Communication Protocol

REF: CP038-CollinelO-DOC03-ComProtocol

21 de November de 2018

Version: V1.0

Authors: JLC, MMC

Approved: JLC

Edited:

Cerro Electronic Design

Avda. Cerro del Águila Nº9

28703 San Sebastián de los Reyes

Madrid, España.

Distribution List

Table 1: Distribution List

Company	Name	Position	No. Of copies

Document Revision History

Table 2: Document Revision History

Version	Issue Date	Brief description of Change
V1.0		First Version of this document

ÍNDICE DE CONTENIDOS

1 INTRODUCTION 6

1.1 Acronyms 6
1.2 Document Property 7
1.3 References 7

2 RS485 PORT 8

2.1.1 Test using a USB-RS485 Converter 10

2.2 MODBUS RTU Communication Protocol 10
2.3 RS485 Bus Configuration 11
2.4 Debug Serial Port: UART TTL 11

3 SOFTWARE PROTOCOL 13

3.1 ModBus Object Types 13
3.2 Frame Format 13
3.3 Function Codes 14

3.3.1 Function 0x02 (Read Digital Input) 14

3.3.2 Function 0x04 (Analog Input Read), (Firmware version) and (PCB model) 15

3.3.2.1 Analog Input Channels 1-2-3-4 16
3.3.2.2 Channels 5-6-7 Temperature PT100 17
3.3.2.3 Channel 8: +/-1V Sensor 18
3.3.2.4 Channels 9-10: Counter channels values 18
3.3.2.5 Reading Firmware version 19
3.3.2.6 Reading PCB model 19

3.3.3 Function 0x06 (Analog Output Set-DAC value Set) 20

3.3.4 Function 0x0F (Write Digital Outputs) 20

3.3.5 Exception Codes 21

4 EXAMPLES 23

4.1 Example of function code 0x02: Digital Inputs 23
4.2 Example function code 0x04: Analog Input channel 24
4.3 Example function code 0x04: Temperature channel 24
4.4 Example function code 0x04: Conductivity channels 25
4.5 Example function code 0x04: pH channel 26
4.6 Example function code 0x04: Counter channel 27
4.7 Example function code 0x04: Firmware version 28
4.8 Example function code 0x04: PCB model 28
4.9 Example of function 0x06 29
4.10 Examples of function 0x0F: Write Digital Outputs 29

LIST OF FIGURES

Figure 1 : Master-Slave Block Diagram 8
Figure 2: RS485 MODBUS port 9
Figure 3: SW7 Microschitch for board ID and RS484 bus termination resistor 9
Figure 4 : USB-RS485 Converter 10
Figure 5: USB-RS485 converter with resistors for polarization and termination 10
Figure 6 : External CPU/MO40 Communication Diagram 11
Figure 7: Debug serial port pinout 12
Figure 8: ModBus Request/Response Protocol 13
Figure 9: Simplied schematic for voltage analog Input channels 17
Figure 10: Simplified schematic for current analog input 17
Figure 11: example for function code 0x02, DI 7 On 23
Figure 12: Example for function code 0x04, reading 5V on channel 5. 24
Figure 13: Example function code 0x04, temperature measurement 24
Figure 14: Example function code 0x04, Conductivity measurement, high range 25
Figure 15: Example function code 0x04, Conductivity measurement, medium range 25
Figure 16: Example function code 0x04, Conductivity measurement, low range 26
Figure 17: Example function code 0x04, pH measurement 26
Figure 18: pH measuring with a negative input voltage 27
Figure 19: Example function code 0x04, counter measurement 27
Figure 2220: Example function code 0x04, PCB model 28
Figure 21: Example of function 0x06, channel AO_2, output 5V. 29
Figure 22: 0x0F function code. All outputs set to On state 30
Figure 23: Example function code 0x0F, all outputs OFF state 31
Figure 24: Example function code 0x0F, OFF-ON-OFF-ON-OFF.... 32

LIST OF TABLES

Table 1: Distribution List 2
Table 2: Document Revision History 3
Table 3: RS485 Bus configuration 11
Table 4: Connector Header 3 Pins 11
Table 5: modbus Object types 13
Table 6: Master request format 14
Table 7: CollineIO answer message 14
Table 8. Function Codes 14
Table 9: Master for 0x02 14
Table 10 : CollineIO for 0x02 14
Table 11: Master for 0x04 15
Table 12 : CollineIO answer for 0x04 16
Table 13: Master for 0x04 16
Table 14: Analog Input Channels 16
Table 15: Master for 0x04 17
Table 16: Temperature Channels 17
Table 18: Master for 0x04 18
Table 19: Master for 0x04 18
Table 20: Counter channels 19
Table 21: Master for 0x04 19
Table 22: Master for 0x04 19
Table 23: Master for 0x06 20
Table 24: CollineIO answer for 0x06 20
Table 25: Master for 0x0F 21
Table 26: CollineIO answer for 0x0F 21
Table 27: CollineIO answer for 0x0F 21
Table 28: table of exception codes 22

1 INTRODUCTION

The present document describes the communication protocol between a Master system and the CollinelO boards.

CollinelO is able to manage digital inputs, digital outputs, analog inputs and analog outputs. In this document we will describe every command that Master system can send to CollinelO (slaves) to read inputs or configure outputs.

1.1 Acronyms

AC, Alternating current

ADC, Analog Digital Converter

ASIC, Application Specific Integrated Circuit

BGA, Ball Grid Array

BOM, Bill Of materials

CAN, Controller Area Network

DAC, Digital Analog Converter

DC, Direct Current

DSP, Digital Signal Processor

EEPROM, Electrically Erasable PROM

FPGA, Field Programmable Gate Array

ICD, Interface Control Document

I2C, Inter-Integrated Circuit, Serial bus data communication protocol

I/O, input/output

NO, Normally Open

NC, Normally Close

NDA, Non-Disclosure Agreement

OC, Open Circuit

PCB, Printed Circuit Board

PSOC, Programmable System On-Chip

RAM, Random Access Memory

RTC, Real Time Clock

RTU, Remote Terminal Unit

SC, Short Circuit

SMD, Surface Mount Device

SMT, Surface Mount Technology

SPI, Serial Peripheral Interface

TBD, To Be defined

TBC, To Be Confirmed

1.2 Document Property

The information contained in this document belongs to the intellectual property of Cerro Electronic Design S.L. This information is strictly confidential and cannot be copied or distributed by third parties without the written permission of the company itself.

