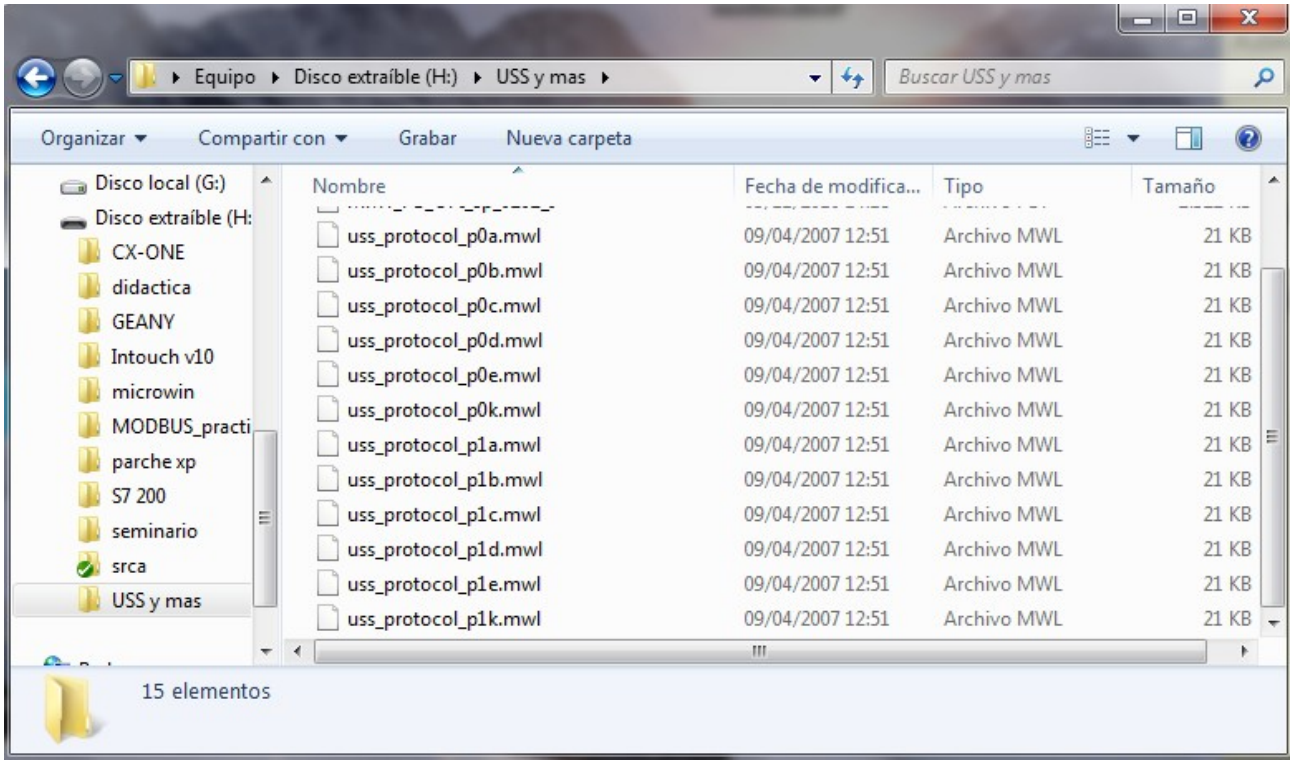
MANUAL DE COMUNICACIÓN MODBUS_RTU S7-200

El presente documento es aplicable en el ámbito de comunicación Master- Slave y se explicará con el máximo de detalles posible para su posible ejecución y funcionamiento tanto en el programa MicroWin S7-200, como en el Geany con el lenguaje Phyton.

1º- Instalación de las librerias para S7- 200.

El primer paso a realizar será la instalación de las librerías ModBus para poder tener las opciones Modbus en el programa MicroWin S7-200. Estas librerías se encuentran en la carpeta USS y más. Uno de los problemas es que estas librerias solo funcionan con el window xp, ya que se ha comprobado que hay problemas con el window 7.



