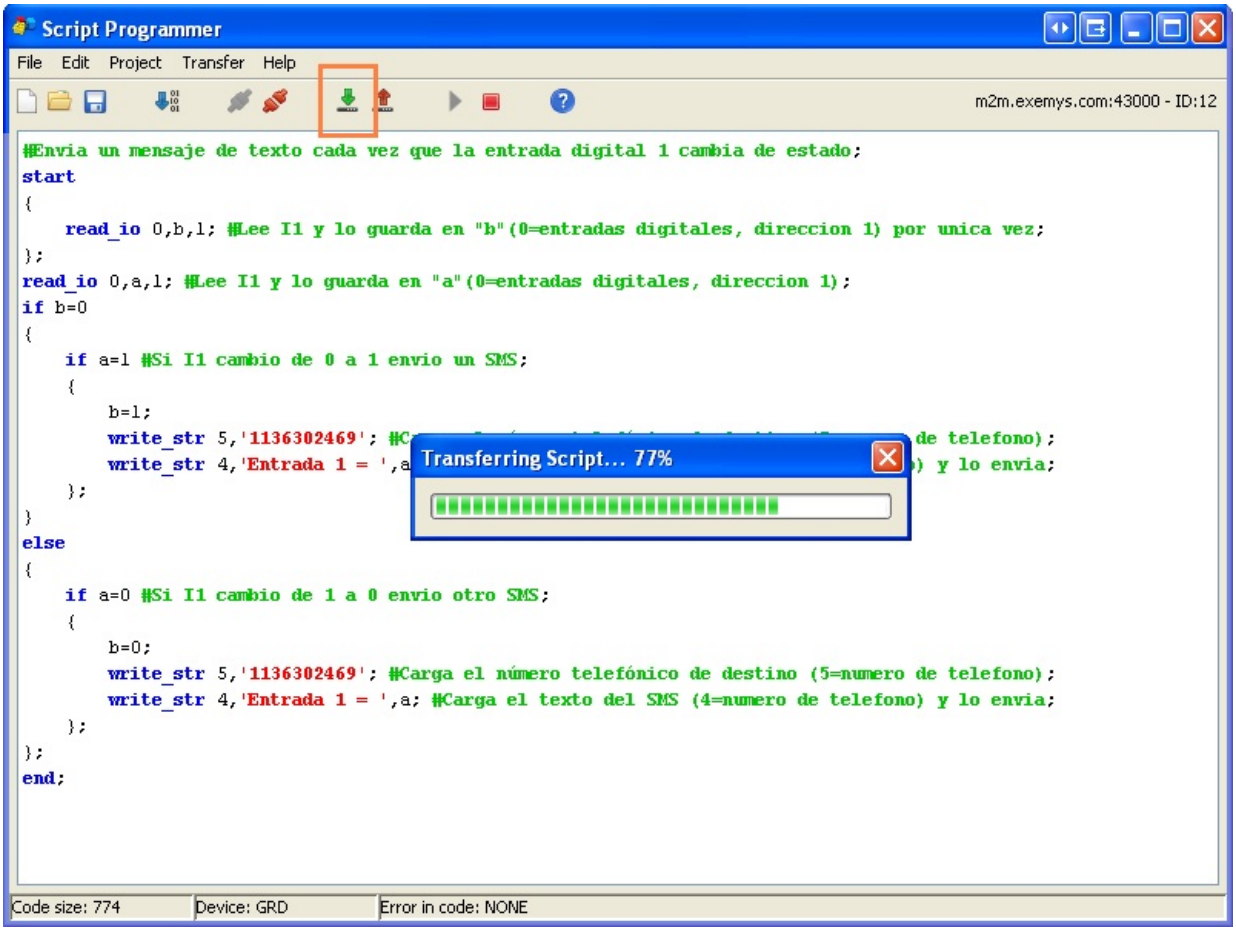
En la siguiente imagen vemos un programa con un error, en este caso falta un ***";"***.

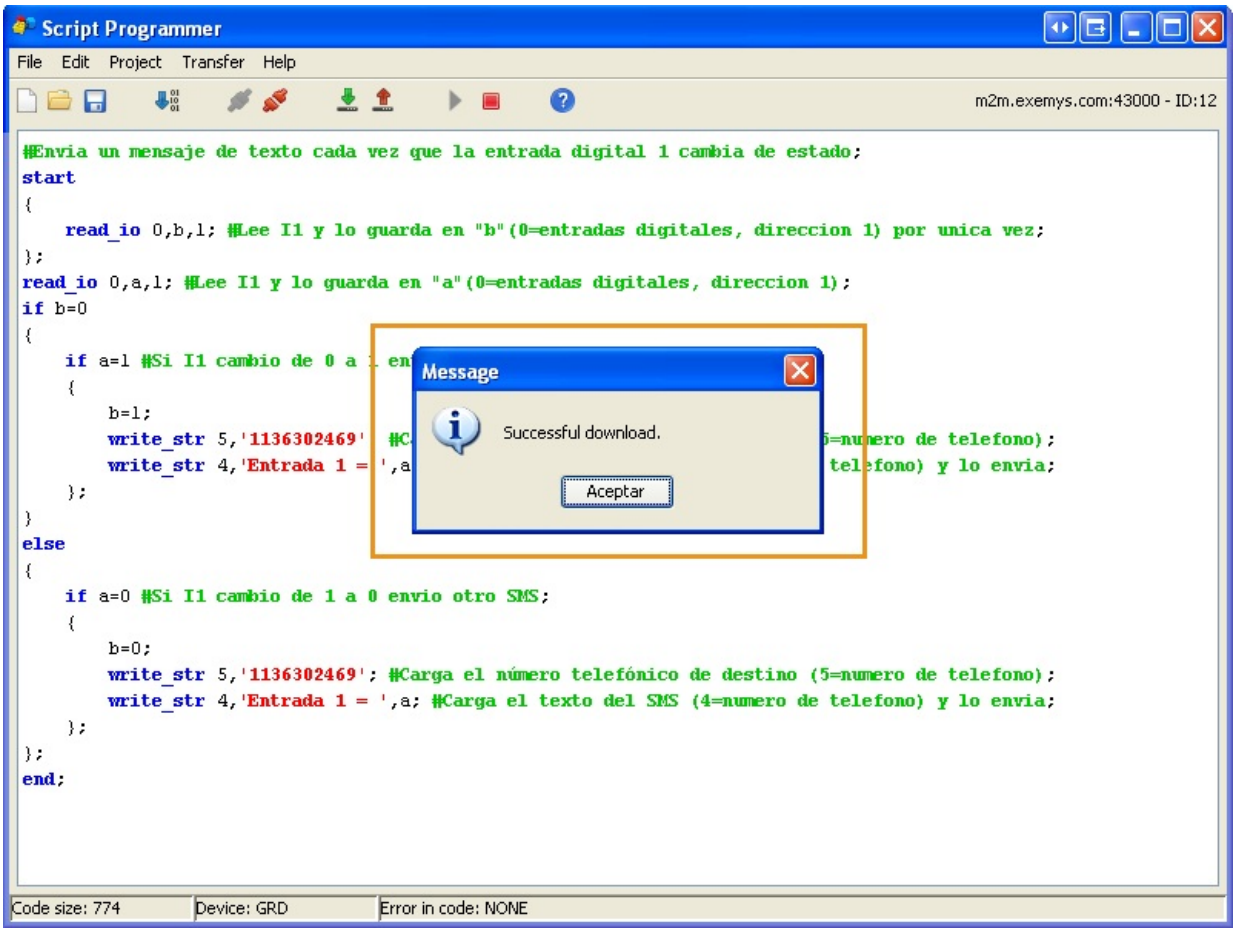


El punto y coma falta en la llave anterior a la que esta marcada en rojo, esto se debe a que el compilador nos indica que encontro un caracter que no es el esperado.

En la siguiente imagen vemos un programa sin errores.

Si no tiene errores podemos, transferir el programa al equipo haciendo clic en ***"Download to device".*** Aparecerá una ventana que nos indica el estado la descarga y luego un cartel que dice si la descarga fue exitosa o no.





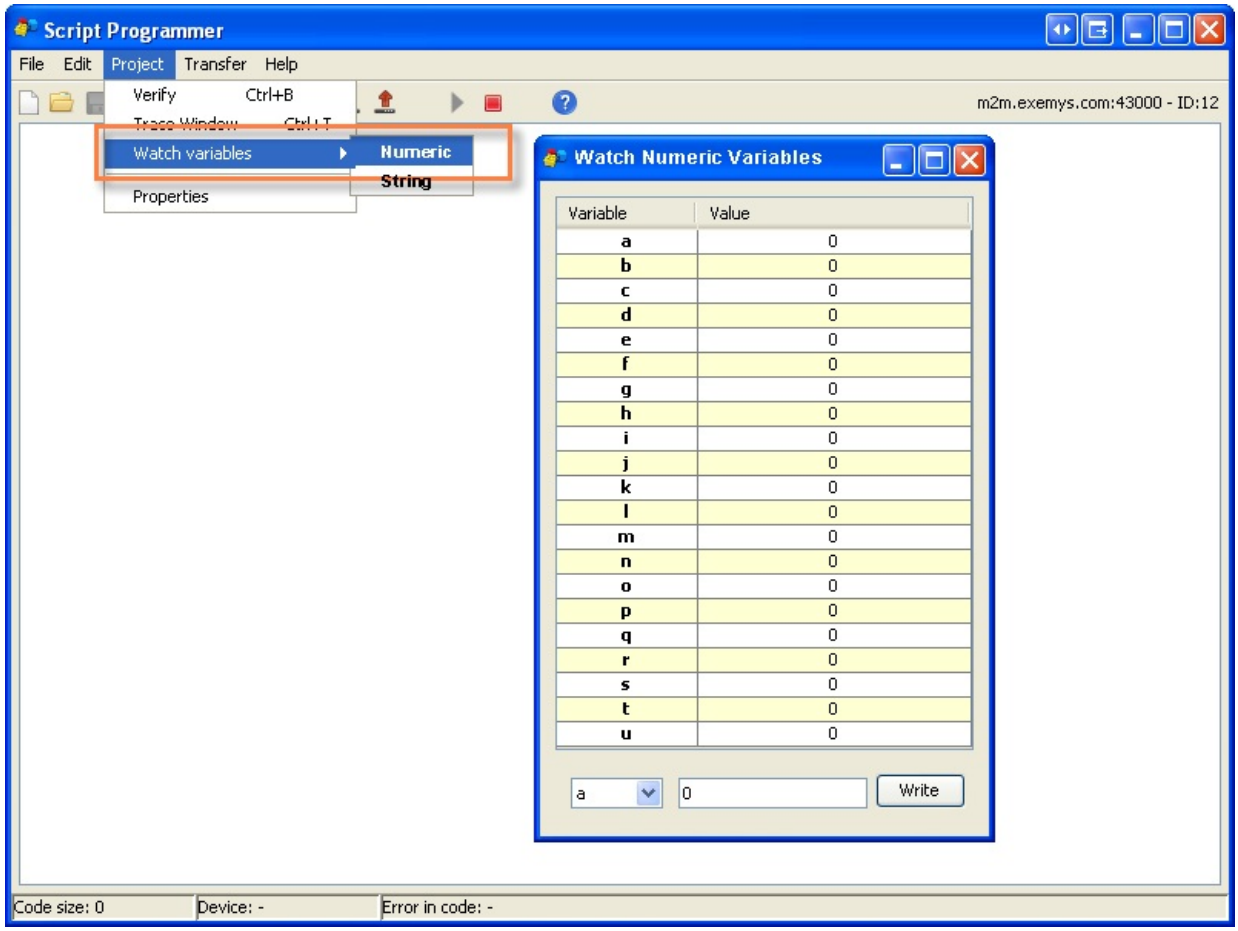
[Depuración de scripts]