1.3 References

- Technical Specification, CP038_DOC1_UserGuide
- Schematic of CollinelO

2 COMMUNICATION PORTS

The CollineIO system (slave) is an I / O data acquisition system that connects to an external CPU (master) via RS485 serial port using the ModBus protocol.

Up to 7 CollineIO cards can be connected in the same bus. Microswitch SW8 on the board will define the ID of the slave (See Figure 3: SW7 Microschitch for board ID and RS484 bus termination resistor).

The Figure 1 : Master-Slave Block Diagram, shows the complete system concept.

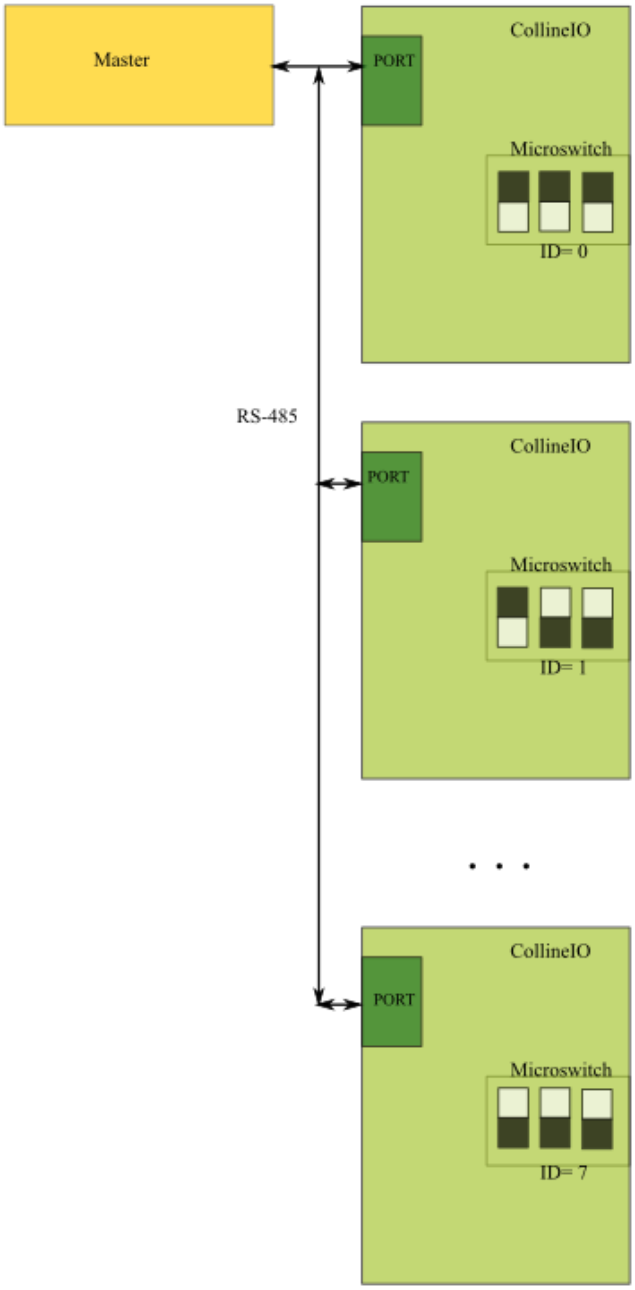

Figure 1 : Master-Slave Block Diagram

Modbus is a serial communication protocol developed and published by Modicon® in 1979 for use with its programmable logic controllers (PLCs). In simple terms, it is a method used for

transmitting information over serial lines between electronic devices. The device requesting the information is called the Modbus Master and the devices supplying information are Modbus Slaves. In a standard Modbus network, there is one Master and up to 247 Slaves, each with a unique Slave Address from 1 to 247. The Master can also write information to the Slaves.

The official Modbus specification can be found at www.modbus.org/specs.php .

Modbus is often used to connect a supervisory computer with a Remote Terminal Unit (RTU).

The interface communication is RS485 or RS232. (See user Guide and Schematic for the hardware configuration)

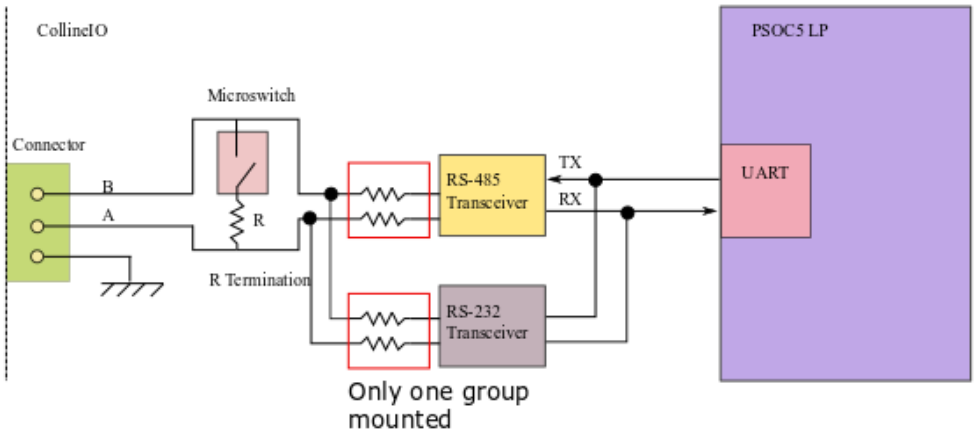

Figure 2: RS485 MODBUS port

The RS485 bus does include a terminator resistor which can be enabled or disabled using a microswitch. The bus should be terminated in the first and last boards of the chain of boards. Port does not include any polarization resistor. Next figure shows the switch to add the termination resistor:

