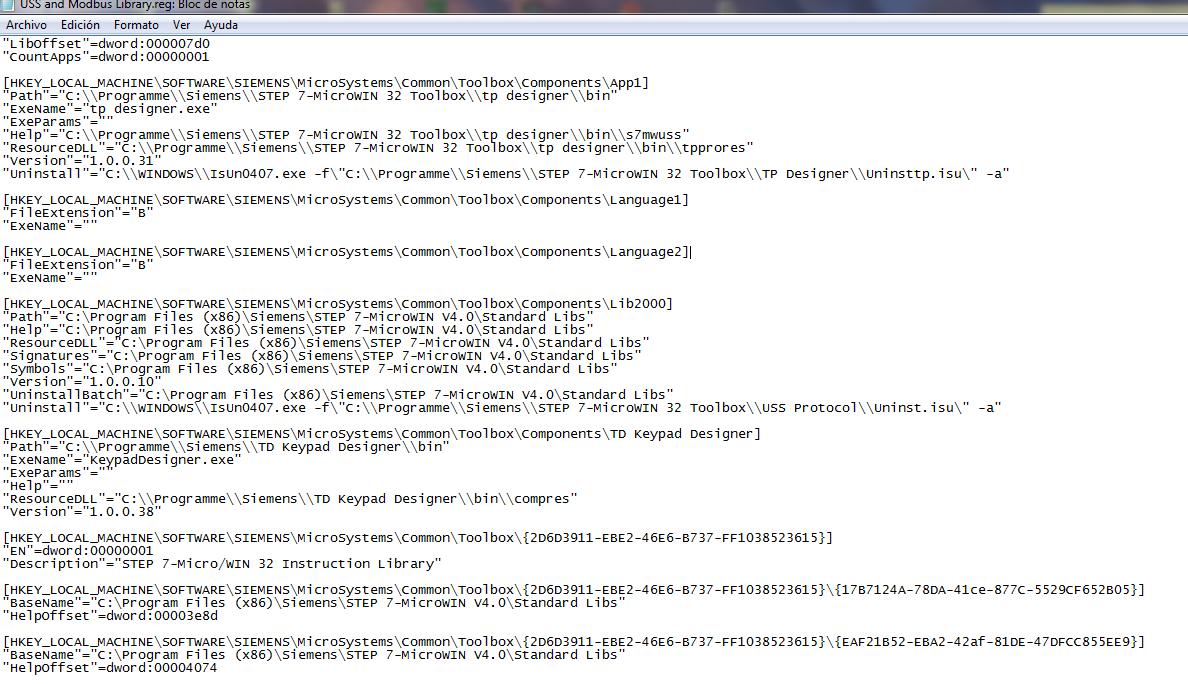
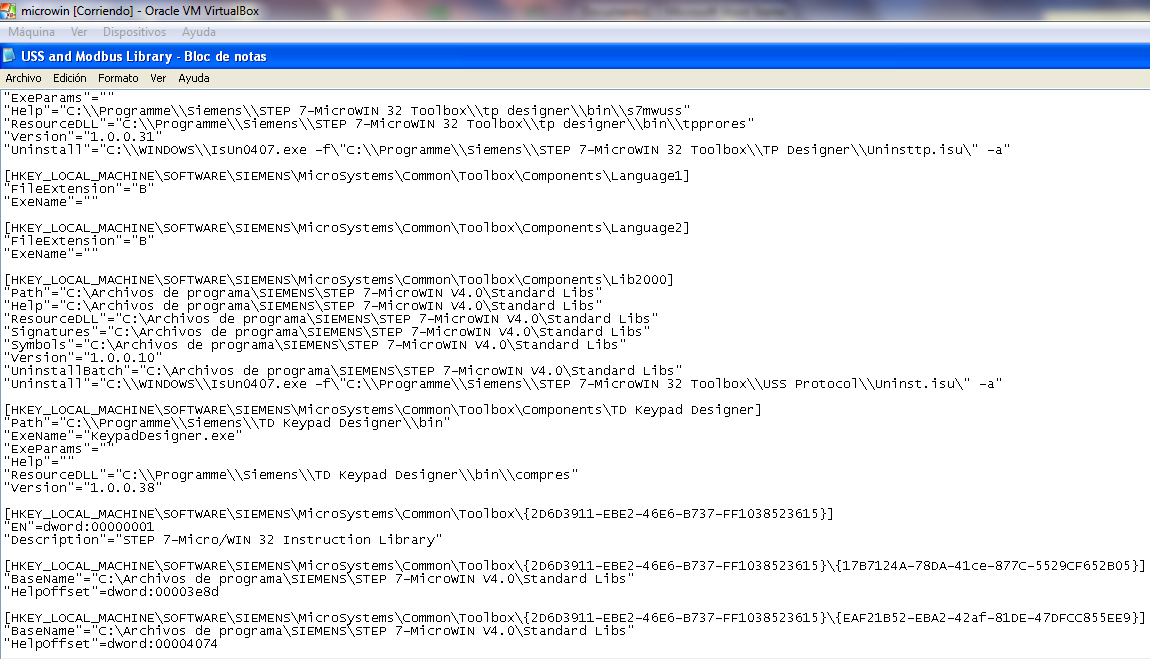
En este documento se explicará todos los pasos que se debe seguir para realizar una comunicación entre uno o más esclavos s7-200 con un simulador de maestro programado en python, la comunicación se realizará a través del protocolo modbus rtu.

