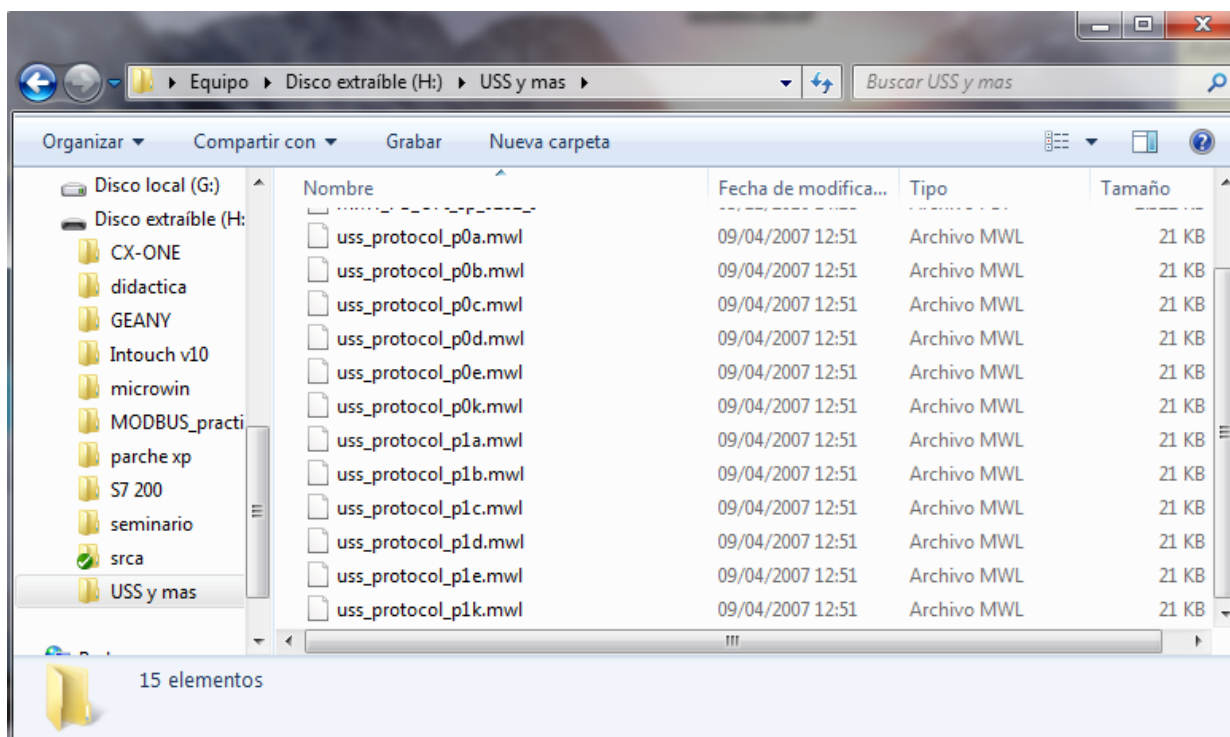


# MANUAL DE COMUNICACIÓN MO- DBUS\_RTU S7-200

El presente documento es aplicable en el ámbito de comunicación Master - Slave (Maestro - Esclavo) y se explicará con el máximo de detalles posible con tal de lograr ejecución y funcionamiento lo más sencilla posible tanto en el programa MicroWin S7-200, como en el Geany y como con el lenguaje Phytton.