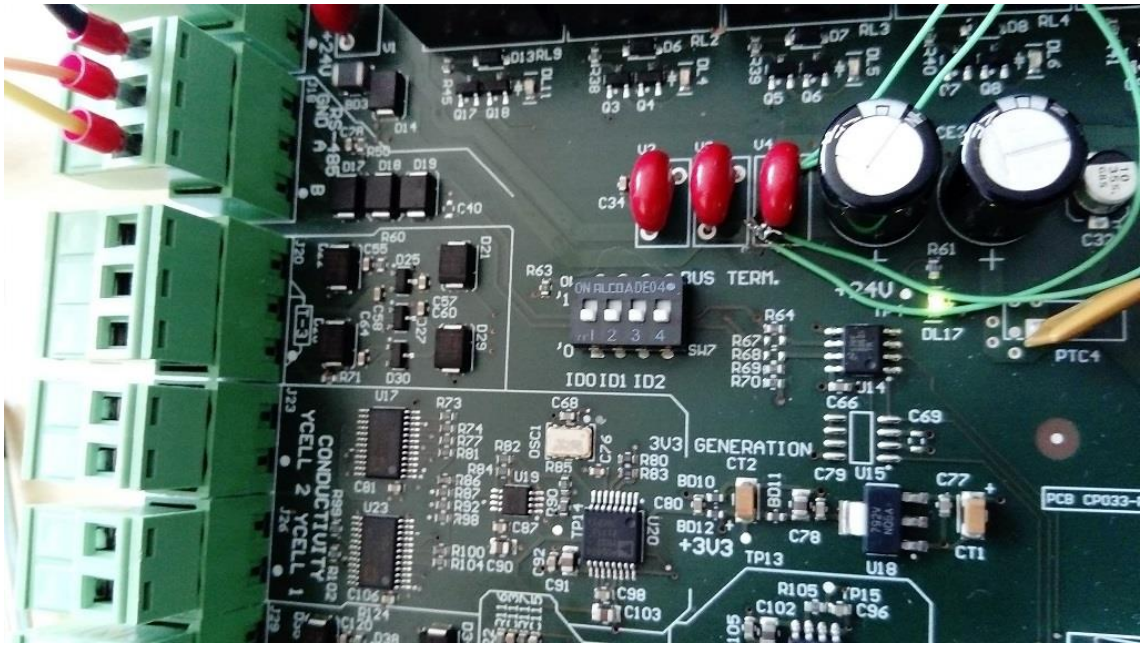

Figure 3: SW7 Microschitch for board ID and RS484 bus termination resistor

On SW7, the 4th position adds the bus termination resistor of 120 Ohms.

2.1.1 Test using a USB-RS485 Converter

Examples in this document has been realized using a USB-RS485 Converter, as for example, Farnell Cod.: 1740357, FTDI USB-RS485-WE-1800-BT.

Figure 4 : USB-RS485 Converter

The test has been done using polarization and terminator resistors.

Next figure shows the polarization and termination of the USB-RS485 converter. Polarization resistors are 220 Ohms and termination 120 Ohms.

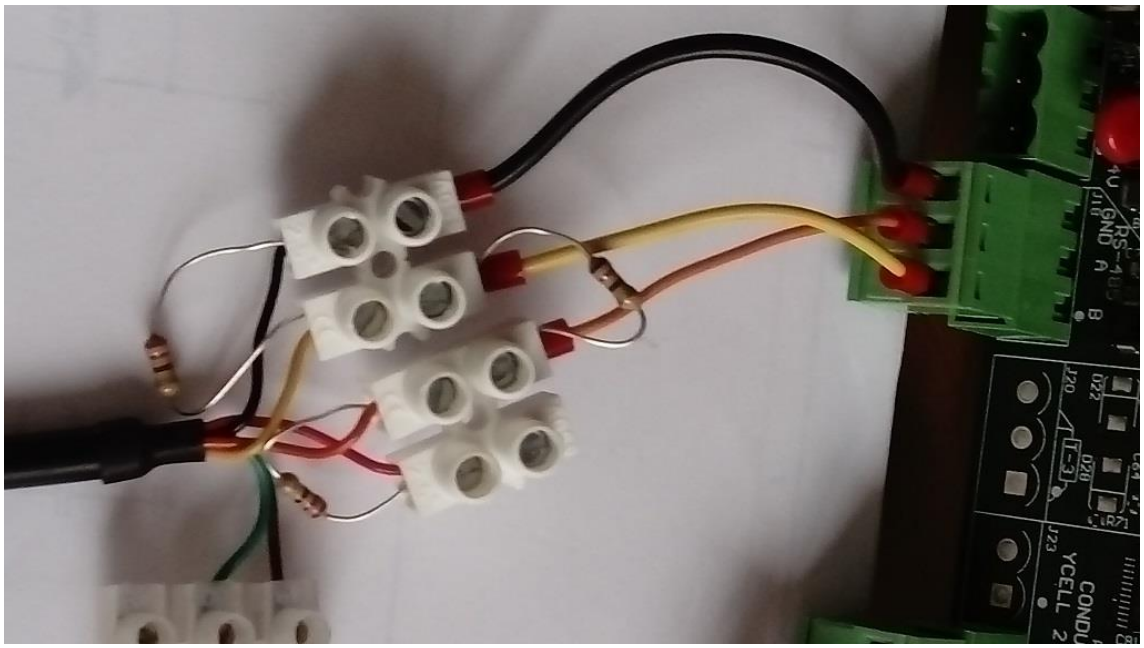

Figure 5: USB-RS485 converter with resistors for polarization and termination

2.2 MODBUS RTU Communication Protocol

The complete system works as a "Master-Slave" protocol, i.e., the master will initiate the communication by sending a command and the CollinelO will respond to the request.

It will be necessary to define a communication protocol between both devices and the Modbus protocol will be used as standard.

It should be noted that all intelligence of the system falls on the external CPU. The CollineIO card simply reads data packets from the external CPU, executes the commands contained in those data packet received and returns the requested values. That is, for example regarding an ADC measurement, CollineIO will return the ADC value measured in the input connector.

The following figure graphically shows the interface between the two subsystems.