El Script Programmer dispone de dos herremientas para la depuración de los scripts escritos. Losl GRD deben tener la versión de firmware 5.2.0 o superior para soportar estas opciones. El cLAN siempre tiene estas opciones disponibles.

Monitoreo de variables

Con esta herramienta puede ver el valor de las variables númericas o de tipo texto mientras el programa está corriendo. Tambíen puede modificar el valor de las variables para simular condiciones de trabajo del script.

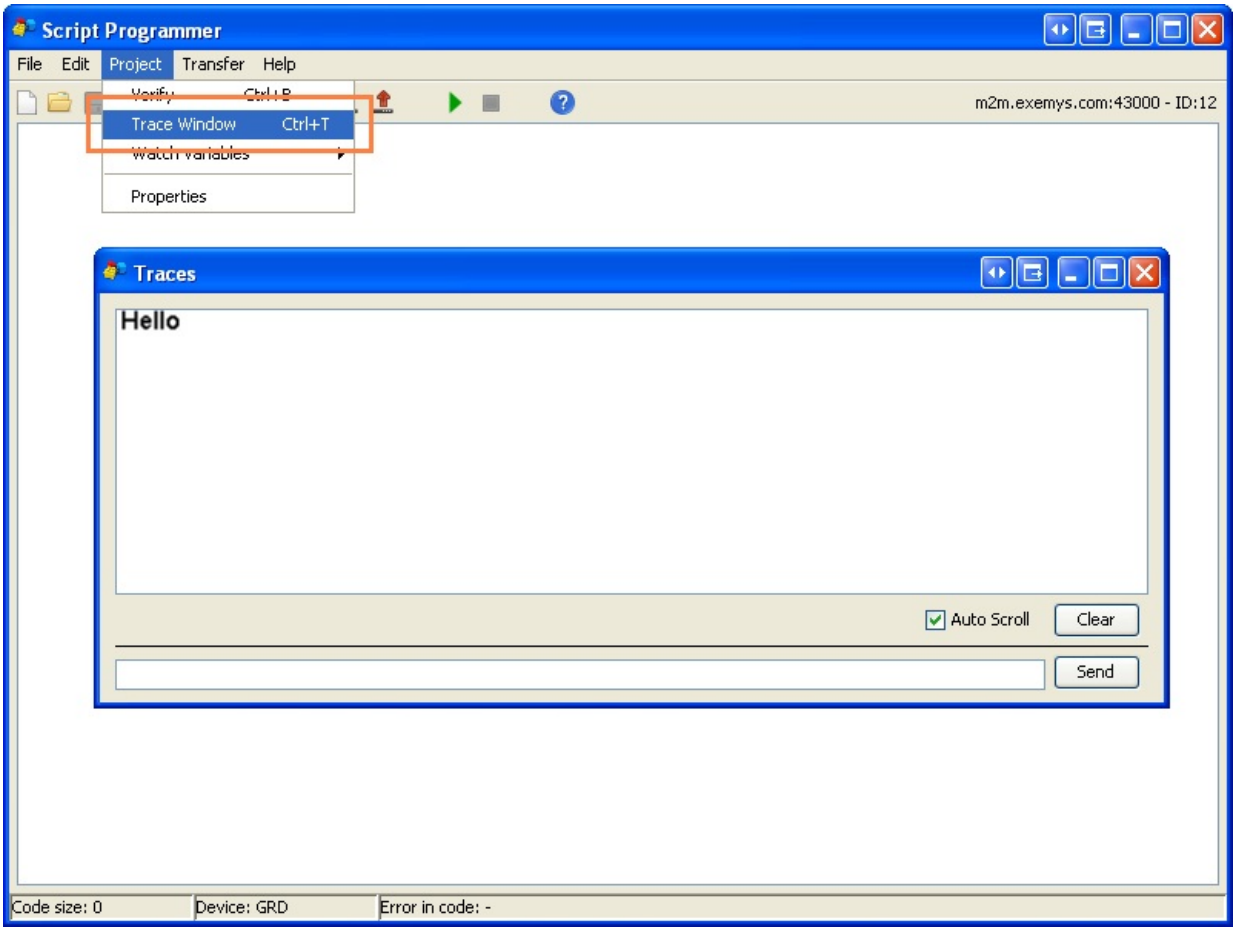
Una vez que este conectado al equipo vaya al menú **"Project",** opción **"Watch variables"** y luego **"Numeric"** o **"String"** según el tipo de variable a monitorea



Envío y recepción de trazas

Con esta herramienta puede enviar textos desde el script al Script Programmer para seguir el funcionamiento del script. Tambíen pueden enviar textos simular condiciones de trabajo del script.

Una vez que este conectado al equipo vaya al menú **"Project",** opción **"Trace Window"**



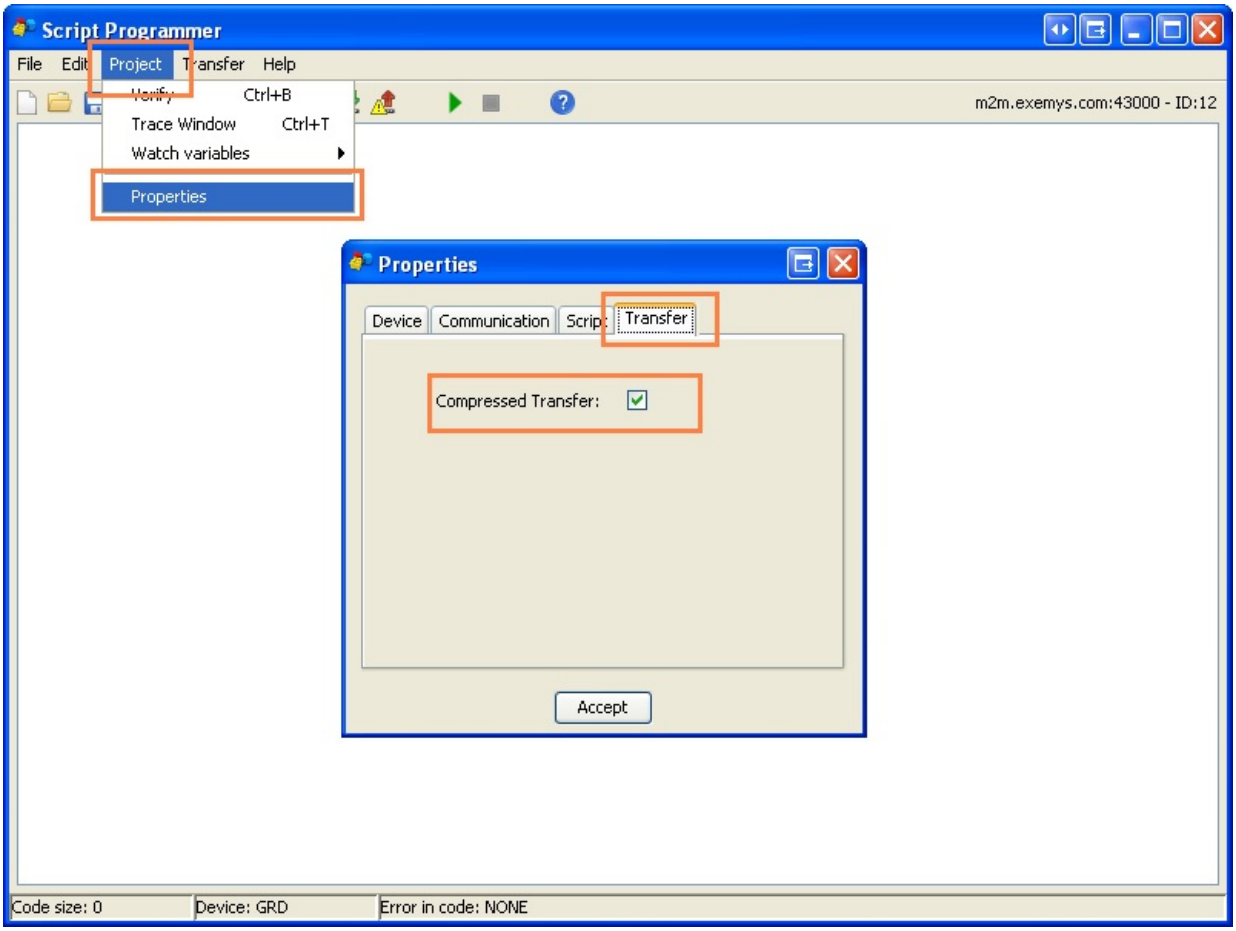
Compresión de scripts

El equipo tiene un espacio limitado para guardar el script. El máximo son 20000 caracteres.