**PRIMER PASO.** Lo primero que se deberá hacer será instalar las librerías modbus para microWIN. Para ello debemos abrir la carpeta “USS y mas” a continuación abriremos la carpeta “USS and Modbus Library” y abriremos el archivo con el blog de nota o con un editor de texto.



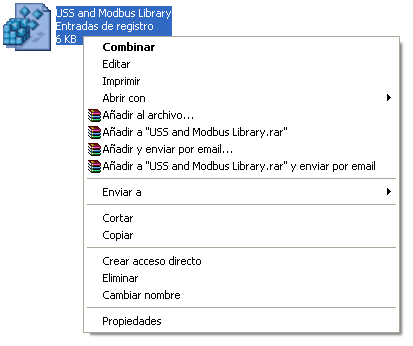
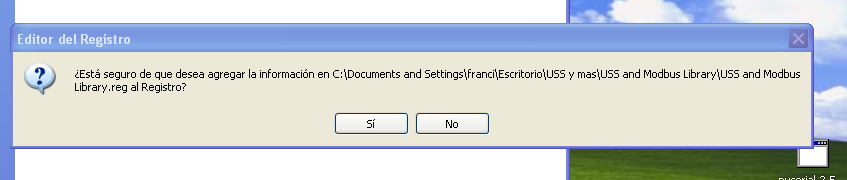
El siguiente paso será cambiar la ruta de Standard Libs en todas las líneas donde aparezca. La ruta que introduciremos será donde este situada nuestra carpeta de Standard libs en el disco duro.

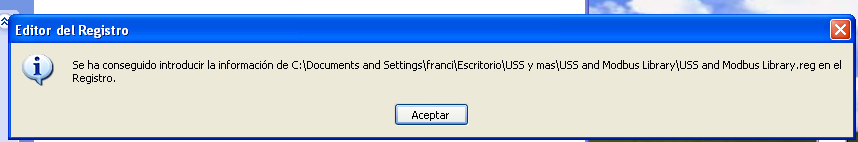
Ejemplo:



Como podemos observar en todas las rutas donde teníamos la ruta "C:\Program Files (x86)\Siemens\STEP 7-MicroWIN V4.0\Standard Libs" la hemos cambiado por la ruta "C:\Archivos de programa\SIEMENS\STEP 7-Microwin V4.0\Standard Libs”.

Después de haber cambiado la ruta guardamos el documento y lo cerramos. A continuación le damos con el botón derecho a combinar y agregamos la información al registro pulsando ‘Si’ y a continuación a aceptar. Con esto ya tendremos las librerías modbus necesarias para los esclavos instaladas.