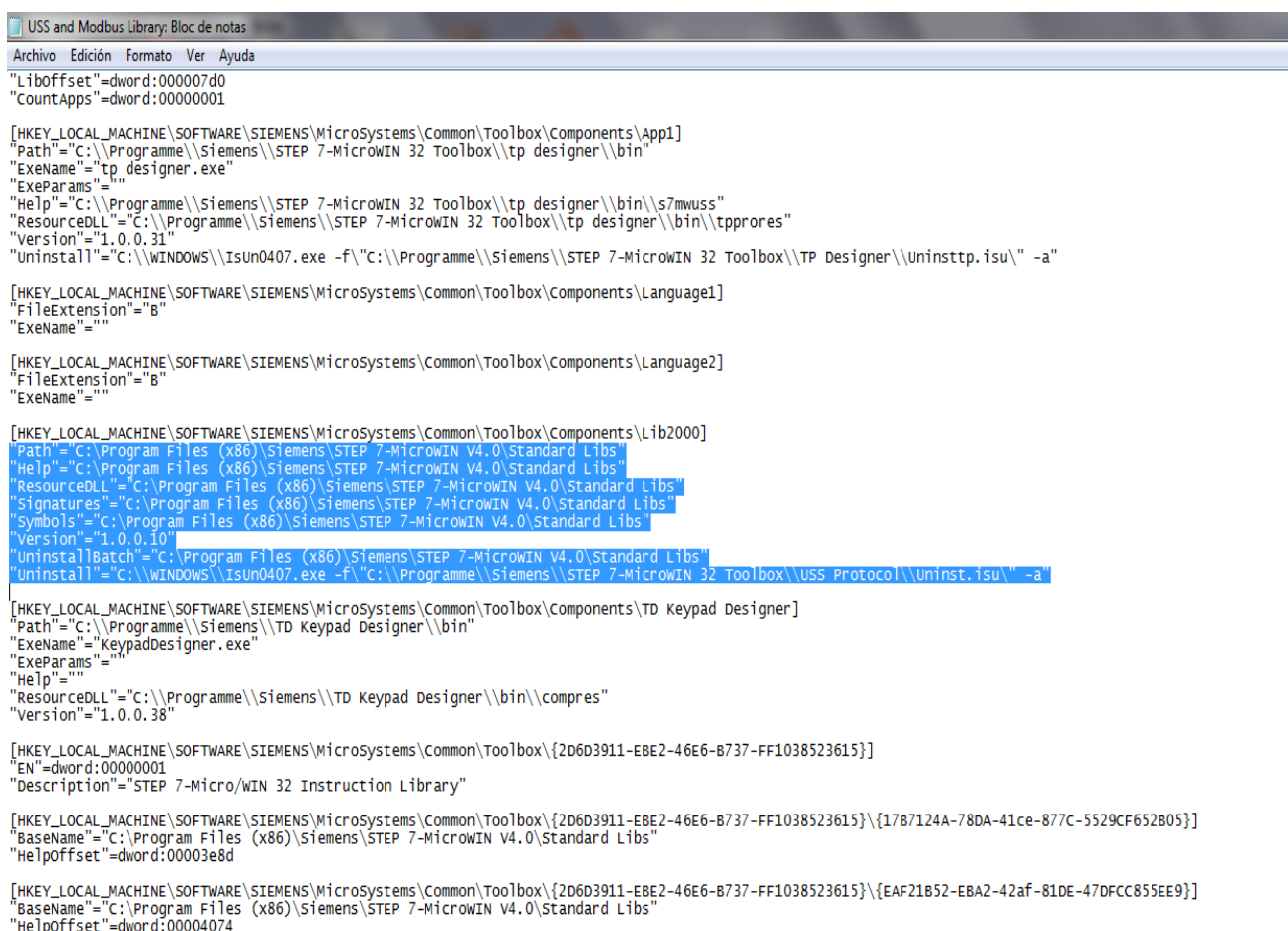
## 1.- Instalación de las librerías para S7- 200

El primer paso que se deberá realizar será la instalación de las librerías de ModBus para poder hallar sus opciones en el programa MicroWin S7-200. Estas librerías se encuentran en la carpeta USS y más. Uno de los problemas es que estas librerías solo funcionan con dos sistemas operativos, ambos de la compañía Microsoft, y son sus modelos Windows xp y Windows 7 , aunque se ha comprobado que en este último hay problemas en cuanto al uso de estas librerías y compatibilidad con el software MicroWin S7-200.



Una vez instaladas las librerías, dentro de la carpeta anteriormente citada (USS y mas), habrá otra carpeta denominada como USS and ModBus library.