Si este espacio no es suficiente para su aplicación puede usar la opción de compresión de scripts. Al hacerlo el Script Programmer quitará todos los comentarios y tabulaciones antes de enviar el programa.

Si quiere conservar los comentarios de su programa guarde una copia en su computadora.

Para habilitar la compresión vaya al menú **"Project"**, opción **"Properties"**, sección **"Transfer"** y habilite la opción **"Compressed Transfer"**


Script Programmer
File Edit Project Transfer Help
Verify Ctrl+B
Trace Window Ctrl+T
Watch variables
Properties
m2m.exemys.com:43000 - ID:12
Properties
Device Communication Script Transfer
Compressed Transfer:
Accept
Code size: 0
Device: GRD
Error in code: NONE

2020-11-06

Introducción

El lenguaje de programación ***Exemys Script*** es de tipo bucle, esto quiere decir que se ejecuta todo el código hasta la última línea y luego vuelve a comenzar.

No hay sentencias para crear bucles dentro del programa, por lo que no se puede detener el flujo del programa en una sección de código. Se asemeja en este sentido a la programación de PLCs, aunque su sintaxis es mas cercana al C o a Pascal.

Además de la lectura de este manual, recomendamos la lectura de scripts de ejemplo. Puede descargar ejemplos desde este link www.exemys.com/EjemplosDeScriptEnGRD

Todos estos ejemplos también funcionan en los cLAN, excepto lo que usan SMS.

Versiones 1 y 2 del script

Existe 2 versiones de script. La 1 y la 2. La versión 2 se diferencia de la 1 porque permite el doble de variables ya que pueden ir en minúscula o mayúscula. En la versión 1 las variables solo se escriben en minúscula.

Los equipos GRD-XF-2G y cLAN V1.x funcionan con versión 1

Los equipos GRD-XF-3G y cLAN V2.x funcionan con versión 2

Todos los GRD-MQ y cLAN MQ funcionan con versión 2.

Variables

Se dispone de 2 tipos de variables, numéricas del tipo ***"long"*** y de texto del tipo ***"string".***

No es necesario definir las variables ya que se tiene una cantidad fija.

En la **versión 1** de script las variable numéricas son 21, de la ***"a"*** a la ***"u"***. Las de texto son 5, de la ***"v"*** a la ***"z"***, con una longitud máxima de 100 caracteres cada una.

En la **versión 2** de script las variable numéricas son 42, de la ***"a"*** a la ***"u"*** y de la ***"A"*** a la ***"U"*** Las de texto son 10, de la ***"v"*** a la ***"z"*** y de la ***"V"*** a la ***"Z"*** y con una longitud máxima de 100 caracteres cada una.

Como las variables numéricas son del tipo "long" cualquier operación que arroje un resultado con decimales se vera truncado.

El valor inicial de las variables numéricas es 0, en las de texto es un string vacío.

Las variables numéricas, pueden "mapearse" para poder ser leídas de algún modo. En el GRD pueden vincularse de los canales de entradas y salidas.

Asignar valor a una variable:

- Variables numéricas:

```
a = 652;
```

- Variables de texto:

```
v = 'Hola mundo'
```

Puede observarse que para cargar texto es necesario colocarlo entre comillas simples.

Concatenación de strings:

Para concatenar variables solamente debemos colocar una a continuación de la otra, separándolas por comas.

Por ejemplo:

```
a = 20;

u = 'La temperatura es de ';

v = ' °C';
```

Si queremos formar la frase 'La temperatura es de 20 °C ' y guardarla en otra variable hacemos lo siguiente:

```
w = u, a, v;
```

Otra forma de hacerlo sería:

```
w = 'La temperatura es de ', a,' °C';
```

La concatenación solo puede hacerse en asignación de textos a variables string y en la función write_str

Inserción de valores ASCII en strings:

Si se quiere insertar un valor ASCII en string se puede usar el operador $. Después del operador se debe indicar al código ASCII en decimal. No se permite el ASCII 0.

Por ejemplo:

```
z = 'Hola mundo',$13,$10;
```

La inserción de valores ASCII solo puede hacerse en asignación de textos a variables string y en la función write_str

Operadores aritméticos

A continuación veremos los operadores que se pueden utilizar en el script, debemos tener en cuenta que solo toma valores del tipo “long” por lo que cualquier operación que arroje resultados con decimales se verá truncada, o sea que solamente obtendremos como resultado la parte entera.

Operador	Significado
=	Igual
^	Exponencial
|	Or
&	And
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resto

Por ejemplo:

```
a = 130;
b = a+5;
```

Resultado: La variable *b* vale 135.

Estructura de un programa

Todas las instrucciones finalizan con punto y coma ***";"***.

Es muy importante tener en cuenta que el programa se ejecuta de manera que cuando se alcanza la ultima sentencia vuelve a comenzar, debido a esta particularidad disponemos de una sentencia llamada "***start"*** que se ejecuta solo una vez cuando el programa se inicia. Dentro de este bloque podremos cargar valores iniciales de variables o todo aquello que consideremos adecuado.

Todo programa debe terminar con la instrucción ***"end;"***.

A continuación veremos la estructura típica de un programa:

```
start
{
    a = 10;
};
a = a + 1;
end;
```

Este simple ejemplo muestra como se utiliza la función ***"start"*** para iniciar por única vez a la variable ***"a"*** con el valor 10 y luego incrementarse en 1 constantemente.
Como podemos ver todas las instrucciones que se encuentran después del bloque ***"start"*** y antes del ***"end;"*** se ejecutaran de manera cíclica.

Comentarios:

En caso de querer agregar un comentario al programa se de anteponer al texto el carácter ***"#"*** y finalizarlo con ***";"***.

Por ejemplo:

```
start
{
    a = 10;  #Inicia variable a = 10;
};
a = a + 1;    #Incrementa variable a;
end;
```

Funciones de Control de flujo

El control de flujo especifica el orden en que se van a llevar a cabo ciertas líneas del código.
En nuestro caso tenemos 3 funciones para realizar dicho control, el ***"start",*** el ***"if-else"*** y el ***"end"***.

Función *"start"*

Se utiliza para ejecutar líneas de código por única vez en el momento que el programa se inicia.
Las instrucciones a ejecutar se colocan entre llaves y luego de la de cierre se debe colocar ***";"***.

Sintaxis:

```
start
{
```

```
...;
...;
};
```

Función *"if-else"*:

Esta función se utiliza para la toma de decisiones, si la condición colocada luego del ***"if"*** es valida se ejecutaran las instrucciones que estén entre llaves debajo de este en caso contrario se ejecutaran las que estén dentro del ***"else"***.

La condición de ejecución puede utilizar los siguientes operadores:

Operador	Significado
=	Igual
!	Distinto
>	Mayor
<	Menor

La condición se escribe dejando un espacio después del ***"if"***

Sintaxis:

Solo ***"if"***:

```
if condición
{
   ...;
   ...;
};
```

Luego de la llave de cierre del ***"if"*** debe colocarse ***";"***.

"if-else":

```
if condición
{
   ...;
   ...;
}

else
{
   ...;
   ...;
};
```

En este caso el ***";"*** se coloca después de la llave de cierre del ***"else"*** y no después del ***"if"***.

Función *"end"*

Solamente se utiliza para indicar el fin del programa luego de ejecutarla se vuelve a la primera línea del código.

Sintaxis:

```
end;
```

Funciones de interfaz

Estas funciones permiten tomar datos de distintas fuentes como también enviar datos a distintos destinos.

Función *"read_io"*

Permite leer parámetros indexados de distintas fuentes, como los valores de canales de entradas digitales, salidas digitales, entradas analógicas, etc.

Se debe indicar la ***"fuente"*** del dato con un número. En caso de que corresponda, con el parámetro ***"índice"***, le decimos la posición dentro de esa fuente.
El resultado de la lectura se guarda en una variable numérica.

Sintaxis:

```
read_io fuente,variable_numerica,indice;
```

Las fuentes disponibles dependen del **equipo** donde se corre el script y la versión de script. En un futuro pueden agregarse nuevas fuentes.
Refiérase a la seeción fuentes/destinos para mas detalles.

Función *"write_io"*

Permite escribir parámetros indexados en distintos destinos, como los valores de canales de salidas digitales, de pulsos, etc.

Se debe indicar el ***"destino"*** donde escribir con un número. En caso de que corresponda, con el parámetro ***"índice"***, le decimos la posición dentro de ese destino.

El ***"valor"*** a escribir puede ser un número o una variable numérica.

Sintaxis:

```
write_io destino,indice,valor;
```

Los destinos disponibles dependen del **equipo** donde se corre el script y la versión de script. En un futuro pueden agregarse nuevos destinos.
Refiérase a la seeción fuentes/destinos para mas detalles.

Función *"read_str"*:

Permite leer strings originados en distintas fuentes, como lo son los datos recibidos por un puerto serie o por SMS. Para realizar esto se debe indicar la ***"fuente"*** con un número.

El resultado de la lectura se guarda en dos variables, una del tipo string y una del tipo numérica, donde se indica la longitud del string. Si no hay datos este parámetro dará 0.

Sintaxis:

```
read_str fuente,variable_numerica,variable_string;
```

Las fuentes disponibles dependen del **equipo** donde se corre el script y la versión de script. En un futuro pueden agregarse nuevas fuentes.
Refiérase a la seeción fuentes/destinos para mas detalles.

Función *"write_str"*:

Permite escribir strings en distintos “destino”, como el puerto serie o enviar SMS. Para realizar esto se debe indicar el ***“destino”*** con un número y el string a enviar.

Sintaxis:

```
write_str destino,string;.
```

Los destinos disponibles dependen del **equipo** donde se corre el script y la versión de script. En un futuro pueden agregarse nuevos destinos.
Refiérase a la seeción fuentes/destinos para mas detalles

El string puede ser una variable o un string escrito en la función. **Esta función permite concatenación e inclusión de valores ASCII.**

Funciones de string

Estas funciones permiten realizar distintas operaciones con las variables del programa del tipo string.

Función *“is_equal”*

Compara una variable string con un string (texto fijo o variable string). Devuelve 0 si son distintos y 1 si son iguales.