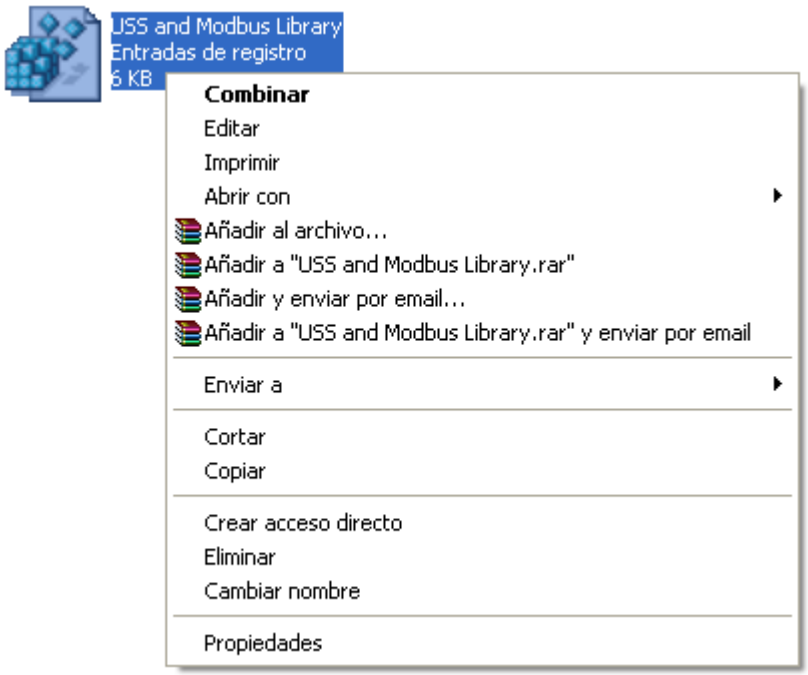
Una vez instaladas las librerías, dentro de la carpeta anteriormente citada, habrá otra carpeta que se llama USS and ModBus library. Al entrar en esta carpeta, abriremos el archivo con el blog de notas y cambiaremos la dirección que aparece por defecto, por la dirección donde está instalada la carpeta Standlib en nuestro ordenador. Una vez realizada la operación ya habremos cambiado la ruta.

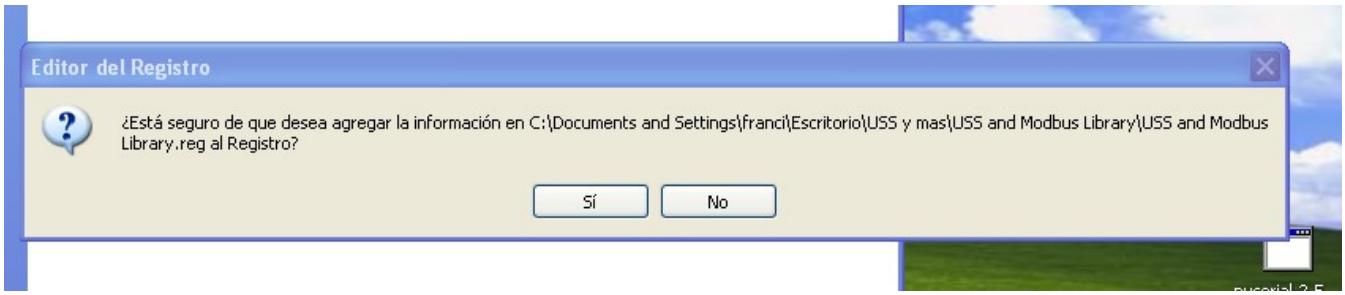
USS and Modbus Library: Bloc de notas

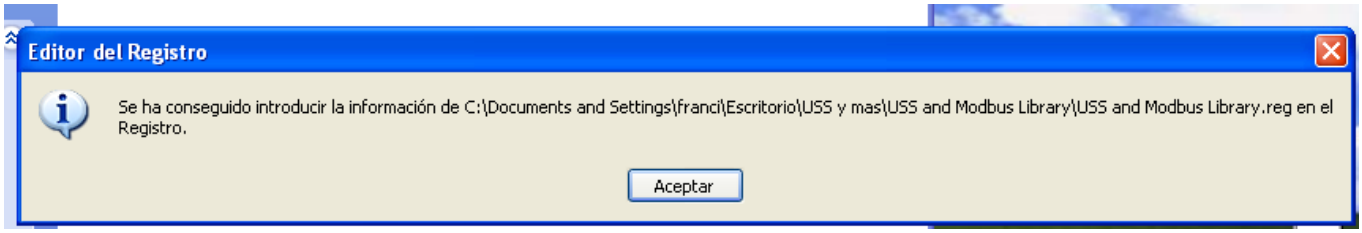
Archivo Edición Formato Ver Ayuda

```
"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\\Unisttp.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. Aceptamos y a partir de entonces, estarán disponibles las librerias ModBus para la comunicación (Master – Slave).