A continuación se deberá entrar en esa carpeta, abriendo el archivo con el bloc de notas o similar, una vez completado ese paso cambiaremos la dirección que aparece por defecto, por la dirección donde está instalada la carpeta Standlib en el ordenador que esté utilizando para ello. Una vez realizada la operación ya habremos cambiado la ruta.



```
USS and Modbus Library: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
"LibOffset"=dword:000007d0
"CountApps"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\Components\App1]
"Path"="c:\Programme\Siemens\STEP 7-MicrowIN 32 Toolbox\tp designer\bin"
"ExeName"="tp_designer.exe"
"ExeParams"=""
"Help"="c:\Programme\Siemens\STEP 7-MicrowIN 32 Toolbox\tp designer\bin\s7mwuss"
"ResourceDLL"="c:\Programme\Siemens\STEP 7-MicrowIN 32 Toolbox\tp designer\bin\tpprores"
"Version"="1.0.0.31"
"Uninstall"="c:\WINDOWS\IsUn0407.exe -f\"c:\Programme\Siemens\STEP 7-MicrowIN 32 Toolbox\TP Designer\Uninstp.isu\" -a"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\Components\Language1]
"FileExtension"="B"
"ExeName"=""

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\Components\Language2]
"FileExtension"="B"
"ExeName"=""

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\Components\Lib2000]
"Path"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"Help"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"ResourceDLL"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"Signatures"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"Symbols"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"Version"="1.0.0.10"