Sintaxis:

```
is_equal variable_numerica,variable_string,string;
```

Ejemplo:

```
v='APAGAR BOMBA';
is_equal c,v,'APAGAR BOMBA';
if c=1 {
        #Los textos son iguales;
};
```

Función *“finish_with”*

Determina si un variable string termina con un string en particular (texto fijo o variable string). Devuelve 0 si es falso o 1 si verdadero

Sintaxis:

```
finish_with variable_numerica,variable_string,string;
```

Ejemplo:

```
v='APAGAR BOMBA';
finish_with c,v,'BOMBA';
if c=1 {
        #El string guardado en v termina en 'BOMBA';
};
```

Función *“begin_with”*

Determina si un variable string comienza con un string en particular (texto fijo o variable string). Devuelve 0 si es falso o 1 si verdadero

Sintaxis:

begin_with *variable_numerica,variable_string,string;*

Ejemplo:

```
v='APAGAR BOMBA';
begin_with c,v,'APAGAR';
if c=1 {
        #El string guardado en v empieza con 'APAGAR';
};
```

Función *"contains"*:

Determina si una variable string contiene un string en particular (texto fijo o variable string). Devuelve 0 si no lo encuentra o la posición donde lo encontró si lo encuentra.

Sintaxis:

contains *variable_numerica,variable_string,string;*

Ejemplo:

```
v='APAGAR BOMBA';
contains c,v,'GAR';
if c>0 {
        #El string guardado en v contiene el texto 'GAR';
};
```

Función *"upper"*

Convierte todos los caracteres de una variable string a mayúsculas, guardando el resultado en la misma variable.

Sintaxis:

upper *variable_string;*

Ejemplo:

```
v='Apagar';
upper v;
#Ahora v es igual a 'APAGAR';
```

Función *"lower"*

Convierte todos los caracteres de una variable string a minúscula, guardando el resultado en la misma variable.

Sintaxis:

lower *variable_string;*

Ejemplo:

```
v='Apagar';
lower v;
#Ahora v es igual a 'apagar';
```

Función *"strlen"*

Devuelve en una variable numérica, la longitud de una variable *string*.

Sintaxis:

```
strlen variable_numerica,variable_string;
```

Ejemplo:

```
v='Apagar';
strlen c,v;
#Ahora c vale 6;
```

Función *"substr"*

Devuelve una parte de una variable string dentro de la misma variable

Sintaxis:

```
substr inicio,fin,variable_string;
```

Ejemplo:

```
v='APAGAR BOMBA';
substr 2,3,v;
#Ahora v vale 'PA';
```

Funciones de conversión

Estas funciones convierten variables de un tipo a otro.

Función *"point"*

Coloca el punto decimal a una variable numérica y la convierte a *string*.
Como parámetros se le pasa, la variable numérica, la variable string y la cantidad de decimales que queremos colocarle.

Sintaxis:

```
point variable_string,variable_numerica,decimales;
```

Ejemplo:

```
c=123;
point v,c,1;
#Ahora v vale '12.3';
```

Función *"aton"*

Convierte un número dentro de string a una variable numérica. Empieza en el principio del string y termina en donde encuentra un valor no numérico o el fin del string.

Sintaxis:

```
aton variable_numerica,variable_string;
```

Ejemplo:

```
v='123 RPM';
aton c,v;
#Ahora c vale 123;
```

Funciones *"day","month","year","hs","min","sec*" y *"nday"*

Estas funciones convierten un ***time_stamp*** a día, mes, año, hora, minuto, segundo y número de día de la semana, respectivamente.
Como vimos anteriormente, el fecha/hora actual se lee con la función ***read_io*** tipo **7** y se guarda en una variable numérica que representa la cantidad de segundos desde el 01/01/2000.

Sintaxis:

```
day dia,timestamp;

month mes,timestamp;

year año,timestamp;

hs hora,timestamp;

min minutos,timestamp;

sec segundos,timestamp;

nday dia_semana,timestamp;
```

La función ***"nday"*** devuelve el número de día de la semana comenzando por el domingo = 0.

Ejemplo:

```
read_io 7,e,0; #Lee la hora actual en e;
day f,e;
month g,e;
year h,e;
hs i,e;
min j,e;
sec k,e;
#La fecha y hora actual es f/g/h i:j:k;
```

Funciones matemáticas y lógicas

Estas funciones realizan ciertas operaciones matemáticas especiales.

Funcion *"neg"*

Se utiliza para negar una variable numérica. La negación se produce a nivel de *bit*.

Sintaxis:

```
neg resultado,inicio;
```

Primero se coloca la variable en la cual se quiere obtener el resultado y luego la variable a negar.

Ejemplo:

```
a=32323; #7E43h
neg b,a;

# b vale 4294934972 (FFFF81BC);
```

Función *"sqtr"*

Realiza la raíz cuadrada. Como el Exemys Script solo maneja variables enteras de tipo "long", la parte

decimal del resultado se verá truncada. Para evitar esto se recomienda multiplicar antes de realizar la operación.

Sintaxis:

```
sqrt resultado,inicio;
```

Ejemplo:

```
a=225;
sqrt b,a;
#Ahora b vale 15;
```

Función *"scale"*

Permite escalar una variable utilizando la ecuación de la recta que pasa por 2 puntos.

Sintaxis:

```
scale resultado,inicio,x0,x1,y0,y1;
```

Ejemplo: Escalar la señal 4-20mA de la entrada AN1 en un valor de 0 a 500

```
read_io 2,a,1; #a = AN1
scale c,a,400,2000,0,500;
#Ahora c tiene el valor escalado;
```

Funciones de temporizado

Estas funciones permiten controlar el flujo del programa en función de tiempos.

Funciones *"timer"* y *"check_timer"*

Con estas funciones podemos definir un periodo de tiempo y ejecutar instrucciones cuando este se cumpla.
La forma de uso es la siguiente. Primero ejecutamos la función ***"timer"*** a la cual le pasamos como parámetros una variable numérica y un tiempo en milisegundos. Luego se consulta esta variable con la función ***"check_timer"***.

Sintaxis:

```
timer variable_numerica,tiempo_en_milisegundos;
...
check_timer variable_numerica
{
    ...
    ...
};
```

Como podemos ver, con la función ***"timer"*** definimos el tiempo, y con ***"check_timer"*** consultamos la variable cargada anteriormente. Cuando se cumple el periodo, las instrucciones colocadas entre llaves se ejecutan.
Debemos tener en cuenta que una vez cumplido el periodo la condición siempre será verdadera, y si volvemos a realizar un ***"check_timer"*** se ejecutaran nuevamente las instrucciones. Normalmente se carga de vuelta la variable dentro del bloque "***check_timer"***

Nota: Las funciones de temporizado no deben usarse en aplicaciones de precisión de tiempo, ya que el temporizado puede presentar cierta dispersión.

2020-11-25

Introducción

Con las funciones read_io, write_io, read_str y write_str se puede acceder a múltiples funciones adicionales. En esta sección del manual se listan las distíntas fuentes/destinos y luego de explica su uso por función.

Listado de fuentes/destinos

Se indica la versión desde la cual está disponilble una fuente/destino en caso que se haya agregedo a la versión inicial.

Fuentes para ***"read_io"***

Fuente	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
0	Canal de entrada digital (Ix)	1 a 100	Si	Si	Si	Si	Si	Si
1	Canal de salida digital (Ox)	1 a 100	Si	Si	Si	Si	Si	Si
2	Canal entrada analógica (ANx)	1 a 100	Si	Si	Si	Si	Si	Si
3	Canal de entrada de pulsos (PIx)	1 a 100	Si	Si	Si	Si	Si	Si
23	Canal de entrada de pulsos vinculado a consulta Modbus Float 32. Devuelve la parte entera del valor por 1000 (PIx)	1 a 100	5.1.2	Si	Si	Si	Si	Si
36	Canal de entrada de pulsos vinculado a consulta Modbus Float 32 con bytes invertidos. Devuelve la parte entera del valor por 1000 (PIx)	1 a 100	5.2.0	Si	Si	Si	Si	Si
305	Memoria de canales	0	-	-	1.9	2.8	Si	Si
7	Reloj de tiempo real (segundos desde 1/1/2020)	0	Si	Si	Si	Si	Si	Si
8	Cantidad de registros históricos no enviados almacenados en memoria	0	Si	Si	Si	Si	Si	Si
9	Estado de conexión GSM (ver tabla abajo)	0	Si	-	Si	-	Si	-
10	Estado de conexión GPRS (ver tabla abajo)	0	Si	-	Si	-	Si	-
11	Estado de conexión MW (ver tabla abajo)	0	Si	Si	Si	Si	-	-i
21	Memoria no volátil para números (lectura)	1 a 20	5.1.1	Si	Si	Si	Si	Si
22	Lee si la conexión al MW está habilitada en la configuración	0	5.2.2	Si	Si	Si	-	-
37	Cantidad de datos en el buffer de puerto serie (Los datos se borran del buffer con write_io 37)	0	5.2.0	Si	Si	Si	Si	Si
38	Lee el valor binario de la posición indicada del buffer del puerto serie	1 a 100	5.2.0	Si	Si	Si	Si	Si
39	Cantidad de SMS en bandeja de salida	0	5.2.4	-	Si	-	Si	-
48	Deshabilitar el envio de registros históricos al MW (1 deshabilitado, 0 habilitado)	0	5.2.0	Si	Si	Si	Si	Si
47	Estado de cliente FTP	0	5.2.2	Si	-	Si	-	Si
55	Estado del envío por modem satelital.	0	5.2.2	Si	Si	Si	-	-
61	Devuelve el 'tipo de canal' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
62	Devuelve el 'timestamp' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
63	Devuelve el 'tipo de historico' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
64	Devuelve el 'numero de canal' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
65	Devuelve el 'valor' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
75	Estado de recepción del socket UDP (1 = listo)	0	-	-	-	2.2	-	Si
76	Estado de transmisión del socket UDP (1 = listo)	0	-	-	-	2.2	-	Si
77	Lectura binaria del socket UDP. Indica cuanto datos se recibieron	0	-	-	-	2.2	-	Si
78	Lee el valor binario de la posición indicada del buffer del socket UDP	0 a 100	-	-	-	2.2	-	Si
81	Largo de la respuesta al cliente HTTP	0	-	-	-	2.2	-	Si
82	Estado del cliente HTTP	0	-	-	-	2.2	-	Si
95	Estado del cliente SMTP	0	-	-	-	2.2	-	Si
195	Estado del cliente POP	0	-	-	-	2.2	-	Si
270	Lectura directa de consultas Modbus - Valor	1 a 100	-	-	1.8	-	Si	-
271	Lectura directa de consultas Modbus - Estado	1 a 100	-	-	1.8	-	Si	-
280	Estado de Roaming (0=no, 1=si)	0	-	-	1.8	-	Si	-
1000	MQTT-Cantidad de mensajes recibidos por suscripciones pendientes de lectura	0	-	-	-	-	Si	Si
1001	MQTT-Estado de conexión con el broker (0=no conectado, 1=conectado)	0	-	-	-	-	Si	Si