2º- PROGRAMA MICROWIN S7-200

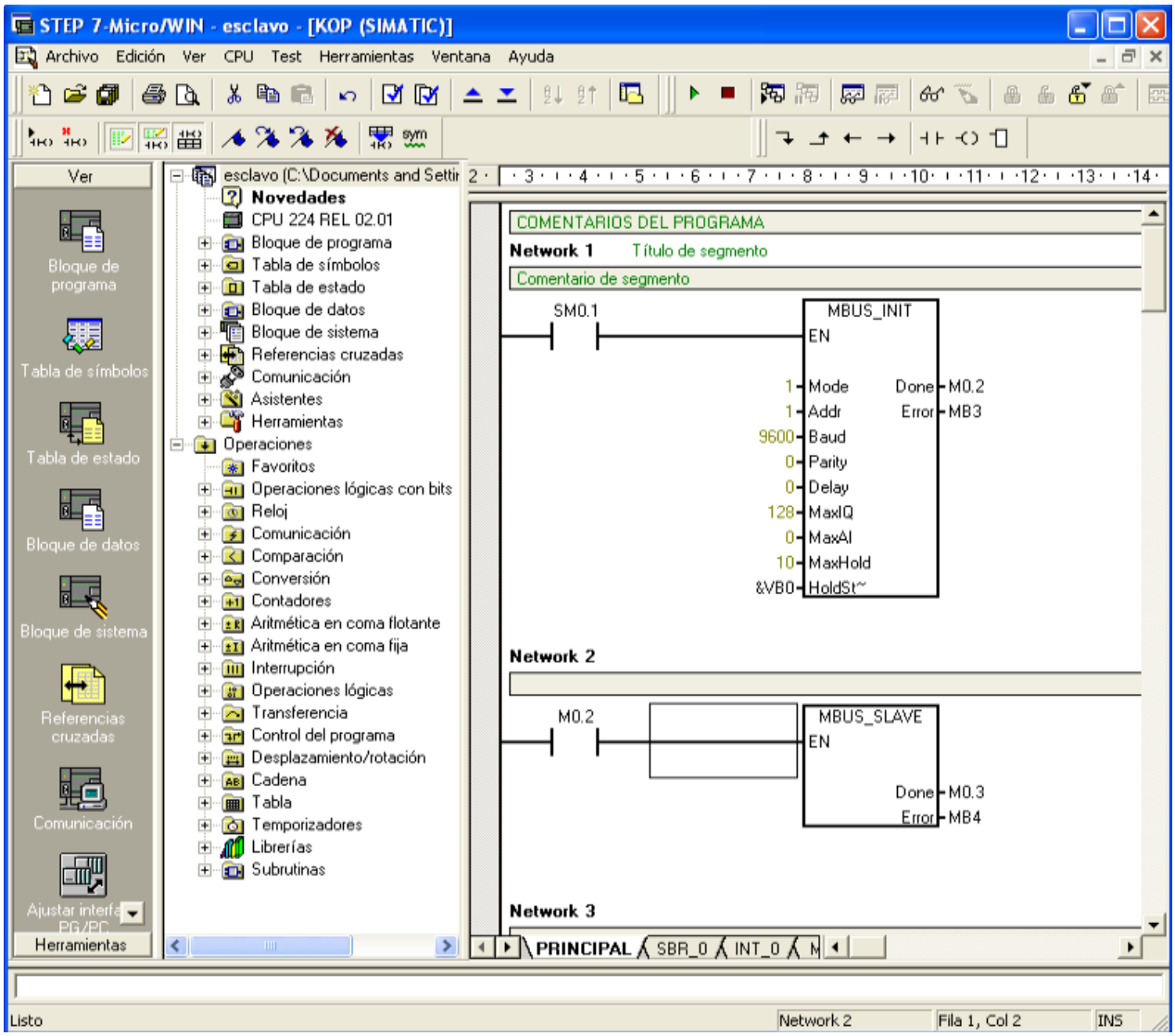
El siguiente paso será la creación de un proyecto en el programa MicroWin en el que utilizaremos las funciones necesarias para la comunicación ModBus.

Estas son las siguientes:

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.



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:

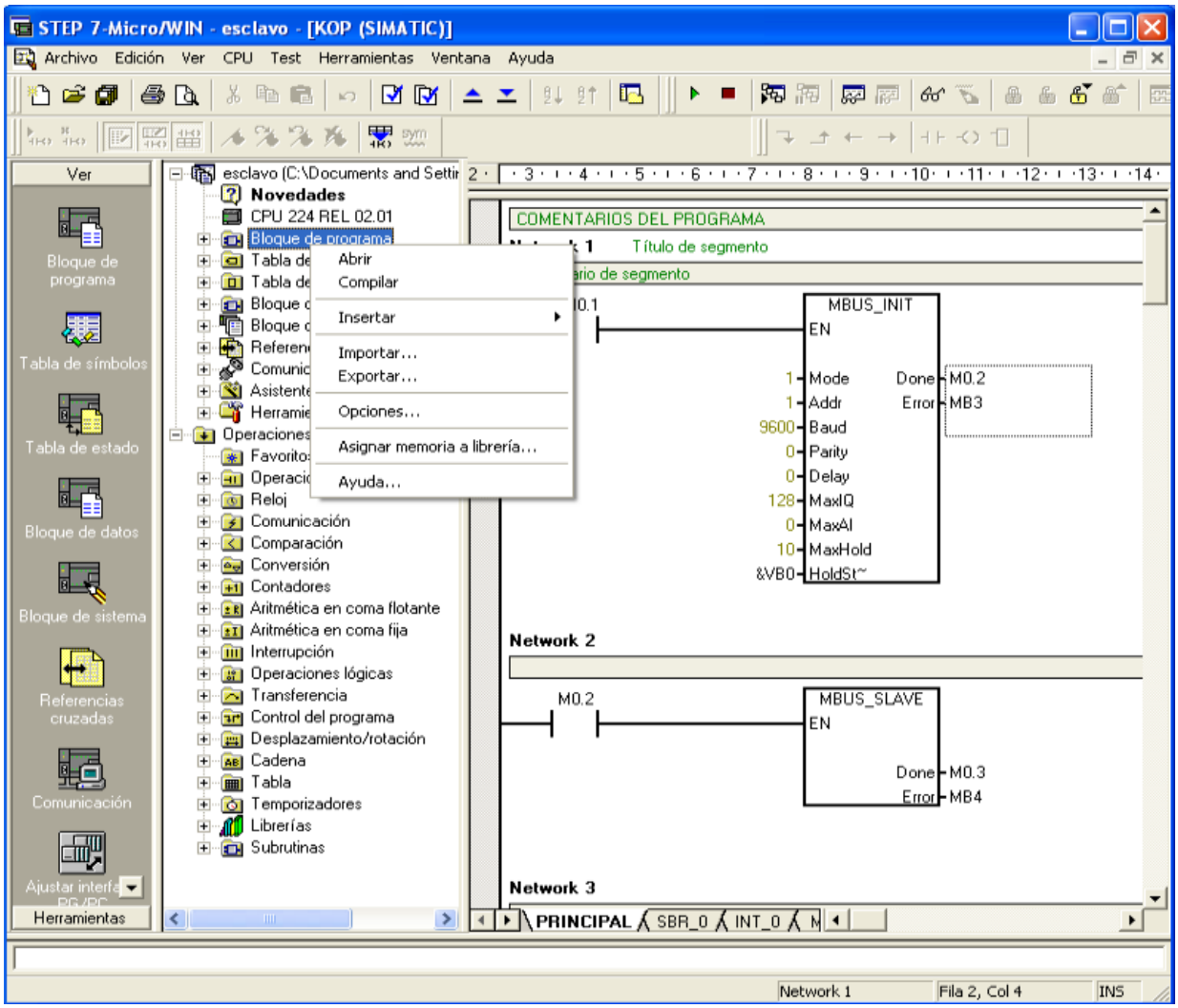
- EN. Ejecuta la operación en cada ciclo cuando este activado este parámetro.

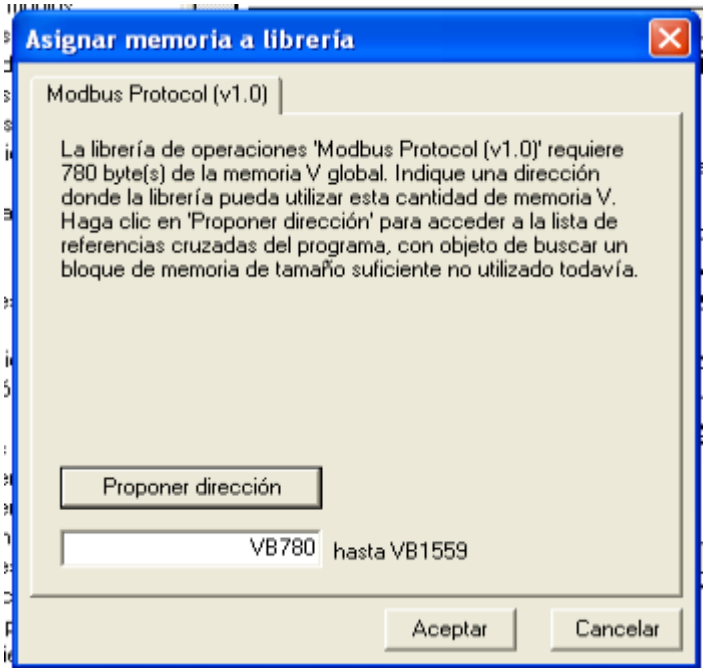
- Done. Este parámetro se activa cuando la operación MBUS_SLAVE responde a una petición del maestro y se desactiva si no se ha procesado ninguna petición.

- Error. Contiene el resultado de ejecución de la operación.

Una vez hecho el programa ahora tendremos que reservar direcciones de memoria para las funciones ModBus, por tanto nos iremos a la parte izquierda del programa, que es donde se encuentran todas las opciones y le daremos con el botón derecho a la opción Bloque de Programa y le daremos a (Asignar memorias a librerias). 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 el programa en MicroWin.