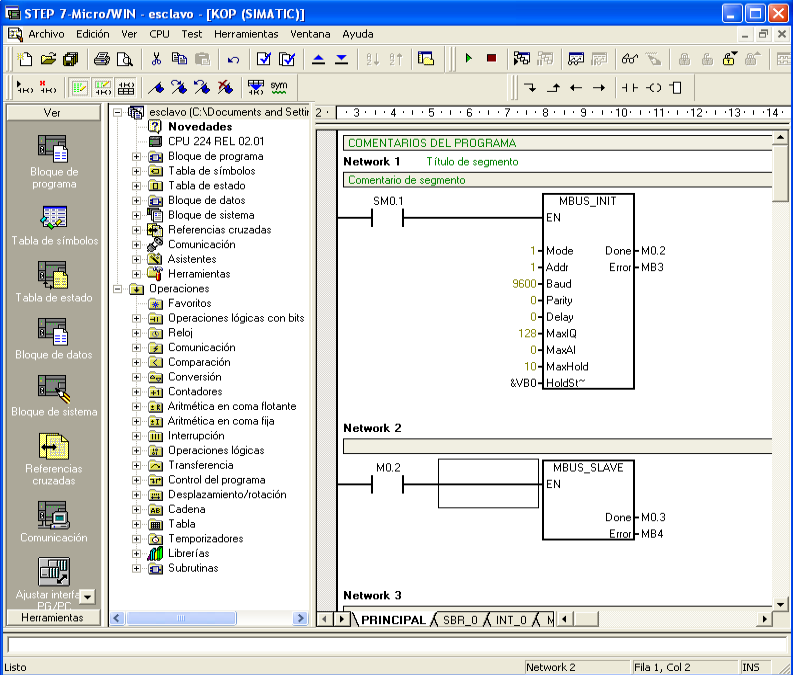


**SEGUNDO PASO.** Crearemos un nuevo proyecto en el MicroWIN y vamos a utilizar las dos funciones para la comunicación modbus.

MBUS\_INIT, esta función sirve para inicializar, activar o desactivar la comunicación y solo se ejecutará dependiendo de las necesidades de nuestro programa, en nuestro programa solo lo ejecutaremos en el primer ciclo. Si se ejecuta sin errores el bit done se a 1.

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. En nuestro caso la función la ejecutaremos si no ha dado ningún error la inicialización o activación ya que utilizaremos el contacto del DONE de la función MBUS\_INIT para ejecutar esta función.

Programa de ejemplo:



Descripción de los contactos y los parámetros de las funciones.

Contactos.

SM0.1- Se utiliza este contacto ya que solo da un pulso en el primer ciclo del programa ya que solo necesitamos en este programa que la función MBUS\_INIT se ejecute al inicio del programa una vez.

M0.2- Se utiliza este contacto ya que se activará si la función MBUS\_INIT se ha ejecutado correctamente. Así que el contacto se activará y la función MBUS\_SLAVE estará preparada para comprobar y responder las peticiones del maestro. No podremos usar la salida Q0.0.

Parámetros de las funciones.

Función MBUS\_INIT:

- EN. Cuando se activa este parámetro se 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. Este parámetro ajusta la velocidad de transferencia bit/s. En nuestro caso será de 9600.

-Addr. Ajusta la dirección a valores entre 1 y 247.

-Parity. Este parámetro tiene que ser igual con la paridad del maestro. Si es igual a 0 - sin paridad, si es igual 1 – paridad impar y si es igual 2 – paridad par.

-Delay. Este parámetro retarda el timeout de fin de mensaje. En redes cableadas suele ser 0 por lo que nosotros pondremos 0.

-MaxIQ. Este parámetro ajusta el numero de entradas y salidas digitales disponibles para direcciones modbus a valores comprendidos entre 0 y 128. El 0 inhibe la lectura y escritura de las entradas y salidas y con el 128 puedes acceder a todas las entradas y salidas.

-MaxAI. Este parámetro ajusta el número de entradas analógicas disponibles para la dirección modbus. En nuestro caso al tener una CPU 221 lo ajustaremos a 0

-MaxHold. Ajusta el número de registros de retención de memoria V disponibles para la dirección modbus.

-HoldStart. Es la dirección inicial de los registros de retención en la memoria V. En nuestro caso será la V0.

-Done. Se activa una vez finalizada la operación MBUS\_INIT. En este parámetro tenemos que tener cuidado ya que no se podrá usar la salida Q0.0 ya que nuestro autómata está utilizando esta salida.