"UninstallBatch"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"Uninstall"="c:\WINDOWS\IsUn0407.exe -f\"c:\Programme\Siemens\STEP 7-MicrowIN 32 Toolbox\USS Protocol\Uninst.isu\" -a"

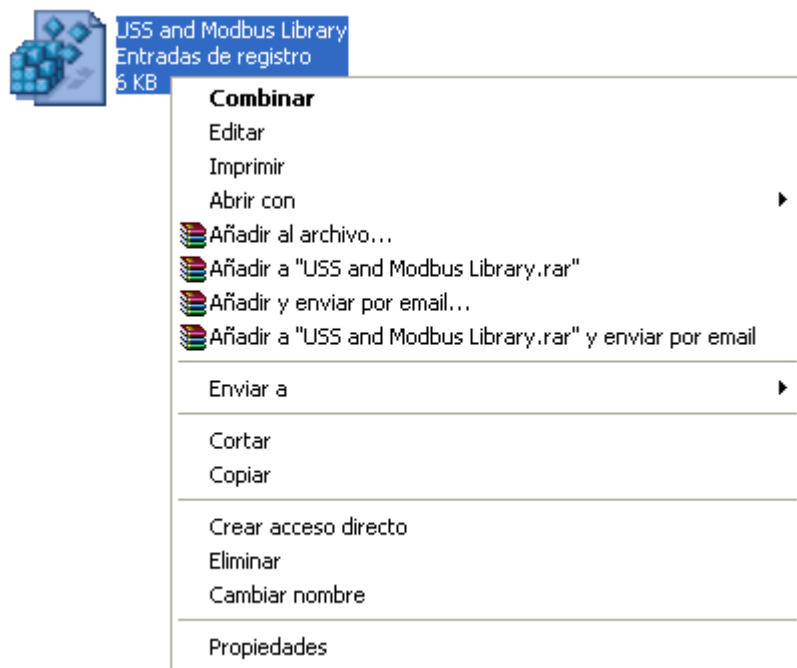
[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\Components\TD Keypad Designer]
"Path"="c:\Programme\Siemens\TD Keypad Designer\bin"
"ExeName"="KeypadDesigner.exe"
"ExeParams"=""
"Help"=""
"ResourceDLL"="c:\Programme\Siemens\TD Keypad Designer\bin\compres"
"Version"="1.0.0.38"

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\{2D6D3911-EBE2-46E6-B737-FF1038523615}]
"EN"=dword:00000001
"Description"="STEP 7-Micro/WIN 32 Instruction Library"

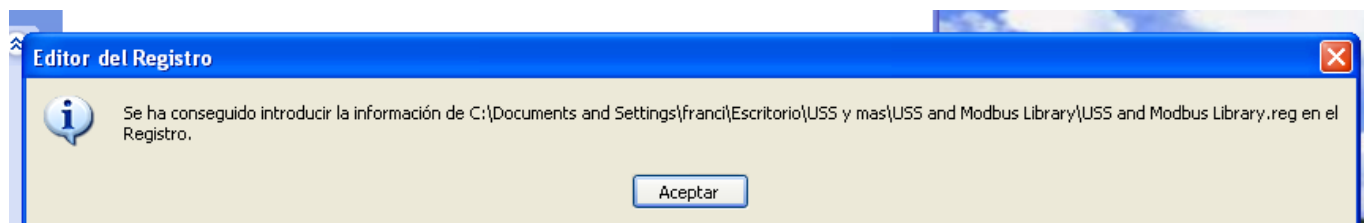
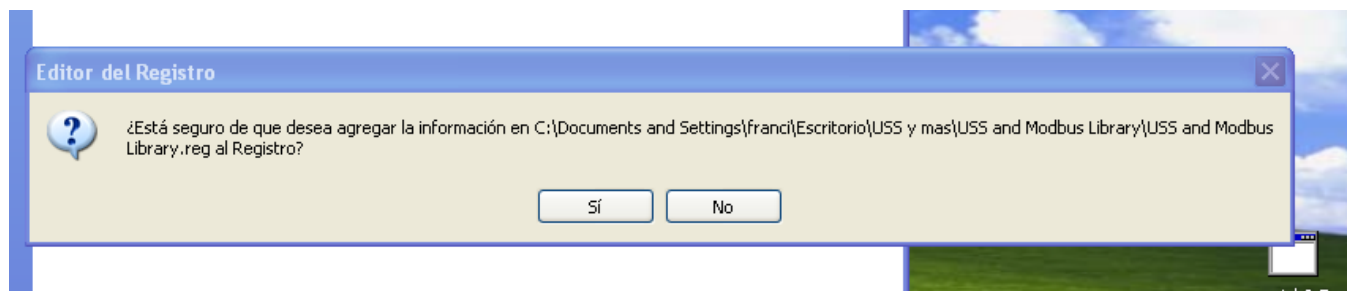
[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\{2D6D3911-EBE2-46E6-B737-FF1038523615}\{17B7124A-78DA-41ce-877C-5529CF652B05}]
"BaseName"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"HelpOffset"=dword:00003e8d

[HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\MicroSystems\Common\Toolbox\{2D6D3911-EBE2-46E6-B737-FF1038523615}\{EAF21B52-EBA2-42af-81DE-47DFCC855EE9}]
"BaseName"="c:\Program Files (x86)\Siemens\STEP 7-MicrowIN V4.0\Standard Libs"
"HelpOffset"=dword:00004074
```