Destinos para ***"write_io"***

Destino	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
1	Canal de salida digital (Ox)	1 a 100	Si	Si	Si	Si	Si	Si
2	Canal entrada analógica (ANx, solo Modbus)	1 a 100	5.1.3	Si	Si	Si	Si	Si

3	Canal de pulsos (PIx)	1 a 100	Si	Si	Si	Si	Si	Si
305	Memoria de canales	0	-	-	1.9	2.8	Si	Si
7	Puesta en hora(segundos desde 1/1/2000)	0	-	-	1.8	2.6	Si	Si
56	Generar Histórico de cambio de canal Ix	1 a 100	5.2.2	Si	1.6	2.4	Si	Si
57	Generar Histórico de cambio de canal Ox	1 a 100	5.2.2	Si	1.6	2.4	Si	Si
12	Generar Histórico por tiempo de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
14	Generar Histórico de alarma máxima de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
15	Generar Histórico de alarma mínima de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
16	Generar Histórico de alarma normal de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
13	Generar Histórico por tiempo de canal PIx	1 a 100	Si	Si	Si	Si	Si	Si
17	Forzar Reporte de canal Ix	1 a 100	Si	Si	Si	Si	Si	Si
18	Forzar Reporte de canal Ox	1 a 100	Si	Si	Si	Si	Si	Si
19	Forzar Reporte de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
20	Forzar Reporte de canal PIx	1 a 100	Si	Si	Si	Si	Si	Si
21	Memoria no volátil para números (escritura)	1 a 20	Si	Si	Si	Si	Si	Si
22	Escribir en la configuración del equipo la habilitación de conexión al MW (0 o 1)	0	5.2.2	Si	Si	Si	-	-
48	Deshabilitar el envio de registros históricos al MW (1 deshabilitado, 0 habilitado)	0	5.2.2	Si	Si	Si	Si	Si
37	Borrar los primeros N datos del buffer del puerto serie (usar junto con read_io 37 y read_io 38)	0	5.2.0	Si	Si	Si	Si	Si
38	Enviar un byte al puerto serie (valor binario)	0	5.2.5	-	Si	Si	Si	Si
60	Leer en memoria los datos de un registro específico de la memoria de registros (usar junto con read_io 61 a 65)	-	5.2.2	Si	Si	Si	Si	Si
66	Borrar los primeros N registros de la memoria de registros históricos	-	5.2.2	Si	Si	Si	Si	Si
54	Envio de registros históricos por modem satelital	0	5.2.2	Si	Si	Si	-	-
32	Inicia chequeo de recepcion de datos por modem satelital	0	-	-	1.3	2.2	-	-
59	Cambia el multiplicador de read_io 23 y 36 de 1000 a otro valor	0	5.2.5	-	Si	Si	Si	Si
44	Iniciar conexión de cliente FTP	0	5.2.2	Si	-	Si	-	Si
46	Cerrar archivo y terminar conexión de cliente FTP	0	5.2.2	Si	-	Si	-	Si
77	Enviar N datos binarios cargados previamente en el buffer del socket UDP	0	-	-	-	2.2	-	Si
78	Carga datos en el buffer binario del socket UDP	0 a 100	-	-	-	2.2	-	Si
82	Iniciar conexión de cliente HTTP	0	-	-	-	2.2	-	Si
195	Iniciar conexión de cliente POP	0	-	-	-	2.2	-	Si

Fuentes para *"read_str"*

Fuente	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
4	Texto del SMS	Si	-	Si	-	Si	-
5	Número telefónico del SMS	Si	-	Si	-	Si	-
6	Puerto Serie	Si	Si	Si	Si	Si	Si
32	Texto recibido por modem satelital (serie transparente en MW)	-	-	1.3	2.2	-	-
35	Traces del Script Programmer	5.2.0	Si	Si	Si	Si	Si
51	Lectura de buffer de proceso con agregado de inicio, fin y checksum de NMEA (cargar antes el buffer de proceso con write_str 50)	5.2.0	Si	Si	Si	Si	Si
77	Socket UDP	-	-	-	2.2	-	Si
81	Cliente HTTP	-	-	-	2.2	-	Si
193	Remitente de email recibido por cliente POP3	-	-	-	2.2	-	Si
194	Asunto de email recibido por cliente POP3	-	-	-	2.2	-	Si
195	Cuerpo de email recibido por cliente POP3. Genera carga el proximo mensaje en bandeja	-	-	-	2.2	-	Si
101 a 108	Nombres de la agenda telefonica 1 a 8	5.1.3	-	Si	-	Si	-
111 a 118	Números de la agenda telefónica 1 a 8	5.1.3	-	Si	-	Si	-
121 a 125	Memoria no volátil para texto 1 a 5 (lectura)	5.2.2	Si	Si	Si	Si	Si
1000	MQTT. Lee primer mensaje en cola de suscripciones	-	-	-	-	Si	Si

Destinos para *"write_str"*

Destino	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
4	Texto del SMS	Si	-	Si	-	Si	-
5	Número telefónico del SMS	Si	-	Si	-	Si	-
6	Puerto Serie	Si	Si	Si	Si	Si	Si

35	Traces del Script Programmer	5.2.0	Si	Si	Si	Si	Si
50	Buffer de proceso (usar junto con read_str 51)	5.2.0	Si	Si	Si	Si	Si
121 a 125	Memoria no volátil para texto 1 a 5 (escritura)	5.2.2	Si	Si	Si	Si	Si
40	Cargar URL para cliente FTP	5.2.2	Si	-	Si	-	Si
41	Cargar usuario FTP	5.2.2	Si	-	Si	-	Si
42	Cargar contraseña FTP	5.2.2	Si	-	Si	-	Si
43	Cargar nombre de archivo para cliente FTP	5.2.2	Si	-	Si	-	Si
45	Cargar linea de texto en archivo FTP y enviarla	5.2.2	Si	-	Si	-	Si
75	Inicializar socket UDP y poner en escucha en el puerto indicado	-	-	-	2.2	-	Si
76	Configurar direccion IP y puerto destino del socket UDP	-	-	-	2.2	-	Si
77	Enviar un string por el socket UDP	-	-	-	2.2	-	Si
80	Configurar URL y puerto del cliente HTTP	-	-	-	2.2	-	Si
84	Configuración de la ruta/nombre de archivo del cliente HTTP	-	-	-	2.8	-	Si
81	Query string a pasar en el GET (xx=123&yy=456…) de cliente HTTP	-	-	-	2.2	-	Si
83	Valor a asignar al campo 'data' en el query string del GET (Alternativo a write_str 81)	-	-	-	2.2	-	Si
89	Configurar URL y puerto de cliente SMTP	-	-	-	2.2	-	Si
90	Configurar correo de remitente de cliente SMTP	-	-	-	2.2	-	Si
91	Configurar usuario de cliente SMTP	-	-	-	2.2	-	Si
92	Configurar contraseña de cliente SMTP	-	-	-	2.2	-	Si
93	Dirección de correo del destinatario de cliente SMTP	-	-	-	2.2	-	Si
94	Asunto del correo de cliente SMTP	-	-	-	2.2	-	Si
95	Cuerpo del correo y disparar el envio de cliente SMTP	-	-	-	2.2	-	Si
189	Configurar URL y puerto del cliente POP	-	-	-	2.2	-	Si
191	Configurar usuario de cliente POP	-	-	-	2.2	-	Si
192	Configurar contraseña de cliente POP	-	-	-	2.2	-	Si
1001	MQTT-Cargar tópico a publicar	-	-	-	-	Si	Si
1002	MQTT-Cargar payload a publicar y hacerlo	-	-	-	-	Si	Si

Lectura/Escritura de canales de entrada/salida

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
0	Canal de entrada digital (Ix)	1 a 100	Si	Si	Si	Si	Si	Si
1	Canal de salida digital (Ox)	1 a 100	Si	Si	Si	Si	Si	Si
2	Canal entrada analógica (ANx)	1 a 100	Si	Si	Si	Si	Si	Si
3	Canal de entrada de pulsos (PIx)	1 a 100	Si	Si	Si	Si	Si	Si
23	Canal de entrada de pulsos vinculado a consulta Modbus Float 32. Devuelve la parte entera del valor por 1000 (PIx)	1 a 100	5.1.2	Si	Si	Si	Si	Si
36	Canal de entrada de pulsos vinculado a consulta Modbus Float 32 con bytes invertidos. Devuelve la parte entera del valor por 1000 (PIx)	1 a 100	5.2.0	Si	Si	Si	Si	Si

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
1	Canal de salida digital (Ox)	1 a 100	Si	Si	Si	Si	Si	Si
2	Canal entrada analógica (ANx, solo Modbus)	1 a 100	5.1.3	Si	Si	Si	Si	Si
3	Canal de pulsos (PIx)	1 a 100	Si	Si	Si	Si	Si	Si
59	Cambia el multiplicador de read_io 23 y 36 de 1000 a otro valor	0	5.2.5	-	Si	Si	Si	Si

Las fuentes 0 a 3 devuelven los valores de los distintos canales de entradas/salidas del equipo. El **"índice"** indica el numero de canal a leer.