- Error. Contiene el resultado de ejecución de la operación

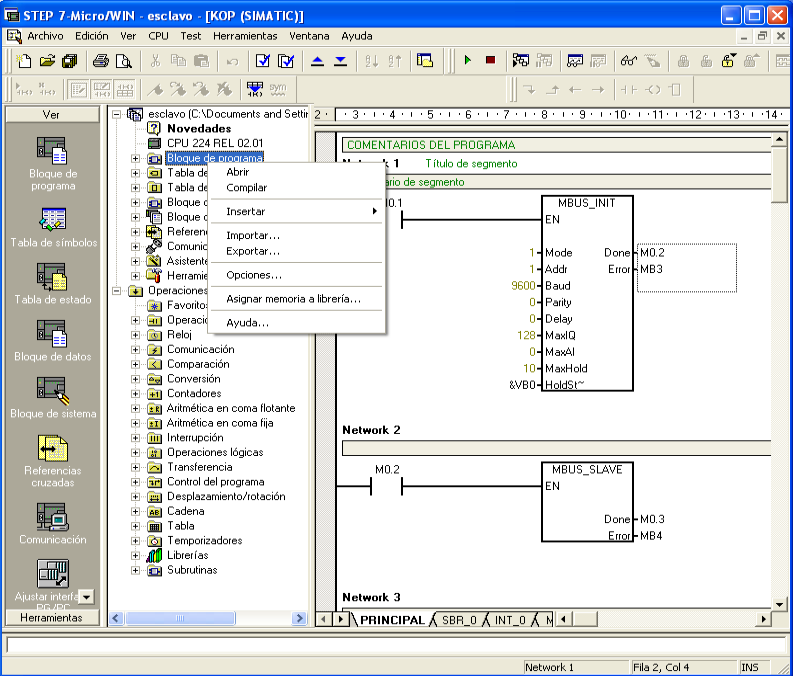
Función MBUS\_SLAVE:

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

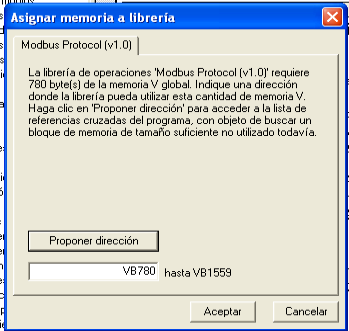
-Done. Este parámetro se activa cuando la operación MBUS\_SLAVE responde a una petición del maestro y se desactiva si no se ha procesado ninguna petición.

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

**TERCER PASO.** Lo siguiente que necesitamos hacer es reservar parte de la memoria para las funciones modbus. Para ello vamos a bloque de programa, botón derecho -> asignar memoria a librería.



Ahora tenemos le damos a proponer dirección para que busque un bloque de memoria que no se esté utilizando todavía.



A partir de la asignación de la memoria ya hemos terminado con el microWIN pero antes de cerrarlo para que no haya problema con la comunicación le cargamos el programa al autómata. Recordar que si queremos comunicar con el maestro no podemos tener abierto el microWIN.

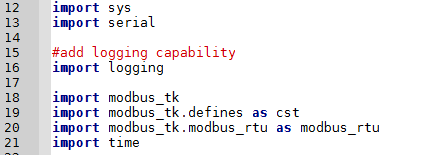
**CUARTO PASO**. Para correr el maestro en nuestro ordenador abriremos el archivo rtumaster\_example. Este archivo es un programa con el lenguaje de programación python en el cual podremos simular como si tuviéramos un maestro de modbus RTU y así poder comunicarnos con uno o más autómatas.

Lo primero que vamos a hacer es la instalación de la librería serial para ello lo único que hay que hacer es ejecutar el archivo y dale a siguiente hasta que ponga finalizar y ya estará instalada esta librería.

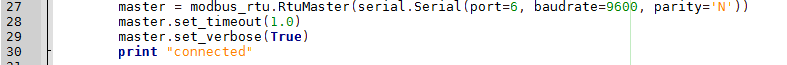
Para la otra librería que utilizaremos modbus\_tk lo único que tenemos que tener en cuenta es que la ruta del archivo rtumaster\_example sea la misma que la carpeta modbus\_tk.

**QUINTO PASO.** Bueno pues cuando tengamos ya instalada las librerías correctamente y el programa este abierto vamos a programar el maestro.

El primer paso es importar las librerías que necesitamos que son la librería sys, serial, logging, time y modbus\_tk.



El segundo paso es configurar la comunicación con el esclavo:

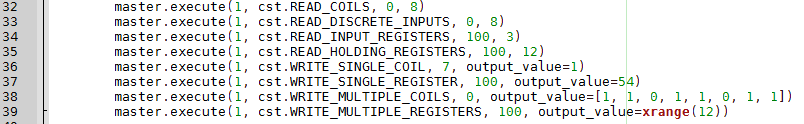


En la variable master se guardara la configuración de la comunicación modbus rtu por puerto serie con las funciones indicadas. En la función serial.Serial configuraremos el puerto en este caso igualaremos la variable port a 6 pero el puerto que utilizaremos en realidad será el 7 ya que a esta variable le tendremos que introducir un número menor al puerto físico utilizado, la variable baudrate la igualaremos 9600 ya que la velocidad configurada en el programa de microWIN es de 9600 bit/s y la variable Parity la igualaremos a ‘N’ esto es la paridad que el programa de siemens la hemos puesto a 0, esta variable tiene que coincidir con lo programado en el microWIN.