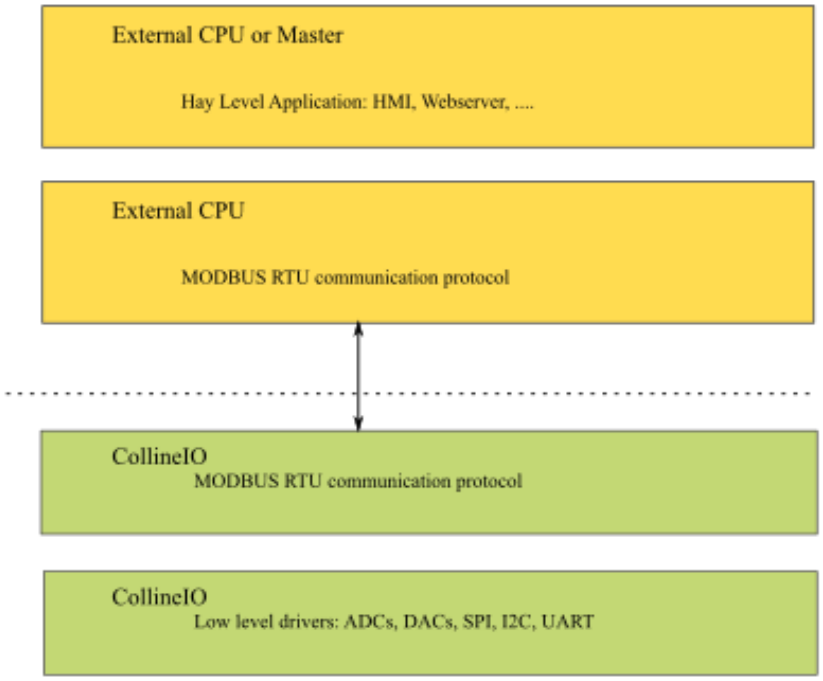

Figure 6 : External CPU/MO40 Communication Diagram

2.3 RS485/RS232 Bus Configuration

The RS485/RS232 is defined as half-Duplex with the following features:

Table 3: RS485 Bus configuration

Parameter	Value
Baud Rate	115200
Data Bits	8
Stop Bits	1
Parity	No
Flow Control	No

2.4 Debug Serial Port: UART TTL

There is serial debug port for debugging and developing purposes. Header Male Molex 22-27-2031 (or equivalent) with the following pinout:

Table 4: Connector Header 3 Pins

Pin #	CollineIO	Direction
1	TX	Output from PSOC
2	RX	Input to PSOC
3	GND	-

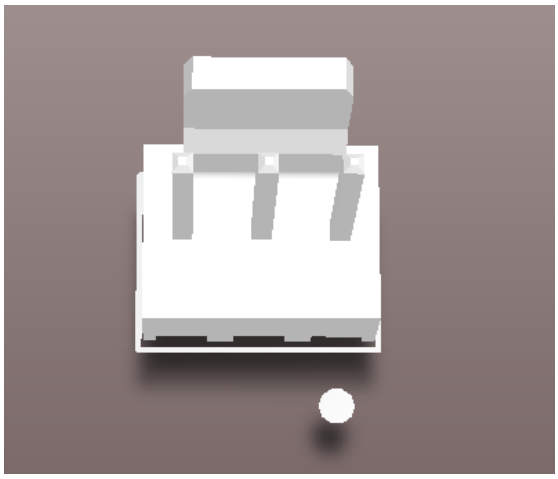

Figure 7: Debug serial port pinout

3 SOFTWARE PROTOCOL

In normal operation there will be an exchange of messages between Master and CollineIO. Both systems will send/receive different commands using the ModBus protocol.

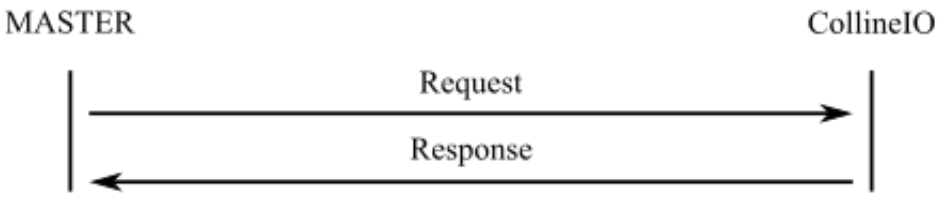

Figure 8: ModBus Request/Response Protocol

Only the master will initiate the communication.

3.1 ModBus Object Types

The following table shows the object types provided by a Modbus slave to a ModBus master:

Table 5: modbus Object types

Modbus Object Type	Access	CollineIO function
Coil	Read/write	Digital Output
Discrete Input	Read only	Digital Input
Input Register	Read only	Analog Input
Holding Register	Read/write	Analog Output

3.2 Frame Format

A Modbus frame is composed of an Application Data Unit (ADU), which encloses a Protocol Data Unit (PDU):

- ADU = Address + PDU + Error check,
- PDU = Function code + Data.

Modbus RTU frame format :(primarily used on 8-bit asynchronous lines like EIA-485)

Name	Length (bits)	Function
Start	28	At least 3 ½ character times of silence (mark condition)
Address	8	ID, Board address
Function	8	Indicates the function code; e.g., read coils/holding registers
Data	$n \times 8$	Data + length will be filled depending on the message type
CRC	16	Cyclic redundancy check (CRC)
End	28	At least 3 ½ character times of silence (mark condition)

Note about the CRC:

- Polynomial: $x^{16} + x^{15} + x^2 + 1$ (CRC-16-ANSI also known as CRC-16-IBM, normal hexadecimal algebraic polynomial being 8005 and reversed A001).
- Initial value: 65,535.
- Example of frame in hexadecimal: 01 04 02 FF FF B8 80 (CRC-16-ANSI calculation from 01 to FF gives 80B8, which is transmitted **least** significant byte **first**).

So, format for commands send/receive are:

Table 6: Master request format

Slave Address	Command Byte	Data Address	Variable value	CRC
1 byte	1 byte	2 bytes	Variable bytes	2 bytes

Table 7: CollinelO answer message

Slave Address	Command Byte	Variable value	CRC
1 byte	1 byte	Variable bytes	2 bytes

3.3 Function Codes

Following table summarizes the commands available:

Table 8. Function Codes

Command	Action	Signal Name	
0x02	Read	Read Digital Input	1-Bit
0x04	Read	Read Analog Input	16-bits
0x06	Write	Write Analog Output	16-Bits
0x0F	Write Multiple	Write digital Output	16-bit

3.3.1 Function 0x02 (Read Digital Input)

Read digital inputs of CollinelO.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x02	0x00 0x00	0x00 0x08	0xCRC

Table 9: Master for 0x02

Where:

Data Address: The data address of the first input to read, always 0.