Una vez establecida la ruta, lo siguiente será volver a darle con el botón derecho al archivo USS y and ModBus library y esta vez darle a la opción combinar.



Seguido de la selección anterior será mostrado el siguiente cuadro en el que habrá que clicar en “Sí”, y así ya estarán disponibles las librerías ModBus para la comunicación (Master – Slave).



## 2.- PROGRAMA MICROWIN S7-200

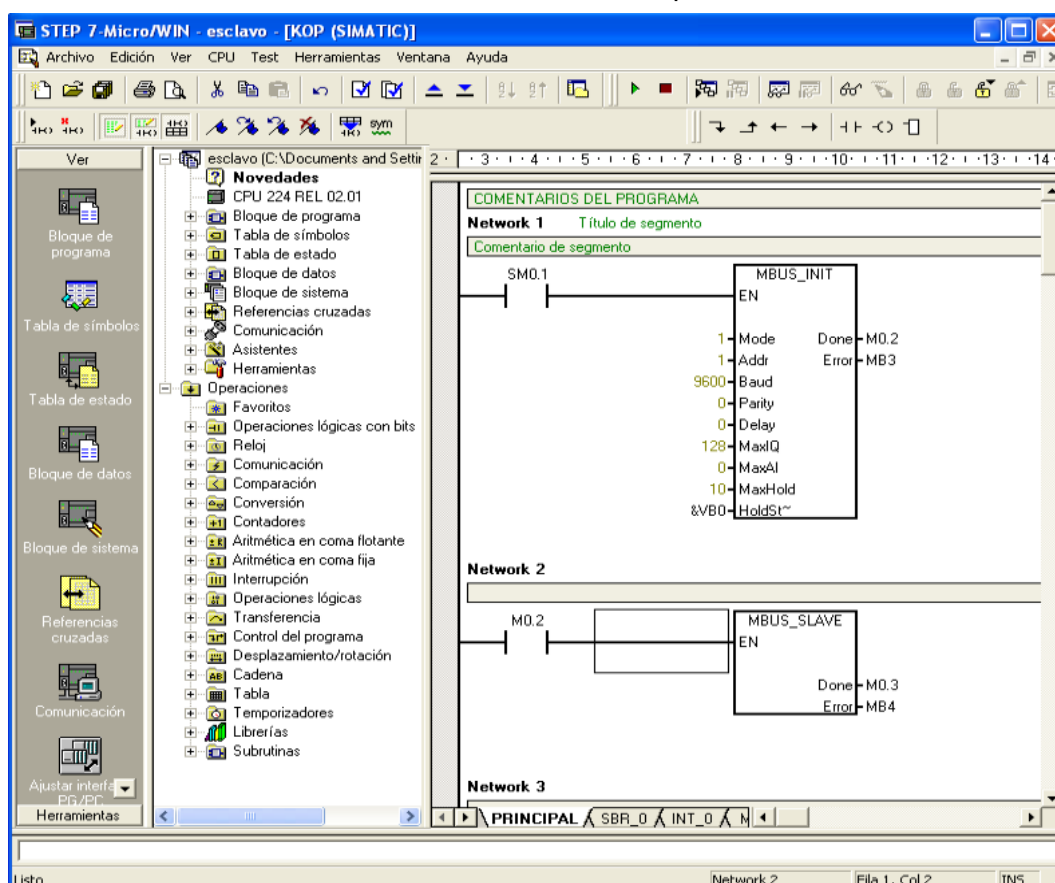
El siguiente paso será la creación de un proyecto en el programa MicroWin en el que utilizaremos las funciones necesarias para la comunicación ModBus.

### Dichas funciones son:

**MBUS\_INIT:** Esta función sirve para establecer el tipo de conexión y la configuración según el tipo del programa a utilizar. Dicha función se configurará o activará con un solo pulso y podrá ser modificada y llevada a la configuración normal que tuviera antes si el automata se pusiera a stop.

En este caso solo se ejecutará en el primer ciclo. Si se ejecuta sin errores la salida se activará (Done) y si hubiera algún error este sería mandado al área de memoria. A continuación se explicará con detalles el resto de parámetros.

**MBUS\_SLAVE:** Esta función se utiliza para procesar una petición del maestro, se debe ejecutar en todos los ciclos para poder comprobar y responder al maestro ModBus. A continuación se muestra la ventana de MicroWin correspondiente a ambas funciones.