La función master.set\_timeout(1.0) esta función hace que si el programa no recibe respuesta en un segundo salta el error.

La función master.set\_verbose (True) esta función hace que el programa muestre mas información en la consola y por ultimo cuando ya hayamos comunicado imprimiremos connected en la consola.

El tercer paso es como el maestro pide o da información al esclavo con las siguientes instrucciones.



Como se puede observar en todas las funciones utilizaremos la función master.execute, la variable master es la que hemos configurado en el paso anterior. El primer parámetro de todas las funciones coinciden, es la dirección que tiene el esclavo en nuestro caso es de 1 ya que en el microWIN le pusimos el valor 1 a la variable Addr, este parámetro será importante para la comunicación entre dos o mas esclavo con cable 485 en el cual un esclavo tendrá la dirección 1 y otro esclavo la dirección 2, así sucesivamente.

En la primera función el segundo parámetro es cst.READ\_COILS con este parámetro lo que hacemos es indicar que queremos leer las salidas digitales, el tercer parámetro es la primer salida que queremos leer en este caso empezará por la salida Q0.0 ya que el direccionamiento de la salida Q0.0 es 00000 y el cuarto parámetro es cuantas salidas a partir de la salida inicial quieres leer en este caso 8 por lo que el esclavo nos enviara si están activadas de la salidas Q0.0 hasta la Q0.7.

En la segunda función el segundo parámetro es cst.READ\_DISCRETE\_INPUTS con este parámetro lo que hacemos es indicar que queremos leer las entradas digitales, el tercer parámetro lo que tenemos que introducir es el direccionamiento de una salida igual que anteriormente con las entradas y el cuarto parámetro es para cuantas entradas a partir de la inicial queremos leer.

En la tercera función el segundo parámetro es cst.READ\_INPUT\_REGISTERS con este parámetro lo que hacemos es indicar que queremos leer las entradas analógicas, el tercer parámetro introduciremos el direccionamiento de una entrada analógica y en el cuarto parámetro cuantas entradas analógicas a partir de la inicial queremos leer.

En la cuarta función el segundo parámetro es cst.READ\_HOLDING\_REGISTERS con este parámetro lo que hacemos es indicar que queremos leer registros de retención de memoria “V”, el tercer parámetro introduciremos el direccionamiento de uno de estos registros y en el cuarto parámetro cuantos registros a partir del inicial queremos leer.

En la quinta función el segundo parámetro es cst.WRITE\_SINGLE\_COIL con este parámetro lo que hacemos es indicar que queremos escribir una salida, el tercer parámetro es el direccionamiento de la salida que queremos escribir y el cuarto parámetro igualaremos output\_value a 1 si la queremos activar o a 0 si la queremos desactivar. Hay que tener en cuenta que si la salida se utiliza en el programa del esclavo, el maestro no podrá escribir la salida.

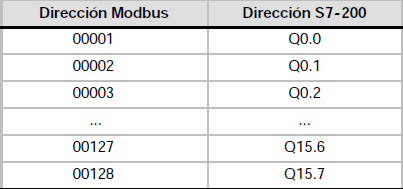
En la sexta función el segundo parámetro es cst.WRITE\_SINGLE\_REGISTER con este parámetro lo que hacemos es indicar que queremos escribir un registro de retención de memoria “V”, el tercer parámetro introduciremos el direccionamiento de uno de estos registros y en el cuarto parámetro igualaremos output\_value al valor que queramos introducir en la memoria.

En la séptima función el segundo parámetro es cst.WRITE\_SINGLE\_REGISTER con este parámetro lo que hacemos es indicar que queremos escribir mas de una salida, el tercer parámetro le introduciremos el direccionamiento de una salida y en el cuarto parámetro igualaremos output\_value lo igualaremos a una lista en la cual cada posición de la lista tendrá que tener los valores 0, si quiere que se desactive la salida, o 0, si quieres que se active la salida. La posición 0 de la lista será el valor de la salida que hemos introducido en el tercer parámetro, la posición 1 de la lista será el valor la salida que sigue a la introducida en el tercer parámetro y así sucesivamente.