Ejemplo: Leer el valor del canal de entradas analógicas 4 (AN4) y guardarlo en la variable c

```
read_io 3,c,4;
```

El destino 0 permite activar las salidas del equipo. El **"índice"** indica el numero de canal a escribir.

Ejemplo: Apagar el canal de salida digital 3 (O3)

```
write_io 1,3,0;
```

El destino 2 de **canales de entrada analógica** permiten escritura solo si están vinculados a consultas Modbus. El llamado a write_io generará un comando es escritura Modbus

El destino 3 de **canales de pulsos** admiten los mismos valores que admita el equipo en esos canales (contadores físico o consulta Modbus de largo 2).

Las fuentes 23 y 36 permiten convertir registros Modbus Float 32 vinculados a canales de pulsos a valores enteros.

Memoria de canales

read_io /			GRD-XF-	cLAN-XF	GRD-XF-	cLAN-XF		

read_io / write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
305	Memoria de canales	0	-	-	1.9	2.8	Si	Si

Esta zona de memoria volatil con 100 posiciones puede funcionar como "Source" de todos los canales.

El valor de esta memoria puede leerse y escribirse con `read_io/write_io 305`

Esto permite liberar variables numéricas del script que antes se ocupaban para vincularlas a canales.

Lectura directa de consultas Modbus

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
270	Lectura directa de consultas Modbus - Valor	1 a 100	-	-	1.8	-	Si	-
271	Lectura directa de consultas Modbus - Estado	1 a 100	-	-	1.8	-	Si	-

Las fuentes 270 y 271 permiten leer el valor de una consulta Modbus sin necesidad de previamente mapear esta consulta en un canal.

Esto permite no despediciar canales para usarlos como punto intermedio antes de procesar valoren en el script.

La fuente 271 devuelve un 0 si hay falla en la comunicación Modbus o 1 si la comunicación no tiene errores

Ejemplo: Leer el valor de la consulta Modbus 4 y guardarlo en la variable c

```
read_io 270,c,4;
```

Lectura de reloj de tiempo real

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
7	Hora actual (segundos desde 1/1/2000)	0	Si	Si	Si	Si	Si	Si

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
7	Puesta en hora(segundos desde 1/1/2000)	0	-	-	1.8	2.6	Si	Si

La fuente 7 permite leer la fecha/hora actual del equipo. El número pueder ser convertido a texto usando las funciones de conversión.

Ejemplo: Obtener el mes actual en la variable g

```
read_io 7,e,0;
month g,e;
```

En reciente versiones de firmware también se puede configurar la hora desde el script con `write_io 7`

Lectura/Escritura de memoria no volatil

Para números

read_io / write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
21	Memoria no volátil (lectura)	1 a 20	5.1.1	Si	Si	Si	Si	Si

La fuente/destino 21 permite leer y escribir hasta valóres numéricos en la memoria no volatil del equipo.

En equipos con firmware 5.1.1, no invocar a esta función permanentemente, solo cuando el valor cambia (se corre el riesgo de dañar la memoria).

Ejemplo: Leer el valor numérico almacenado en al posición 15 de la memória no volatil en la variable g

```
read_io 21,g,15;
```

Para texto

read_str / write_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
121 a 125	Memoria no volátil para texto 1 a 5	5.2.2	Si	Si	Si	Si	Si

Las fuentes/destinos 121 a 125 permiten leer y escribir hasta 5 textos en la memoria no voltatil del equipo.

Ejemplo: Escribir la palabra 'hola' en la tercera posición de la memoria no volatil para textos.

```
write_str 123,'hola';
```

Lectura de estados de GSM/GPRS/MW y configuración de conexión al MW

Lectura de estados

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
9	Estado de conexión GSM (ver tabla abajo)	0	Si	-	Si	-	Si	-
10	Estado de conexión GPRS (ver tabla abajo)	0	Si	-	Si	-	Si	-
11	Estado de conexión MW (ver tabla abajo)	0	Si	Si	Si	Si	-	-
280	Estado de Roaming (0=no, 1=si)	0	-	-	1.8	-	Si	-

Desde las fuentes 9, 10, 11 y 280 se puede conecer el estado de GSM, GPRS y la conexión al MW. Los posibles estados se listan a continuación

Estados de GSM (solo para GRD)

#	Estado de GSM
0	OFF
1	ATTACHING
2	SIM NOT INSERTED
3	PIN REQUIERED
4	PIN ERROR
5	PIN OK
6	BLOQUED
7	**LOW SIGNAL**
8	ACCESS DENIED
9	**READY**

Estados de GPRS (solo para GRD)

#	Estado de GPRS
0	OFF
1	WAIT GSM READY
2	ATTACHING
3	**CONNECTED**
4	ERROR
5	WAIT RECONNECTION

Estados de Middleware

#	Estado de Middleware
0	OFF
1	WAIT GPRS READY (solo GRD)
2	-
3	CONNECTION REFUSED
4	CONNECTION FAILED
5	HOST UNREACHABLE
6	HOST CLOSED CONNECTION (solo GRD)
7	**CONNECTED**
8	ERROR
9	WAIT RECONNECTION
10	DNS FAILURE
11	LOGGING IN (Solo GRD desde 5.1.2)

Configuración de conexión con MW

read_io / write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
22	Configuración de conexión al MW (1 habilitada / 0 deshabilitada)	0	5.2.2	Si	Si	Si	-	-

La fuente/destino 22 permite configurar si el equipo se conecta o no al MW.

Tenga en cuenta que esta configuración se guarda en la memoria no volatil del equipo, y que una vez desconectado del MW no podrá acceder mas en forma remota. Puede volver a conectarlo al MW mediante el SMS de conexión (solo en GRD)

Deshabilitar el envio de registros históricos al MW

read_io / write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
48	Deshabilitar el envio de registros históricos al MW (1 deshabilitado, 0 habilitado)	0	5.2.2	Si	Si	Si	-	-

La fuente/destino 48 permite deshabilitar el envío de registros históricos al MW.

Tenga en cuenta que esta configuración se guarda en la memoria no volatil del equipo, y que si desea que el equipo vuelta a mandar históricos debe volver a habilitar el envío.

Esta función es util cuando se desea enviar registros históricos por algún medio alternativo.

Ejemplo: Deshabilitar el envío de registro históricos al MW.

```
write_io 48,0,1;;
```

Envio/Recepción de SMS. Agenda telefónica (Solo en GRD)

Recepción

read_str	Descripción	GRD-XF-2G	GRD-XF-3G	GRD-MQ
4	Texto del SMS	Si	Si	Si
5	Número telefónico del SMS	Si	Si	Si

Con las fuentes 4 y 5 se pueden recibir mensajes de texto. Para saber si llegó un mensaje se debe leer el texto del mensaje constantemente hasta que el largo del sea diferente de 0.

Ejemplo: Verificar si llega un mensaje de texto

```
if 1 {
     read_str 4,a,v;
     read_str 5,b,w;
};
if a!0 {
     #Se recibio un SMS;
};
```

Envío

write_str	Descripción	GRD-XF-2G	GRD-XF-3G	GRD-MQ
4	Texto del SMS	Si	Si	Si
5	Número telefónico del SMS	Si	Si	Si

Los destinos 4 y 5 se pueden usar para enviar mensajes de texto. El mensaje se envia al escribir el texto en el destino 4. Previamente debe cargar el numero de destino en el destino 5.

Si se acaba de recibir un mensaje y lo quiere responder, puede escribir directamente la respuesta en el destino 4 sin escribir antes el número telefónico.

Ejemplo: Enviar un mensaje de texto

```
write_str 5,'1166041241'; #Carga el número telefónico destino;
write_str 4,'Hola'; #Carga el texto del SMS y lo envia;
```

read_io	Descripción	Indice	GRD-XF-2G	GRD-XF-3G	GRD-MQ
39	Cantidad de SMS en bandeja de salida	0	5.2.4	Si	Si

Permite saber cuantos SMS están esperando para ser enviados.

Agenda telefónica

read_str	Descripción	GRD-XF-2G	GRD-XF-3G	GRD-MQ
101 a 108	Nombres de la agenda telefonica 1 a 8	5.1.3	Si	Si
111 a 118	Números de la agenda telefónica 1 a 8	5.1.3	Si	Si

Las fuentes 101 a 108 permiten leer los nombres de la agenda telefónica del GRD

Las fuentes 111 a 118 permiten leer los números telefónicos de la agenda telefónica del GRD

Estas fuentes resultan prácticas si se quiere cambiar el destinatario de un mensaje enviado desde el script sin tener que editar el programa.

Ejemplo: Leer el número telefónico en la posición 5 de la agenda del GRD

```
read_str 115,a,v;
```

Envio/Recepción de mensajes al Script Programmer ("Traces")

read_str / write_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
35	Trazas del Script Programmer	5.2.0	Si	Si	Si	Si	Si

El destino 35 permite enviar texto a la ventana de "Traces" en el Script Programmer (disponible desde Script Programmer V2.0) . Esto es particularmente util cuando se está probando un script nuevo.

Hay una consideración con respecto a los caracteres de espacio y guión bajo. El caracter de espacion será reemplazado por un guión bajo antes de leer con *read_str 35*. El guión bajo será reemplazado por un espacio luego de enviar con *write_str 35*.

Si el Script Programmer no se encuenta conectado al escribir en el destino 35 el texto simplemente se perderá pero no afectará al funcionamiento del programa.

Acceso al puerto serie en modo texto

read_str / write_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
6	Puerto Serie	Si	Si	Si	Si	Si	Si

La fuente/destino 6 permite recibir y enviar texto desde y hacia el puerto serie del equipo. Para saber si llegó texto al puerto serie se debe leer la fuente 6 constantemente hasta que el largo del sea diferente de 0.

Para enviar un texto simplemente escriba sobre el destino 6.