Num data: Number of input to read is 8 (hardware defined), so we need at least 8 bits to read (0x08).

CollinelO answer:

Slave Address	Command Byte	Num Bytes	Byte Data	CRC
0xID	0x02	0x01	0xXX 0xXX	0xCRC

Table 10 : CollinelO for 0x02

Where:

Num. Bytes: number of bytes with data to follow, always 0x01, two bytes to read 16 outputs maximum.

Byte Data: Actual value of the outputs, where for example, for 0x01 0x02 means

MSB---LSB

0100 0010,

Input channel 1, DI1= 0ff

Input channel 2, DI2= On

Input channel 3, DI3= Off

…

Input channel 10, DI7= On

Input channel 11, DI8= Off

For example, a message exchange between master and slave, on board ID=0x01, with DI 5 On, would be:

Master: 0x01 02 00 00 00 10 79 C6

Slave: 0x01 02 02 00 40 B8 48

3.3.2 Function 0x04 (Analog Input Read), (Firmware version) and (PCB model)

Command used to read the actual value of analog input channel, +/-1V sensor, temperatures, pulse counters, reading the firmware version and reading de PCB model.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0xXY	0x00 0x02	0xCRC

Table 11: Master for 0x04

Where:

Data address: The Data Address of the first register requested, ranging from 0x00 0x01 (DEC 1) to 0x00 0x21 (DEC 33) in steps of 2. Given that we have 10 analog inputs.

Num Data: 0x00 0x02. We will always read 2 16-bit registers.

Channels 1-2-3-4, four analog input channels hardware configurable 0-20mA/0-10V, associated to AI_1 to AI_4 respectively.

Channels 8-9-10: PT100 temperature sensors,

channel 8 is T-1 on board silkscreen

channel 9 is T-2 on board silkscreen

channel 10 is T-3 on board silkscreen

Channel 13: +/-1V Sensor

Channel 14-17: two pulse counters channels

channel 9 is DC-1 on board silkscreen

channel 10 is DC-2 on board silkscreen

Num. Data: always 0x00 0x02, because we will read just one channel with 2 16-bit registers at a time.

CollinelO answer:

Slave Address	Command Byte	Num Bytes	Byte Data	CRC
0xID	0x04	0x04	0xXX 0xXX 0xXX 0xXX	0xCRC

Table 12 : CollinelO answer for 0x04

Where:

Num. Bytes: The number of data bytes to follow, always 0x04, four bytes, 32 bits; the value is four bytes. Internal ADC is 20 bits, so we need 32 bits.

Byte Data: Value of the analog input requested. MSB first.

3.3.2.1 Analog Input Channels 1-2-3-4

CollinelO-Base board has an internal **20 bits** Analog to digital converter with a range 0-2.048V. All values in the input will be referenced to that range.

The value returned is in mV and to know the real input value must be multiplied by 5 as explained below.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0xXY	0x00 0x02	0xCRC

Table 13: Master for 0x04

Data address: The Data Address can be **0x00 0x01 to 0x00 0x21**

Num Data: 0x00 0x02. We will always read 2 16-bit registers.

Next table shows correspondence between channel number and silkscreen on board

Table 14: Analog Input Channels

Silkscreen on Board	Channel number	Channel Data Address
AI_1	1	0x00 0x01
AI_2	2	0x00 0x03
AI_3	3	0x00 0x05
AI_4	4	0x00 0x07

The values stored in the registers are represented in mV and must be multiplied by 5 to find out the voltage applied to the inputs.

Next paragraphs explain the ADC **internal** functioning.

ADC converter code is:

$$Code = (2^{20} * VO) / 2.048V$$

Being VO the input voltage to the ADC.

For input voltage 0-10V, next figure shows the simplified schematic.

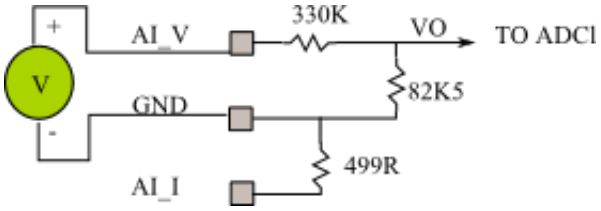

Figure 9: Simplied schematic for voltage analog Input channels

if AI_V= 0, Vo=0, so ADC gives us 20 bits: 0000 0000 0000 0000 0000

If AI_V= 10V,

10=I* (330K+82.5K); I= 0.024mA

Vo= I*82.5K=0.024*82.5=2 Volts,

so, Code= 1024000 (decimal), 1111 1011 0000 0000 0000 (binary), F A000 (hex)

For input voltage 0-5V, the input value is amplified internally by 2 so have the same range than before. So get the real value divide by two.

For input current 0-20mA, next figure shows the simplified schematic.

Figure 10: Simplified schematic for current analog input

So, if I=0 mA, VI=0V, VO=0V, ADC sees 0 Volts, gives us 20 bits: 0000 0000 0000 0000 0000

If I=20mA, VI=20mA*499=9.98V, and therefore the calculus is similar to the previous paragraph.

For example, an external board voltage input of 5V on AI_3 on board ID=0x01, channel 3, would be:

Master: 0x01 04 00 05 00 02 61 CA

Slave: 0x01 04 04 00 00 03 D7 BB 2A

3.3.2.2 Channels 8-9-10 Temperature PT100

The value of the temperature is 20 bits complement 2 values of the measured temperature. The returned values must be divided by 100 to know the temperature

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0xXY	0x00 0x02	0xCRC

Table 15: Master for 0x04

Data address n: The Data Address can be **0x00 0x0F**, **0x00 0x11 or 0x00 0x13**

Num Data: 0x00 0x02

Next table shows correspondence between channel number and silkscreen on board

Table 16: Temperature Channels

Silkscreen on Board	Channel number	Channel Data Address
T-1	8	0x00 0x0F

T-2	9	0x00 0x11
T-3	10	0x00 0x13

For example, a measured temperature of 22.41 ºC, on board ID=0x01, in channel 9, would be:

Master: 0x01 04 00 11 00 02 21 CE

Slave: 0x01 04 04 00 00 08 C1 3D D4

Note that if you measured a temperature on a channel without PT100 the value returned is too high. Next answer shows de concept:

Slave: 0x 01 04 04 **FF FF FA 82** 39 61

3.3.2.3 Channel 13: +/-1V Sensor

The value returned when measuring on +/-1V connector is a 20-bit value of the voltage differences between the two electrodes of the pH sensor. Its range is +-0.5V. As we can have negative values it is a two-complement value.

