# eRxP Modbus communication example for 3 x MAC400 motors.

**Introduction.**

In firmware v2.16 Beta for the MAC400 through MAC4500, a command was added to perform serial Modbus communications between one master drive/motor and up to 254 slave drives/motors.

The number may be reduced so it's maximum 32 motors due to the RS485 hardware limitation.

The instructions can each transfer one register between the local motor and the remote motor.

They support 16-bit and 32-bit values. At this time, values are not scaled with the on-the-fly scaling system.

The intended use is that the Modbus master motor works as an inexpensive PLC, and use inexpensive MAC00-B4 and MAC00-B41 modules on all motors. Note this also gives the master motor access to remote inputs and outputs.

IMPORTANT: More info can be found in the "eRxP Modbus communication" document.

## Example:

**Configuration of Master and Slave.**

In the example the Master has got the address no. 1 and the Slave no. 2 which are set in MacTalk at the main page.

Next step is to setup the Modbus communication for the master and slave.

Each remote motor must also use its register 213, UART1_SETUP, with bits [25:24] set to 0.
The baudrate is selected by the lowest four bits [3:0] and must be the same on the master and the slave motor(s). This can be set using MacRegIO.

Master for 1 Mbit: 0000 0011 0001 0000 0000 1011 0001 0111 = 0x3100B17 = 51383063
Slave for 1 Mbit: 0000 0000 0000 0000 0000 1011 0001 0111 = 0x0000B17 = 2839

Bit 0-3 =7 - 1Mbit
Bit 4-7 =1 - Modbus ( motor address )
Bit 8-9 =3 - 8 Data bits
Bit 10 =0 - Not used
Bit 11-13 = 1 - Odd parity
Bit 14-15 = 0 - 1 stop bit
Bit 16-19 =0 - Not used
Bit 20 =0 - Multi-drop
Bit 21 =0 - Half duplex
Bit 22-23 =0 - Not used
Bit 24-25 =3 - Master and slave 0
Bit 26-27 = 0 - Reserved
Bit 28-31 =0 - Timeout ms.

**Usage**

**0x0E**: Special command to perform **Communications.**

**0x01 or 0x02**: Operation type. The following operations are supported:

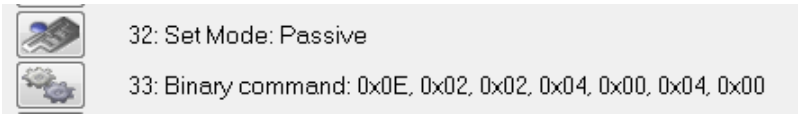
0x01 selects **Rx32** - reading of one remote 32-bit register to a local register.
0x02 selects **Tx32** - send one local 32-bit register to one remote register.

**0x02: Address of the remote motor** in this case address 2. This is the motor address that be configured on the Main tab in MacTalk.

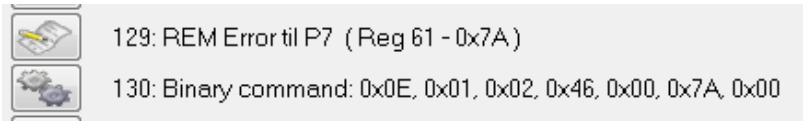
**0x04, 0x00: The 16-bit register number of the remote register.** Please note this uses the 16-bit-register numbering to be compatible with Modbus, so normal motor register numbers must be multiplied by two. In this case the value 0x0006 points to register 2, Mode (2*2 = 4 = 0x04).

**0x04, 0x00: The 16-bit register number of the local register.** Please note this uses the 16-bit-register numbering to be compatible with Modbus, so normal motor register numbers must be multiplied by two. In this case the value 0x0004 (= decimal 04) points to register 2, P_IST (2*2 = 4 = 0x04).

This first command ( line 32 ) set the master to Passive mode.
Line 36 copies this register value into the slave mode register.




This command reads the slaves error register ( Reg. 35 ) and copy it into the masters position register 7.




**0x46, 0x00: The 16-bit register number of the remote register.** Please note this uses the 16-bit-register numbering to be compatible with Modbus, so normal motor register numbers must be multiplied by two. In this case the value 0x0046 points to register 35, Error (2*35 = 70 = 0x46).

**0x7A, 0x00: The 16-bit register number of the local register.** Please note this uses the 16-bit-register numbering to be compatible with Modbus, so normal motor register numbers must be multiplied by two. In this case the value 0x007A points to register 61, Position 7 (2*61 = 122 = 0x7A).

**System with 3 x MAC400 and B41 running syncronized.**

To be able to run synchronized the motors need to be in the same position at power up.

**Slave.**

The program in the Slaves are just checking if the communication to the Master is ok.

When the Master communicates with the Slaves the Timer 4 is set to the default value saved in the beginning of the program. In this version it's 5 ms. So if the time until next communication with the Master is more than 5 ms it goes into passive mode. At 3000 rpm this is 90 degrees of the motor shaft. It might be possible to decrease this value to 3 or maybe 2 ms but this is not checked at the moment.

When the communication is up running again the register is updated again and it continues to run and the mode and other parameters can be changed from the Master.

**Master.**

The program in the master is the one that control the Slaves. It starts up in passive mode.

In the beginning it resets the errors in the motors and then synchronize the position to a certain value ( Normally it's 0 ) which can be changed in the Master so they start at the same position.

Next step is to set up the Acceleration, Velocity and mode which is done in the main loop in the Master. They are keyed-in into **"Acceleration 4" [Rpm/s], "Velocity 5" [Rpm], "Velocity 2" [Cts/Smp].**
Also the position can be keyed-in into **"Position 5"** register, but this doesn't make the motors run.

When input 1 ( using output 1 ) on the Master is activated the main loop is activated and the "Position 5" ( reg. 57 ) value is copied into "Requested position" ( Reg. 3 ) and also to the slaves "Requested position" register and the 3 motors start to run towards this position and when reaching this point in the sample loop.
Normally this sample loop is run every 1.3ms but the 3 motors are not synchronized. That means that there might be a timing difference between the 3 axis of 1.3ms and sometimes even more.
At 3000 rpm 1.3 ms is equal to 533 counts.
The main loop in the Master takes about 1-2ms at 1 Mbit Modbus communication.

The Master is then reading the "Actual position" of the slave and check if the position error between the Master and the Slave 2 is too big. It compares the difference with "Velocity 3" ( > ) and "Velocity 4" ( < ).
If it's out of range it make both Master and Slave 2 stop and send "999" to "Position reg. 8".
At the moment the limits are set to +/-4096 counts. I have tested with 3000 counts but this error came up when changing direction several times.
This distance is not the real distance between the Master and the Slave 2 because there's a time difference between the slave position transferred to the master through Modbus and the master position read just after.

If it's within the range and continues and also check if the error register in the Slave 2 a showing "any error" ( Reg 35. Bit 24 ). If there is an error in the Slave 2 all motors go into passive mode and "Position reg. 8" shows "777".

In case there is an error in the Master the Slave 2 is also told to go into passive mode and "Position reg. 8" shows "888".

If any of these errors have been activated you need to restart the Master program after checking for any errors.

Otherwise the program continues to run in a loop where "Velocity" and "Position" can be changed all the time – also during running. If acceleration or mode should be changed input 1 should go low and then high again.

**Test with 2 Mbit:**

I made a test with 2 Mbit too. The motor was running fine, and the slave position values looked much better, but the main loop time changed between 1 and 2 ms and there was a lot of communication errors.

A better cable for communication would probably make it better.