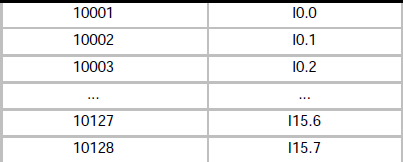
En la octava función el segundo parámetro es cst.WRITE\_MULTIPLE\_REGISTERS con este parámetro lo que hacemos es indicar que queremos escribir mas de un registro de retención de memoria “V”, el tercer parámetro introduciremos el direccionamiento de uno de estos registros y el cuarto parámetro igualaremos

DIRECCIONAMIENTO MODBUS.

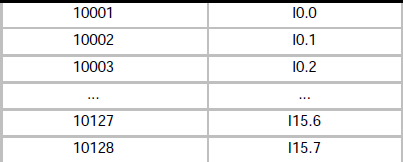
- De 00001 hasta 00128 son salidas digitales asignadas a Q0.0 hasta Q15.7



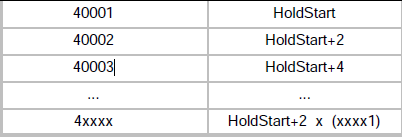
- De 10001 hasta 10128 son entradas digitales asignadas a I0.0 hasta I15.7.



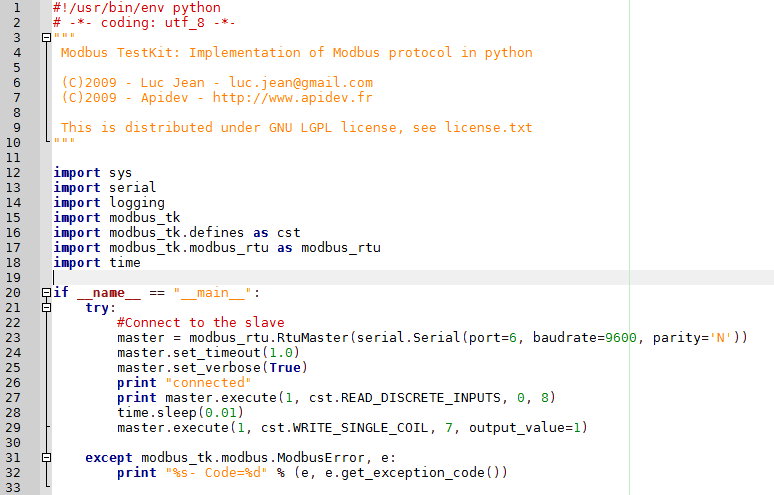
-De 30001 hasta 30032 son registros de entradas analógicas asignados a AIW0 hasta AIW62.



-De 40001 hasta 4xxxx son registros de retención asignados a la memoria V.



**SEXTO PASO.** Vamos a hacer un pequeño programa en el cual comuniquemos un esclavo s7-200 con el programa que hicimos anteriormente en microWIN y con un maestro el cual leerá las ochos primeras entradas del autómata y activará la salida Q0.7:

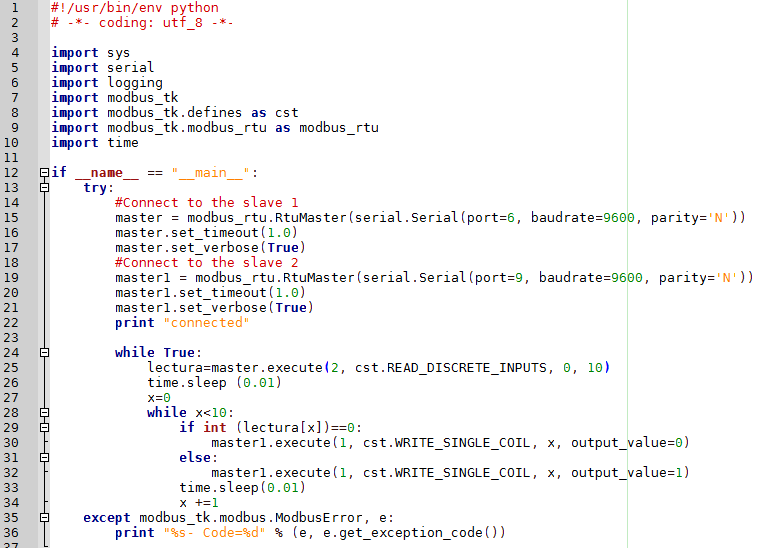


Como podemos observar lo único que haremos es utilizar las dos funciones que antes hemos explicado, la función cst.READ\_DISCRETE\_INPUTS que imprimirá una lista con el estado de las entradas Q0.0 hasta Q0.7.