The value is given in mV.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0x15	0x00 0x02	0xCRC

Table 17: Master for 0x04

Data address n: The Data Address always 0x00 0x15

Num Data: 0x00 0x02, two registers

For example, a voltage input of 0.240V, on board ID=0x01, channel 13, would be:

Master: 0x01 04 00 15 00 02 60 0F

Slave: 0x01 04 04 00 00 00 F0 FB C0

A second example, for a voltage input of -0.180V, on board ID=0x01, channel 13, would be:

Master: 0x01 04 00 15 00 02 60 0F

Slave: 0x01 04 04 FF FF FF 4B FA 67

Note that real pH has to be compensated with a temperature.

3.3.2.4 Channels 14-15: Counter channels values

The value of the frequency is a 20-bit value of the pulse frequency in the input.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0xXY	0x00 0x02	0xCRC

Table 18: Master for 0x04

Data address: The Data Address can be **0x00 0x17**, **0x00 0x19**

Num Data: 0x00 0x02

Next table shows correspondence between channel number and silkscreen on board

Table 19: Counter channels

Silkscreen on Board	Channel number	Channel number hexadecimal
DC 1	14	0x00 0x17
DC 2	15	0x00 0x19

For example, a measured frequency of 200Hz, on board ID=0x01, in channel 15 (0x19), would be:

Master: 0x01 04 00 19 00 02 01 CC

Slave: 0x01 04 04 00 00 00 C7 BA 16

Note that if you measured a counter on a channel without any input the value returned is cero. Next answer shows de concept:

Slave: 0x01 04 04 00 00 00 00 FB 84

3.3.2.5 Reading Firmware version

The value of the firmware version is a 32 bits. Its structure is 4 bytes in ASCII value hexadecimal.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0xXY	0x00 0x02	0xCRC

Table 20: Master for 0x04

Data address: The Data Address **0x00 0x33**

Num Data: 0x00 0x02

Next table shows correspondence between channel number and silkscreen on board

For example, on board ID=0x01 and firmware version=01.4:

Master: 0x01 04 00 33 00 02 81 C4

Slave: 0x01 04 04 30 31 2E 34 B9 3C

The value returned in hexadecimal ASCII would be;

30 = '0', 31 = '1', 2E = '.', 34 = '4' -- (**01.4**)

3.3.2.6 Reading PCB model

This command is used for hardware configuration of the board, that is, for version of the board with does not have all components and son on.

The value of the PCB model is a 32 bits. Its structure is 4 bytes in numeric value hexadecimal.

Master message:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x04	0x00 0xXY	0x00 0x02	0xCRC

Table 21: Master for 0x04

Data address: The Data Address **0x00 0x35**

Num Data: 0x00 0x02

For example, on board ID=0x01 and PCB model =0:

Master: 0x01 04 00 35 00 02 81 C4

Slave: 0x01 04 04 00 00 00 00 FB 84

The possible return values are:

0 - 15

3.3.3 Function 0x06 (Analog Output Set-DAC value Set)

Write a new value in the analog outputs channel. Output are Voltage or current hardware configurable using two microsliders per channel.

Voltage output: 0-10V

Current output:0-20mA

Master message:

Slave Address	Command Byte	Data Address	Value Data	CRC
0xID	0x06	0x00 0x0n	0x00 0xXY	0xCRC

Table 22: Master for 0x06

Where:

Data address n: The data address of the analog channel, i.e., number of analog output to write, can be 1-4, because there are 4 DACs

Channel 1, AO_1 on board silkscreen

Channel 2, AO_2 on board silkscreen

Value Data: The value to write. Only one byte will be used because DACs are 8-bit. Valid values 0-255.

CollineIO answer:

Slave Address	Command Byte	Data Address	Byte Data	CRC
0xID	0x06	0x00 0x0n	0X00 0xXY	0xCRC

Table 23: CollineIO answer for 0x06

Note that the response is the same as master message.

For example, to set a voltage output of 5V (dec 127, hex 0x7F), board ID=0x01, analog output channel 2:

Master: 0x01 06 00 02 00 7F 38 2A

Slave: 0x01 06 00 02 00 7F 38 2A

3.3.4 Function 0x0F (Write Digital Outputs)

Write all digital outputs.

Master message:

Slave	Command	Data	Num Data	Num	Value Data	CRC

Address	Byte	Address		Bytes		
0xID	0x0F	0x00 0x00	0x00 0x10	0x01	0xXY	0xCRC

Table 24: Master for 0x0F

Where:

Data address: The data address of the first output, always 0x00 0x00, because we write all digital outputs from 0 channel.

Num. Data: The number of outputs to write, always 0x00 0x08, only the 8 existing outputs on hardware.

Num. Bytes: The number of data bytes to follow, always 0x01, because we will have 16 bits maximum to write

Value Data: actual value to be written. For example, we have 8 digital output channels and want to write first 4 channels (channels 1-4) On and 4 last one off (channels 5-8), so: 0x00 0x0F.

CollineIO answer:

Slave Address	Command Byte	Data Address	Num Data	CRC
0xID	0x0F	0x00 0x00	0x00 0x08	0xCRC

Table 25: CollineIO answer for 0x0F

Where:

Data address: The Data Address of the first output, always 0.

Num. Data: The number of outputs to write, always 0x00 0x08; 0x08, decimal 8 is the total number of digital outputs channels we have to write.

For example, a message exchange between master and slave, on board ID=0x01, to set all channels to On state, would be:

Master: 0x01 0F 00 00 00 10 02 00 FF A0 50

Slave: 0x01 0F 00 00 00 10 54 07

3.3.5 Exception Codes

For a normal response, slave repeats the function code, but when a Slave report and Error it will reply the requested function code plus 128 (hex 0x80), so for example, an error on function code 0x03 will be reported with 131, that is, hex 0x83.