Para que funcione correctamente debe configurar el puerto serie en modo "Script"

Si quiere enviar caracteres binarios puede utilizar el operador $

Ejemplo: Hacer "eco" del texto recibido por el puerto serie.

```
read_str 6,a,v;
if a!0 {
    write_str 6,v;
};
```

Acceso al puerto serie en modo binario

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
37	Cantidad de datos en el buffer de puerto serie (Los datos se borran del buffer con write_io 37)	0	5.2.0	Si	Si	Si	Si	Si
38	Leer el valor binario de la posición indicada del buffer del puerto serie	1 a 100	5.2.0	Si	Si	Si	Si	Si

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
37	Borrar los primeros N datos del buffer del puerto serie (usar junto con read_io 37 y read_io 38)	0	5.2.0	Si	Si	Si	Si	Si
38	Enviar un byte al puerto serie (valor binario)	0	5.2.5	-	Si	Si	Si	Si

La fuente/destino 37, mas la fuente 38 permiten interpretar datos binario binarios recibos en el puerto serie.

Para que funcione correctamente debe configurar el puerto serie en modo "Script"

Si quiere enviar datos 'binarios' puede usar el destino puede usar write_str 6 junto con el operador $

Ejemplo: Esperar a recibir mas de 2 bytes. Luego verificar si el tercer byte recibido es el binario 126. Finalmente borrar 3 bytes del buffer

```
read_io 37,a,0;
if a>2 {
    read_io 38,b,3;
    if b=126 {
        #El 3er byte recibido es el binario 126;
    };
    write_io 37,0,3;
};
```

Generación de registros históricos

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
56	Generar Histórico de cambio de canal Ix	1 a 100	5.2.2	Si	1.6	2.4	Si	Si
57	Generar Histórico de cambio de canal Ox	1 a 100	5.2.2	Si	1.6	2.4	Si	Si
12	Generar Histórico por tiempo de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
14	Generar Histórico de alarma máxima de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
15	Generar Histórico de alarma mínima de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
16	Generar Histórico de alarma normal de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
13	Generar Histórico por tiempo de canal PIx	1 a 100	Si	Si	Si	Si	Si	Si

Estos destinos permiten generar registros **históricos** desde el script, además de los que genera el equipo por si mismo. El valor del histórico se debe indicar en el campo *valor*.

Se recomienda usar esta función con cuidado para no generar registros históricos permanentemente.

Ejemplo: Generar un registro histórico por tiempo del canal AN2 cada 10 segundos con el valor 457

```
check_timer t
{
    timer t,10000;
    write_io 12,2,457;
};
```

Forzado de envío de reportes

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
17	Forzar Reporte de canal Ix	1 a 100	Si	Si	Si	Si	Si	Si
18	Forzar Reporte de canal Ox	1 a 100	Si	Si	Si	Si	Si	Si
19	Forzar Reporte de canal ANx	1 a 100	Si	Si	Si	Si	Si	Si
20	Forzar Reporte de canal PIx	1 a 100	Si	Si	Si	Si	Si	Si

Los destinos 17 a 20 permiten forzar el envío de **reportes**, además de los que genera el equipo por si mismo. El valor del reporte es el que tenga el canal en ese momento. Se ignora el campo **valor**.

Se recomienda usar esta función con cuidado para no generar tráfico GPRS de manera permanentemente (solo GRD)

Ejemplo: Generar un reporte del canal AN3 cada 10 segundos con su valor actual

```
check_timer t
{
    timer t,10000;
    write_io 19,3,0;
};
```

Acceso a memoria de registros históricos

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
8	Cantidad de registros históricos no enviados almacenados en memoria	0	Si	Si	Si	Si	Si	Si
61	Devuelve el 'tipo de canal' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
62	Devuelve el 'timestamp' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
63	Devuelve el 'tipo de historico' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
64	Devuelve el 'numero de canal' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si
65	Devuelve el 'valor' de un registro leido de la memoria con write_io 60	0	5.2.2	Si	Si	Si	Si	Si

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
60	Leer en memoria los datos de un registro específico de la memoria de registros (usar junto con read_io 61 a 65)	0	5.2.2	Si	Si	Si	Si	Si
66	Borrar los primeros N registros de la memoria de registros históricos	-	5.2.2	Si	Si	Si	Si	Si

Las fuentes/destinos 60 a 66 permiten leer la memoria de registros históricos del equipo. Estas funciones se utilizan para poder enviar el contenido de esta memoria por un medio que no sea el envío al Middleware, por ejemplo, por FTP.

Antes de intentar leer un registro puede ver cuando registros hay en memoria usando *read_io 8*. Luego puede invocar a *write_io 60* para leer un registro particular y *read_io* 61 a 65 para obtener los valores de los campos del registro leido. Finalmente puede usar *write_io 66* para borra el/los registros leidos.

Ejemplo: Leer los registro históricos de la memoria del equipo y enviarlos a la consola de "Traces" del Script Programmer

```
read_io 8,a,0;
if a>0
{
    write_io 60,0,1 ; #Lee el primer registro de la memoria;
    read_io 61,b,0; #Carga en b el tipo de canal;
    read_io 62,c,0; #Carga en c el timestamp;
    read_io 63,d,0; #Carga en d el tipo de historico;
    read_io 64,e,0; #Carga en e el numero de canal;
    read_io 65,f,0; #Carga en f el valor
    w=b,'-',b,'-',c,'-',d,'-',e,'-',f,$13,$10;
    write_str 35,w; #Envia el texto del registro a la consola;
    write_io 66,0,1; #Borra el registro enviado de la memoria del
equipo;
};
```

Para un ejemplo mas completo del uso de esta funciones vea el ejemplo de uso de FTP.

Modem satelital

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
55	Estado del envío por modem satelital.	0	5.2.2	Si	Si	Si	-	-

#	Estado del envío
-7	No envía. Conectado al MW
-6	Error enviando registros
-5	Inicializando modem

-4	Error en la configuración del puerto serie (Modo satelite y a 19200 bps)
-3	No hay registros para enviar
-2	Error al leer la memoría de registros
-1	Enviando registros
0	Listo para enviar
> 0	Cantidad de registros enviados

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
54	Envio de registros históricos por modem satelital	0	5.2.2	Si	Si	Si	-	-
32	Inicia chequeo de recepcion de datos por modem satelital (genera consumo)	0	-	-	1.3	2.2	-	-

read_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
32	Texto recibido por modem satelital (serie transparente en MW)	-	-	1.3	2.2	-	-

Si tiene conectado un modem SBD Iridium (EDGE/ITAS) al puerto serie del GRD/cLAN, puede enviar los registros históricos en su memoria al MW usando la red satelital de Iridium. En el manual del equipo puede ver mas detalles de la solución.

Se recomienda usar con discrecionalidad y conocer el costo del envio de datos por modem satelital.

El destino 54 permite iniciar el envio de los registros que el equipo tenga en memoria. Para iniciar el envio el modem debe estar en estado "Listo para enviar". El GRD/cLAN enviará todos los registros que pueda en un solo mensaje satelital.

Desde la versión 1.3 de GRD-XF-3G y 2.2 de cLAN tambien se pueden recibir datos con el modem satelital. Los datos no llegan espontaneamente, el GRD/cLAN deb chequear si llegaron datos nuevos. Esto se realiza automaticamente cada vez que se envian datos. También se puede iniciar un chequeo de recepción de datos usando el write_io 32. Tenga en cuenta que cada vez que se chequee, Iridium genera un cargo. Para el envío de datos desde el MW hacia el GRD/cLAN se debe contratar y configurar en el MW el servicio de Iridium llamado “Static IP addresses for Mobile Terminated SBD”

Usando read_str 32 se puede leer el texto enviado usando el modem satelital. Este texto se introduce al MW usando la conexión de puerto serie transparente.

Cada vez que se envian (write_io 54) o chequean datos (write_io 32) también se pueden recibir comandos desde el MW generados desde la tabla de escrituras en la base de datos. De esta manera se pueden modificar canales de salidas digitales, analogicos de consultas Modbus o canales vinculados a variables del script de manera remota.

Como ejemplo por favor mire el archivo *satelite.sce* que puede descargar junto a los otros ejemplos de uso de script www.exemys.com/EjemplosDeScriptEnGRD

Cliente FTP

read_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
47	Estado de cliente FTP	0	5.2.2	Si	-	Si	-	Si

#	Estado de cliente FTP
0	**IDLE**
2	CONNECTING
3	CONNECTION FAIL
7	ERROR
8	SENDING FILE
9	WAIT READY TO SEND
10	**READY TO SEND**

write_io	Descripción	Indice	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
44	Iniciar conexión de cliente FTP	0	5.2.2	Si	-	Si	-	Si
46	Cerrar archivo y terminar conexión de cliente FTP	0	5.2.2	Si	-	Si	-	Si

write_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
40	Cargar URL para cliente FTP	5.2.2	Si	-	Si	-	Si
41	Cargar usuario FTP	5.2.2	Si	-	Si	-	Si
42	Cargar contraseña FTP	5.2.2	Si	-	Si	-	Si
43	Cargar nombre de archivo para cliente FTP	5.2.2	Si	-	Si	-	Si

45	Cargar linea de texto en archivo FTP (máximo 100 caracteres)	5.2.2	Si	-	Si	-	Si

Las fuentes/destinos listados en estas tablas permiten implementar un cliente FTP para la subida de archivos de texto a un servidor. Normalmente esta función será utilizada para enviar registros históricos de la memória del equipo y en conjunto con las fuentes/destinos que permiten acceder a la memoria de registros históricos.

Para poder enviar datos por FTP el GRD-2G debe estar registrado en la red GPRS (solo GRD).

Como ejemplo por favor mire el archivo *ftp.sce* que puede descargar junto a los otros ejemplos de uso de script www.exemys.com/EjemplosDeScriptEnGRD

Cálculo de checksums y CRCs

write_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
50	Buffer de proceso (usar junto con read_str 51)	5.2.0	Si	Si	Si	Si	Si

Este destino permite cargar un string en el buffer de proceso para luego aplicar algún proceso al texto cargado

Proceso de protocolo NMEA

read_str	Descripción	GRD-XF-2G	cLAN-XF V1	GRD-XF-3G	cLAN-XF V2	GRD-MQ	cLAN-MQ
51	Leer el buffer de proceso con agregado de inicio, fin y checksum de NMEA (cargar antes el buffer de proceso con write_str 50)	5.2.0	Si	Si	Si	Si	Si

La fuente 51 guardara en la variable la trama NMEA previamente cargada en el buffer de proceso con *write_str 50*.