Tenemos que tener cuidado al utilizar dos peticiones seguidas al esclavo ya que la velocidad del esclavo es menor a la velocidad del maestro por lo cual lo que haremos es parar el programa con la función time.sleep() por un tiempo de una centésima de segundo entre medio de las dos comunicaciones.

Después de la parada lo único que haremos es activar la salida Q0.7 por medio de la función cst.WRITE\_SINGLE\_COIL. Para terminar el programa tendremos un print que nos imprimirá si hemos tenido un error.

Cuidado con activar alguna salida desde el maestro ya que no podremos activar las salidas que ya se estén utilizando en el programa de microWIN.

**SEPTIMO PASO.** Por último vamos hacer un programa en el cual vamos a comunicar dos esclavos s7-200 con el simulador de python de maestro. El programa tendrá la finalidad de activar la salida Q0.1 de el esclavo 2 si esta activada la salida I0.1 del esclavo 1 así con las primeras diez salidas. El programa que introduciremos en los dos s7-200 será el mismo programa que introdujimos anteriormente. El siguiente programa será el maestro: 

Igual que en el programa anterior las primeras 14 primeras líneas es para la importación de las librerías necesarias y el inicio del programa.

De la línea 15 a la 17 vamos a configurar el esclavo 1, el cual dicha configuración la guardaremos en la variable master, este esclavo será el encargado de la lectura de las entradas. La comunicación será a través del puerto 7.

De la línea 19 a la 21 vamos a configurar el esclavo 2, el cual la configuración de este esclavo la guardaremos en la variable master 1, este esclavo activará y desactivará las salidas que el maestro le indique. La comunicación será a través del puerto 10.

Tendremos while True: para que el programa se quede dentro del bucle y pueda estar leyendo y escribiendo constantemente. Lo primero que haremos en le bucle es igualar la variable “lectura” a la lectura de las diez primeras entradas del esclavo 1. El cual tendrá un resultado de una lista con 10 posiciones con valores de 0 si están desactivadas las entradas y de 1 si están activadas.

Después de la lectura el programa sufrirá una parada de 0.01 segundos por lo explicado en el programa anterior e igualaremos la variable x a 0 para empezar un nuevo bucle.

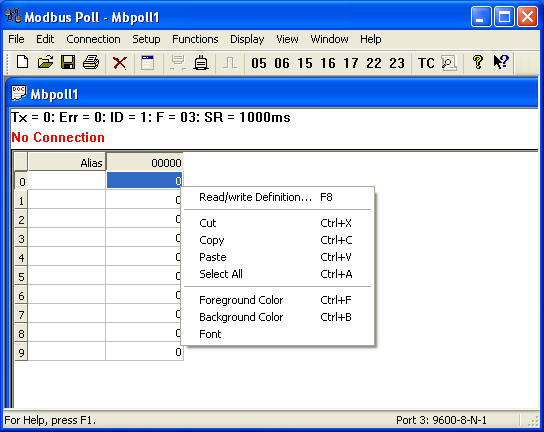
El programa se quedará dentro del bucle siempre que la variable x sea menor que 10. Dentro del bucle se irá comprobando todas las posiciones de la lista de la variable lectura. Si la posición de la lista es igual 0 el bucle entrará en el if y desactivará la salida sino entrara en el else y la activará, así con las diez posiciones. Entre la comprobación de cada posición se parara el programa durante 0,01 segundo.

Cuando haya comprobado las 10 posiciones el ciclo comenzará de nuevo y así podremos ir leyendo las entradas del esclavo 1 y escribiendo las salidas del esclavo 2 constantemente.

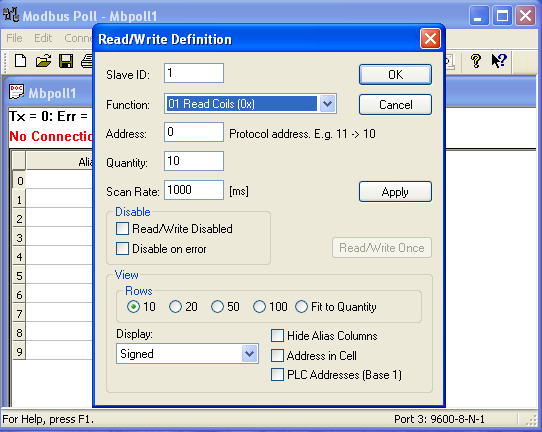
***ANEXO:***

Si tenemos algún problema con la comunicación del autómata y el programa del maestro podremos comprobar a través del siguiente programa si el autómata es el causante del problema para ello vamos a utilizar el programa Modbus Poll. Para ello vamos a seguir los siguientes pasos:

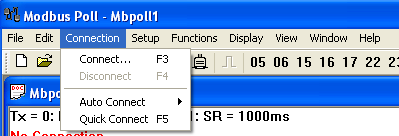
**PRIMER PASO.** Ejecutaremos dicho programa después nos aseguraremos que el cable RS232 este bien conectado en el PLC y el PC. A continuación pulsaremos con el botón derecho del ratón y entramos en ‘’Read/Write Definition’’.



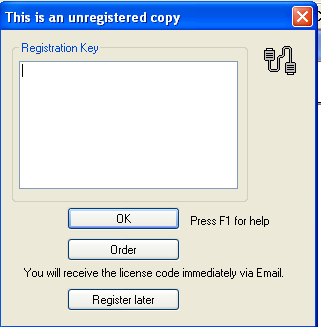
**SEGUNDO PASO.** En el segundo paso vamos a especificar la tarea que queremos hacer. Para ello donde pone function podemos observar las distintas tareas que podemos hacer que son las mismas que podemos hacer desde el programa del maestro. En nuestro caso vamos a leer las salidas así que le daremos a “Read Coils (0x)” los demás parámetros no los cambiaremos. Por último le damos a ok.

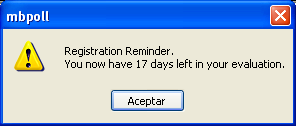


**TERCER PASO.** En este paso vamos a definir la conexión. Para ello primero le vamos a dar en el menú de arriba a connection y después a connect.

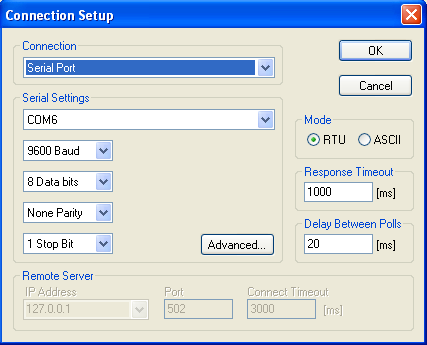


A continuación nos saldrá una ventana que tenemos que cerrar. Ya que esta ventana sale porque esta versión del programa es de prueba. Después nos saldrá una ventana en la cual nos pondrá los días que nos queda de utilización del programa y le damos a aceptar.

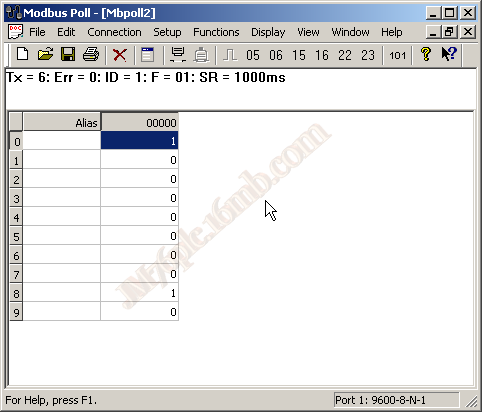




**CUARTO PASO.** Siguiendo con la definición de la conexión en la nueva ventana que se abrirá en apartado de connection introduciremos Serial Port, en el parámetro Serial Setting se introduciré el puerto serie que estemos utilizando en nuestro caso el 7, la velocidad de comunicación de 9600 Baud, la paridad que en nuestro caso es None Parity. En el parámetro Mode seleccionaremos RTU y los demás parámetros no los cambiaremos. A continuación pulsemos en el OK.



**QUINTO PASO.** Después le daremos al botón de conectar  y si el autómata no es el que está causando el problema en la conexión nos saldrá en la pantalla las salidas que están activadas en el esclavo poniendo un 1 en el cuadro del direccionamiento o si están desactivadas poniendo un 0. Como podemos observar en la siguiente imagen la salida Q0.0 y Q0.8 están activadas y las demás salidas están desactivadas.



**SEXTO PASO**. En este pasa vamos a escribir una salida del esclavo para ello damos doble clic sobre una de las salidas donde pone 0 o 1, en nuestro caso en la 5 y nos saldrá una ventana. En el parámetro Value seleccionamos On si queremos activarla u Off si queremos desactivarla y le damos a Send. En nuestro caso activaremos la salida Q0.5.