## Parámetros de las funciones:

### Función MBUS\_INIT:

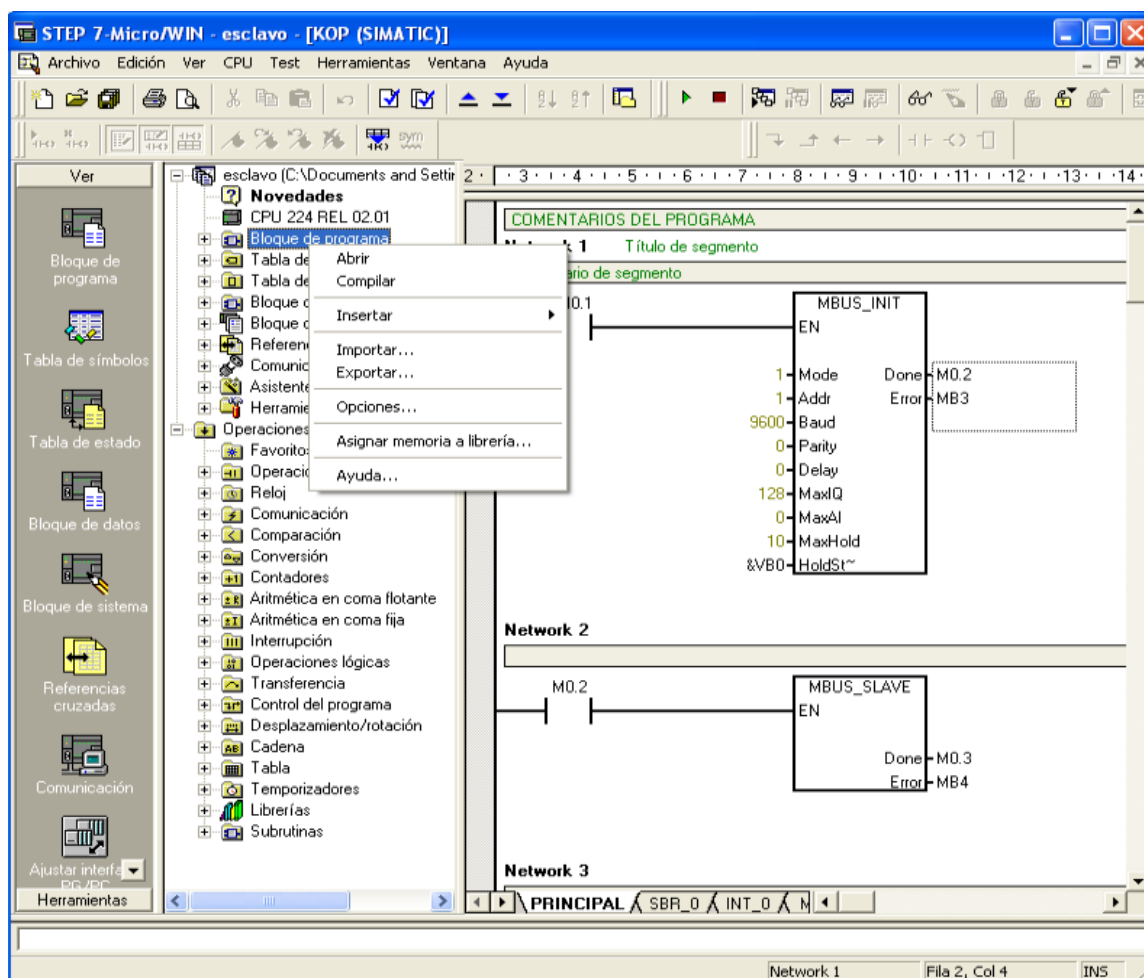
- EN. Ejecuta la función.
- Mode. Si esta entrada es igual a 1 el puerto 0 se asignará al protocolo modbus y si es igual a 0 el puerto 0 se asignará a PPI y se inhibirá el protocolo modbus.
- Boud. Velocidad de transferencia bit/s.
- Addr. Ajusta la dirección a valores entre 1 y 247.
- Parity. Este parámetro tiene que ser igual con la paridad del maestro.
- Delay. Este parámetro retarda el timeout de fin de mensaje.
- MaxIQ. Ajusta el numero de entradas y salidas digitales .
- MaxAI. Este parámetro ajusta el número de entradas analógicas disponibles para la dirección modbus.
- MaxHold. Numero de registros de memorias.
- HoldStart. Es la dirección inicial de los registros de retención en la memoria .
- Done. Se activa una vez finalizada la operación MBUS\_INIT.
- Error. Contiene el resultado de la operación.