Esta fuente agrega a la trama original el inicio, el fin y el checksum de NMEA. Esto permite simular un "talker" NMEA con el equipo

Ejemplo: Enviar la sentencia NMEA *GPMWV,145.8,R,87.2,K,A* por el puerto serie del equipo, luego de agregarle el caracter de inicio, el de fin y el checksum

```
write_str 50,'GPMWV,145.8,R,87.2,K,A';
read_str 51,a,w;
write_str 6,w;
```

Socket UDP (solo cLAN)

read_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
75	Estado de recepción del socket UDP (1 = listo)	0	-	2.2	Si
76	Estado de transmisión del socket UDP (1 = listo)	0	-	2.2	Si
77	Lectura binaria del socket UDP. Indica cuanto datos se recibieron	0	-	2.2	Si
78	Lee el valor binario de la posición indicada del buffer del socket UDP	0 a 100	-	2.2	Si

write_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
77	Enviar N datos binarios cargados previamente en el buffer del socket UDP	0	-	2.2	2.2
78	Carga datos en el buffer binario del socket UDP	0 a 100	-	2.2	2.2

read_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
77	Socket UDP	-	2.2	Si

write_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
75	Inicializar socket UDP y poner en escucha en el puerto indicado	-	2.2	Si
76	Configurar direccion IP y puerto destino del socket UDP (no resuelve URL, solo usar dirección IP)	-	2.2	Si
77	Enviar un string por el socket UDP	-	2.2	Si

Las fuentes/destinos listados en estas tablas permiten enviar y recibir datos usando un socket UDP. Esto permite generar comunicación entre equipos cLAN o con algún otro equipo o software desarrollado por el usuario.

Existen dos modos de uso, texto (read_str/write_str 77)

```
read_str 77,d,z;
```

```
write_str 77,'Hello';
```

y binario (read_io 77 y 78 y write_io 77 y 78)

Para empezar a usar el socket se lo debe inicializar usando write_str 75 y 76.

```
write_str 75,'2680';
write_str 76,'192.168.0.39:1520';
```

Antes de recibir o enviar datos se debe chequear que el socket este listo con read_io 75 y read_io 76

Se recomienda observar los ejemplos UDPtexto.sce, UDPbinario.sce y UDPmultidestino.sce que puede descargar junto a los otros ejemplos de uso de script www.exemys.com/EjemplosDeScriptEnGRD

Cliente HTTP (solo cLAN)

read_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
81	Largo de la respuesta al cliente HTTP	0	-	2.2	Si
82	Estado del cliente HTTP	0	-	2.2	Si

#	Estado de cliente SMTP
0	IDLE
1	ENVIANDO
2	ENVIO OK
3	ERROR AL ENVIAR

write_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
82	Iniciar conexión de cliente HTTP	0	-	2.2	Si

read_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
81	Leer la respuesta al Cliente HTTP (cuerpo HTML)	-	2.2	Si

write_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
80	Configurar URL y puerto del cliente HTTP	-	2.2	Si
84	Configuración de la ruta/nombre de archivo del cliente HTTP	-	2.8	Si
83	Query string a pasar en el GET (xx=123&yy=456…) de cliente HTTP	-	2.2	Si
81	Valor a asignar al campo 'data' en el query string del GET (Alternativo a write_str 81)	-	2.2	Si

Las fuentes/destinos listados en estas tablas permiten enviar y recibir de un servidor web usando un cliente HTTP. Los datos se envian en la URL del mensaje (método GET). Lo que se recibe es el cuerpo de la respuesta.

Lo primero que se debe hacer es configurar los datos del servidor al cual se va realizar la comunicación (URL y puerto).Si no se indica puerto se utilizará el puerto 80.

```
write_str 80,'m2m.exemys.com:80';
```

Luego deben cargar los datos a enviar en el query. Previamente se debe verifiar que el cliente HTTP este disponible para hace una conexión con read_io 82.

```
write_str 83,'campo=va&info=test&ver=version';
```

Si en cambio se usa write_str 81 el cLAN enviará el texto indicado en la variable 'data' dentro de la URL.

En PHP los datos recibidos pueden procesarse usando este código.

```
<?php;
    $a = $_GET["data"];
    echo $a;//Responde con una copia de los datos recibidos
?>
```

Para leer los datos de la respuesta se debe usar read_io 81 y read_str 81

Como ejemplo por favor mire el archivo *HTTPejemplo.sce* que puede descargar junto a los otros ejemplos de uso de script www.exemys.com/EjemplosDeScriptEnGRD

Encontra tambien un ejemplo avanzado *(HTTPenvioRegistros.sce)* que muestra como enviar los datos de la memoria de registro del cLAN a una página WEB.

Desde la versión 2.8 del cLAN se puede llamar a write_str 84 despues de write_str 80 para cargar la ruta/nombre de archivo

```
write_str 80,'m2m.exemys.com:80';

write_str 84,'lectura/1/index.html';
```

Cliente SMTP (solo cLAN)

read_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
95	Estado del cliente SMTP	0	-	2.2	Si

#	Estado de cliente

#	SMTP
0	IDLE
1	ENVIANDO
2	ENVIO OK
3	ERROR AL ENVIAR

write_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
89	Configurar URL y puerto de cliente SMTP (si no se indica el puerto se usa el 25)	-	2.2	Si
90	Configurar correo de remitente de cliente SMTP	-	2.2	Si
91	Configurar usuario de cliente SMTP	-	2.2	Si
92	Configurar contraseña de cliente SMTP	-	2.2	Si
93	Dirección de correo del destinatario de cliente SMTP	-	2.2	Si
94	Asunto del correo de cliente SMTP	-	2.2	Si
95	Cuerpo del correo y disparar el envio de cliente SMTP	-	2.2	Si

Las fuentes/destinos listados en estas tablas permiten enviar emails usando un cliente STMP.

El primer paso es configurar los datos del servidor SMTP que se va a usar y los datos del remitente.

```
write_str 89,'smtp.exemys.com:25';
write_str 90,'clan@exemys.com';#remitente;
write_str 91,'clan@exemys.com';#usuario;
write_str 92,'password';#palabra clave;
```

Luego cada vez que quiera enviar un correo debe indicar destinatario, asunto y cuerpo del email

```
write_str 93,'exemys@exemys.com';
write_str 94,'Asunto';
write_str 95,'Cuerpo';
```

Puede usar read_io 95 para ver el resultado del envio. No se puede enviar mas de un correo por vez.

Como ejemplo por favor mire el archivo *SMTPejemplo.sce* que puede descargar junto a los otros ejemplos de uso de script www.exemys.com/EjemplosDeScriptEnGRD

Cliente POP (solo cLAN)

read_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
195	Estado del cliente POP	0	-	2.2	Si

#	Estado de cliente SMTP
0	IDLE
1	RECIBIENDO
2	SE RECIBIO 1 O MAS EMAILS
3	NO HAY EMAIL RECIBIDOS
4	ERROR AL RECIBIR

write_io	Descripción	Indice	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
195	Iniciar conexión de cliente POP	0	-	2.2	Si

read_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
193	Remitente de email recibido por cliente POP3	-	2.2	Si
194	Asunto de email recibido por cliente POP3	-	2.2	Si
195	Cuerpo de email recibido por cliente POP3. Genera carga el proximo mensaje en bandeja	-	2.2	Si

write_str	Descripción	cLAN-XF V1	cLAN-XF V2	cLAN-MQ
189	Configurar URL y puerto del cliente POP (si no se indica el puerto se usa el 110)	-	2.2	Si
191	Configurar usuario de cliente POP	-	2.2	Si
192	Configurar contraseña de cliente POP	-	2.2	Si

Las fuentes/destinos listados en estas tablas permiten recibir emails usando un cliente POP3. La bandeja de entrada puede almacenar hasta 5 correos con una longitud máxima de 198 caracteres.

El primer paso es configurar los datos del servidor POP3 que se va a usar

```
write_str 189,'smtp.mexemys.com:110';
write_str 191,'clan@exemys.com';
write_str 192,'password';
```

Con read_io 195 puede saber si se recibieron correos

```
write_io 195,0,0;
```

Puede usar read_io 95 para ver el resultado de la recepción.

```
read_io 195,e,0;
```

Si hay correos en la bandeja de entrada los puede leer con read_str 193,194 y 195. Al ejecutar read_str 195 se carga el próximo correo de la bandeja de entrada.

```
read_str 193,i,z;#remitente;
read_str 194,i,x;#asunto;
read_str 195,i,y;#cuerpo;
```

Como ejemplo por favor mire los archivos *POP3ejemplo.sce* y *POP3fast.sce* que puede descargar junto a los otros ejemplos de uso de script www.exemys.com/EjemplosDeScriptEnGRD

MQTT Estado y publicación (GRD-MQ y cLAN-MQ)

read_io	Descripción	Indice	GRD-MQ	cLAN-MQ
1001	MQTT-Estado de conexión con el broker (0=no conectado, 1=conectado)	0	Si	Si

Desde el script de pueden públicar mensajes mas allá de lo configurado para reportes e históricos.

write_str	Descripción	GRD-MQ	cLAN-MQ
1001	MQTT-Cargar tópico a publicar	Si	Si
1002	MQTT-Cargar payload a publicar y hacerlo	Si	Si

Ejemplo:

```
read_io 1001,h,0;

if h=1 {

   write_str 1001,'v1/devices/me/telemetry';

   write_str 1002,'{A1:52}';

};
```

MQTT recepción de suscripciones (GRD-MQ y cLAN-MQ)

Desde el script de pueden recibir los mensajes de las suscripciones realizadas en el "GRD Config" en la sección MQTT

Se pueden suscribir hasta 10 tópicos.

Los mensajes quedan en una cola de recepción y deben ser leidos de a uno desde el script.

read_io	Descripción	Indice	GRD-MQ	cLAN-MQ
1000	MQTT-Cantidad de mensajes recibidos por suscripciones pendientes de lectura	0	Si	Si

read_str	Descripción	GRD-MQ	cLAN-MQ
1000	MQTT. Lee primer mensaje en cola de suscripciones	Si	Si

Ejemplo:

```
read_io 1000,b,0; b

if b!0 {

   read_str 1000,c,z;

};
```

z contiene el texto recibido

2020-11-09