Message will include only one byte of data with the exception code:

CollineIO answer for error code:

Slave Address	Command Byte	Exception Code	CRC
0xID	Function +0x80	0xXX	0xCRC

Table 26: CollineIO answer for 0x0F

Where Exception Code is one of the following:

Exception Code	Description
0x01	Function code received in the query is not recognized or allowed by slave
0x02	Data address of some or all the required entities are not allowed or do not exist in slave
0x03	Value is not accepted by slave

Table 27: table of exception codes

4 EXAMPLES

To be filled with real examples of all functions codes commands.

4.1 Example of function code 0x02: Digital Inputs

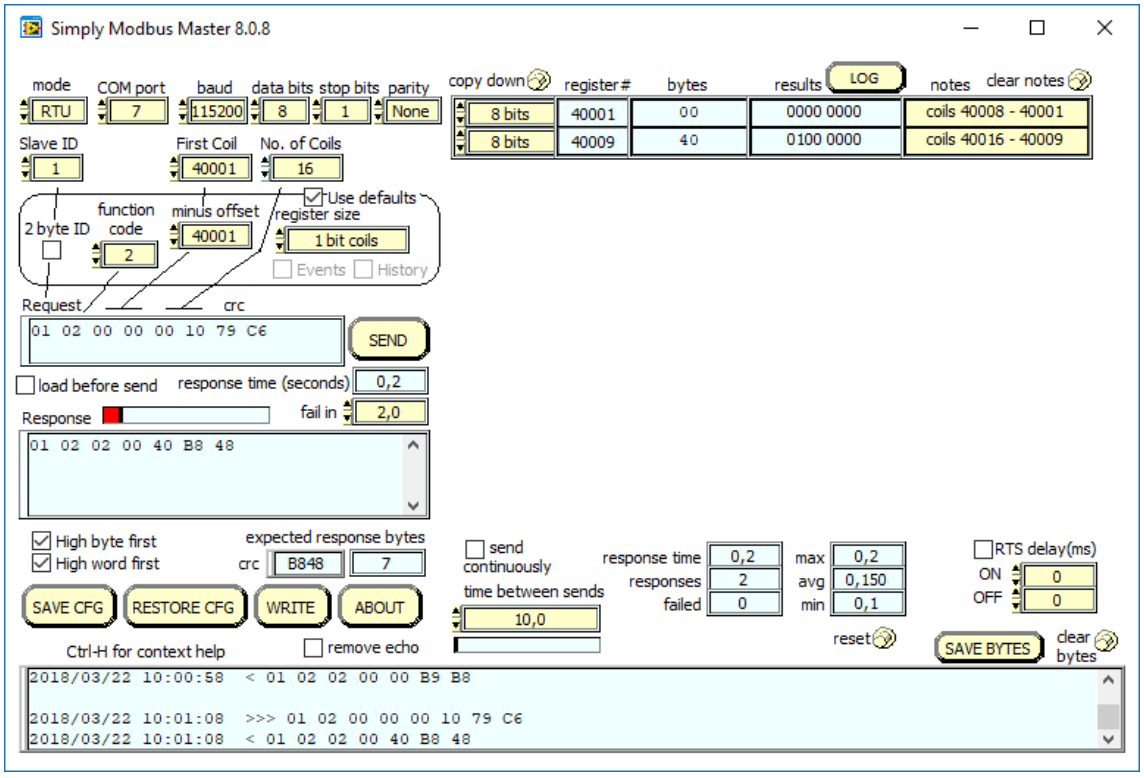

Figure 11: example for function code 0x02, DI 7 On.

In the previous figure only, input number 7 is active, the rest are on the off state (0, low level).

4.2 Example function code 0x04: Analog Input channel

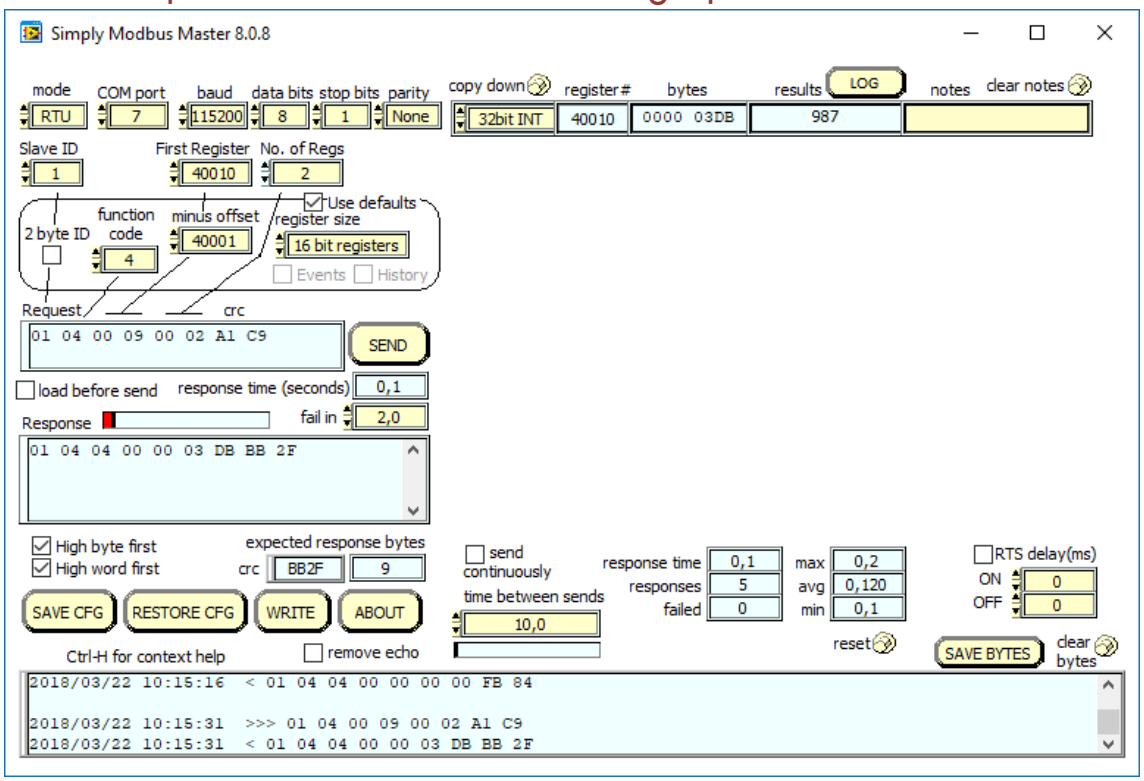

Figure 12: Example for function code 0x04, reading 5V on channel 5.