### Función MBUS\_SLAVE:

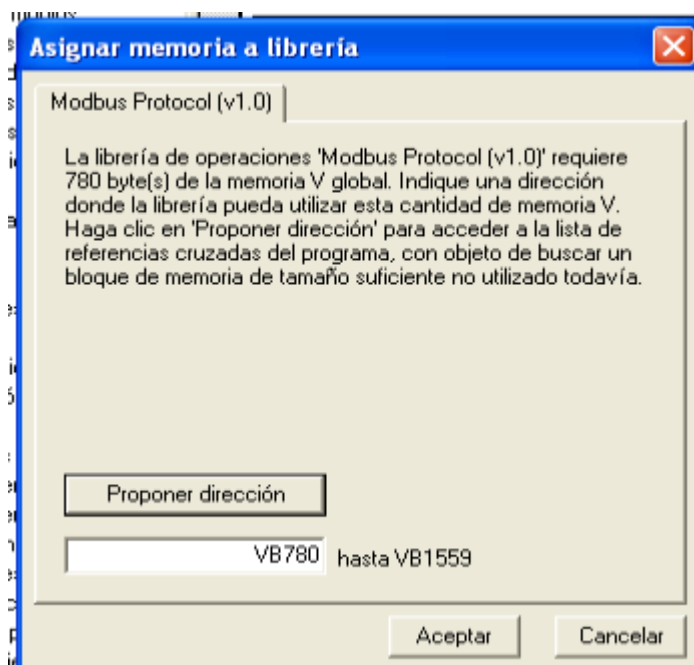
- EN. Ejecuta la operación en cada ciclo cuando este activado este parámetro.
- Done. Este parámetro se activa cuando la operación MBUS\_SLAVE responde a una petición del maestro y se desactiva si no se ha procesado ninguna petición.
- Error. Contiene el resultado de ejecución de la operación.

Una vez hecho el programa ahora tendremos que reservar direcciones de memoria para las funciones de ModBus, por tanto debemos ir a la parte izquierda de la ventana del programa, que es donde se encuentran todas las opciones y una vez ahí haremos click con el botón

derecho sobre la opción “Bloque de Programa” y seleccionando a continuación “Asignar memorias a librerías”.



Entramos y le daremos un click a la opción proponer dirección. Una vez realizado este paso el programa automáticamente asignará un bloque de memoria suficiente para soportar los 780 Bytes de la memoria global. A partir de este proceso habremos terminado con la parte correspondiente al programa en MicroWin.



Hay que tener en cuenta que hay que cerrar el microwin ya que al abrir el maestro entran en conflicto un programa con otro.

### 3.-PROGRAMACIÓN MAESTRO CON PHYTON.

Para que el programa funcione y se ejecute correctamente habrá que instalar la librería (serial py) para ello basta con ejecutarla. Otra librería necesaria para llevar a cabo la programación es la librería ModBus Tk, por tanto para que funcione correctamente lo único a tener en cuenta será que la dirección de la librería sea la misma que la del archivo RTU\_Master\_ example, si ambas no son coincidentes el programa no responderá como deseamos.

Una vez procesado estos pasos el programa hecho en Geany mediante el lenguaje en Phyton estará listo para compilarse y ejecutarse para su correcto funcionamiento como simulador de un maestro ModBus\_RTU y poder comunicarnos con uno o más autómatas con la función de esclavo.