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

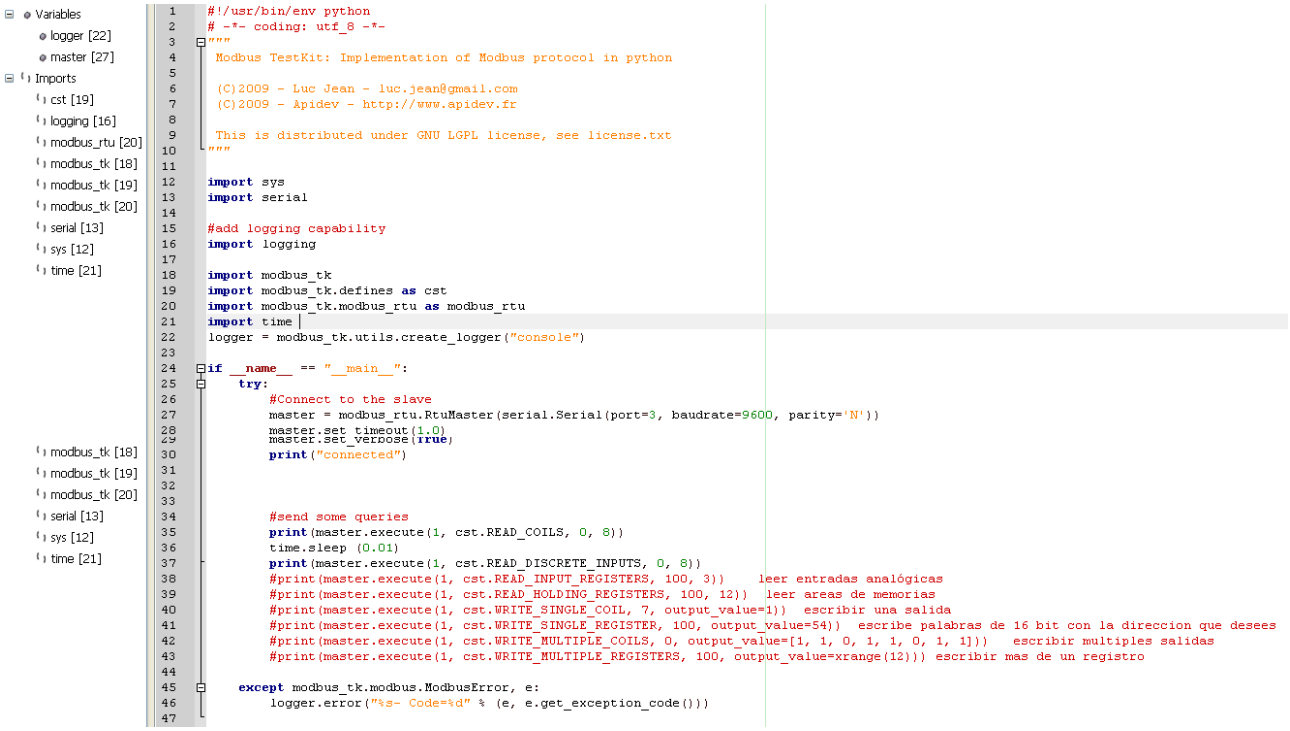




3º-PROGRAMACIÓN MAESTRO CON PHYTON.

Para que el programa funcione y se ejecute correctamente habrá que instalar la librería (seria py) lo ejecutamos y ya está instalado. Otra librería necesaria a utilizar el 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. Una vez procesado estos pasos el programa hecho Geany con 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:



Este es el programa hecho en phyton que simula un master RTU.
En nuestro caso el programa nos lo dió el profesor de Comunicación Industrial y solo ha hecho falta realizar una pequeña modificación. Esta ha sido, añadir una librería "(Import time)" que ha sido utilizada para poder incluir bases de tiempo en el programa.

A continuación se explicaran las funciones necesarias:

#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")

En las anteriores lineas de programa, lo que hemos hecho es configurar el maestro.
Los parámetros a tener en cuenta han sido que el puerto sea uno inferior al que nos haya dado el esclavo al conectarlo por 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 linea 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 sálidas del autómata (lineas 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 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 desees
```
#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)))
```
escribir mas de un registro