#### PROGRAMACIÓN EN PHYTON:

```
1  #!/usr/bin/env python
2  # -*- coding: utf_8 -*-
3  """
4  Modbus TestKit: Implementation of Modbus protocol in python
5  """
6  (C)2009 - Luc Jean - luc.jean@gmail.com
7  (C)2009 - Apidev - http://www.apidev.fr
8
9  This is distributed under GNU LGPL license, see license.txt
10 """
11
12 import sys
13 import serial
14
15 #add logging capability
16 import logging
17
18 import modbus_tk
19 import modbus_tk.defines as cst
20 import modbus_tk.modbus_rtu as modbus_rtu
21 import time
22 logger = modbus_tk.utils.create_logger("console")
23
24 if __name__ == "__main__":
25     try:
26         #Connect to the slave
27         master = modbus_rtu.RtuMaster(serial.Serial(port=3, baudrate=9600, parity='N'))
28         master.set_timeout(1.0)
29         master.set_verbose(True)
30         print("connected")
31
32         #send some queries
33         print(master.execute(1, cst.READ_COILS, 0, 8))
34         time.sleep(0.01)
35         print(master.execute(1, cst.READ_DISCRETE_INPUTS, 0, 8))
36         #print(master.execute(1, cst.READ_INPUT_REGISTERS, 100, 3)) leer entradas analógicas
37         #print(master.execute(1, cst.READ_HOLDING_REGISTERS, 100, 12)) leer areas de memorias
38         #print(master.execute(1, cst.WRITE_SINGLE_COIL, 7, output_value=1)) escribir una salida
39         #print(master.execute(1, cst.WRITE_SINGLE_REGISTER, 100, output_value=54)) escribe palabras de 16 bit con la direccion que desees
40         #print(master.execute(1, cst.WRITE_MULTIPLE_COILS, 0, output_value=[1, 1, 0, 1, 0, 1, 1])) escribir multiples salidas
41         #print(master.execute(1, cst.WRITE_MULTIPLE_REGISTERS, 100, output_value=xrange(12))) escribir mas de un registro
42
43     except modbus_tk.modbus.ModbusError, e:
44         logger.error("%s- Code=%d" % (e, e.get_exception_code()))
45
46
47
```

Este es el programa hecho en phyton que simula un master RTU.

En nuestro caso el programa de python nos fue entregado por J.L.Redrejo profesor de Comunicación Industrial en el I.E.S. Santiago Apóstol con lo cual solo ha habido que realizar unas pequeñas modificaciones para su correcto funcionamiento.

### **A continuación serán explicadas las funciones necesarias en python:**

En las líneas mostradas a continuación se realiza la configuración del maestro.

```
#Connect to the slave
```

```
master = modbus_rtu.RtuMaster(serial.Serial(port=3, baudrate=9600, parity='N'))
```

```
master.set_timeout(1.0)
```

master.set\_verbose(True) “no afecta a la comunicación solo es para mostrar mas o menos sms”

```
print("connected")
```

Los parámetros a tener en cuenta han sido los siguientes:

El puerto será uno inferior al que le hayamos otorgado al esclavo al ser conectado mediante el puerto serie. Recordar que hay que mirar en administrador de dispositivos que puerto nos ha dado y poner en el master uno menos.

La velocidad de transferencia será 9600, y la paridad será igual que la del esclavo.

En la siguiente línea lo que hace el master es esperar un segundo peticiones del esclavo “master.set\_timeout(1,0)”.

En nuestro programa solo hemos leído el estado de las entradas y salidas del autómata (líneas de abajo) y en medio de las dos lecturas se pondrá un time.sleep para dar tiempo a procesar los datos ya que el programa hecho en el geany que compila más rápido de lo que puede hacerlo el autómata.

```
print(master.execute(1, cst.READ_COILS, 0, 8))
```

```
time.sleep (0.01)
```

```
print(master.execute(1, cst.READ_DISCRETE_INPUTS, 0, 8))
```



El programa también incluye más funciones que aunque en el ejemplo no han sido usadas, se explicaran a continuación de manera resumida:

```
#print(master.execute(1, cst.READ_INPUT_REGISTERS, 100, 3))
```

#### **leer entradas analógicas**

```
#print(master.execute(1, cst.READ_HOLDING_REGISTERS, 100, 12))
```

#### **leer areas de memorias**

```
#print(master.execute(1, cst.WRITE_SINGLE_COIL, 7, output_value=1))
```

#### **escribir una salida**

```
#print(master.execute(1, cst.WRITE_SINGLE_REGISTER, 100, output_value=54))
```

#### **escribe palabras de 16 bit con la direccion que deseess**

```
#print(master.execute(1, cst.WRITE_MULTIPLE_COILS, 0, output_value=[1, 1, 0, 1, 1, 0, 1, 1]))
```

#### **escribir multiples salidas**

```
#print(master.execute(1, cst.WRITE_MULTIPLE_REGISTERS, 100, output_value=xrange(12)))
```