4.3 Example function code 0x04: Temperature channel

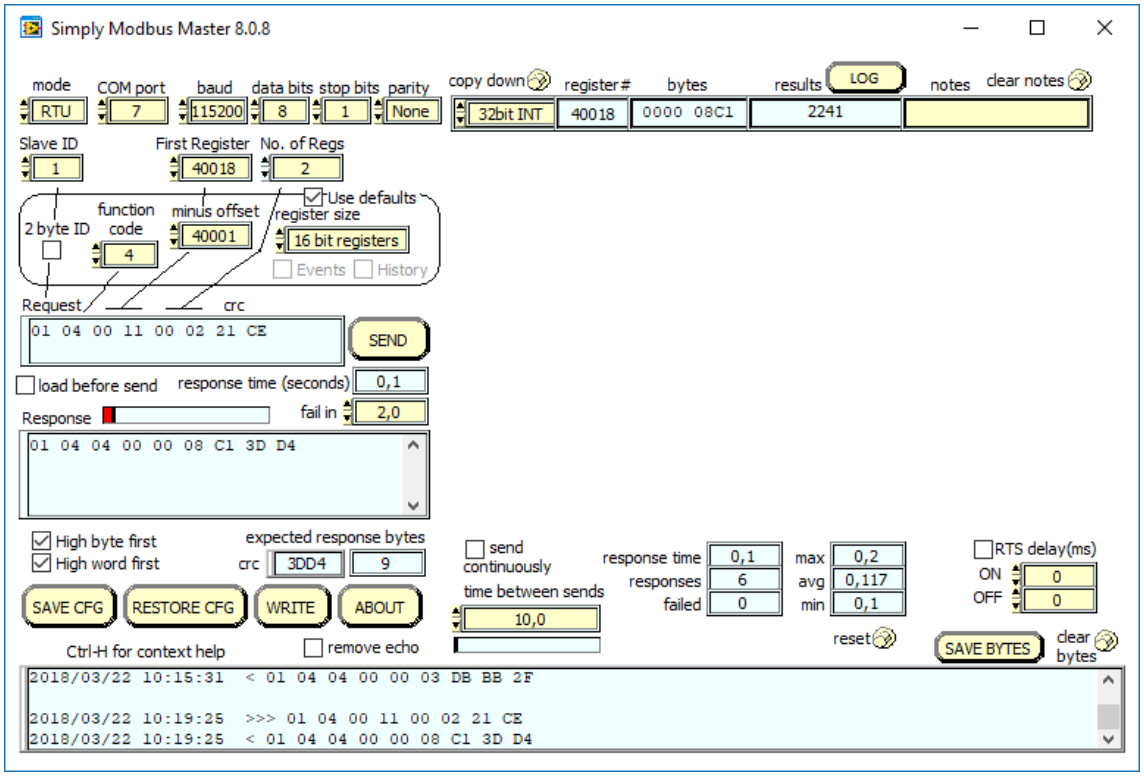

Figure 13: Example function code 0x04, temperature measurement

4.4 Example function code 0x04: Conductivity channels

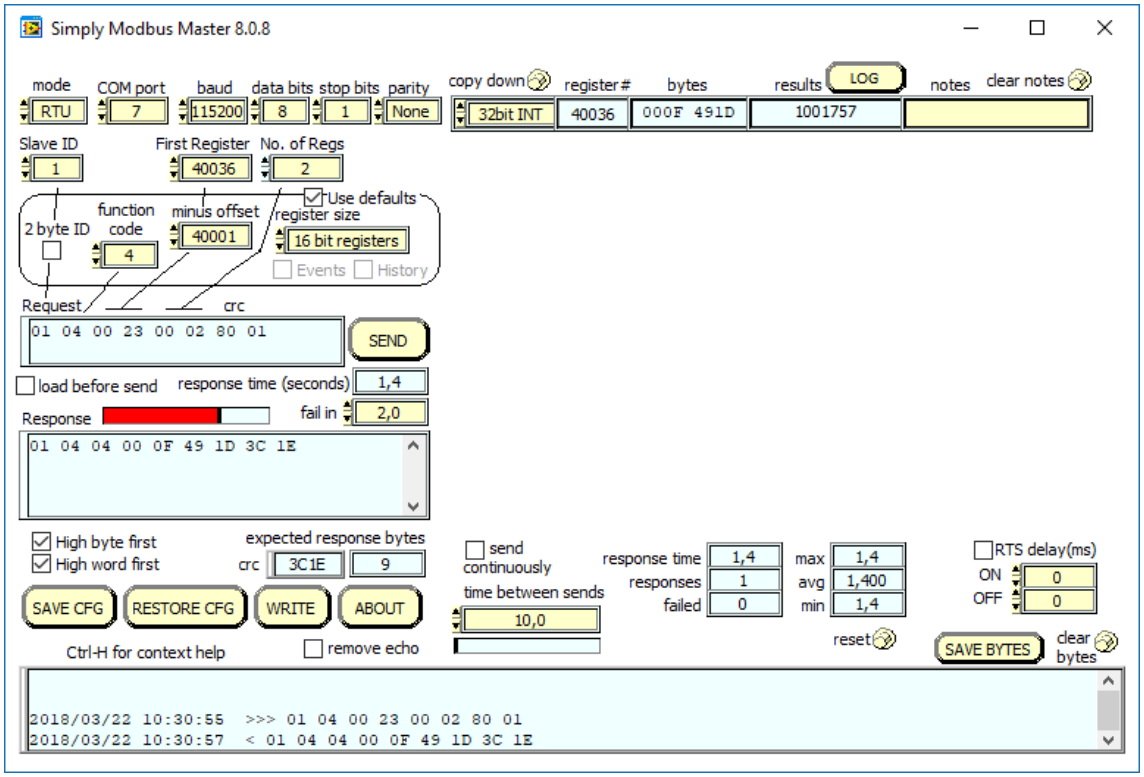

Figure 14: Example function code 0x04, Conductivity measurement, high range

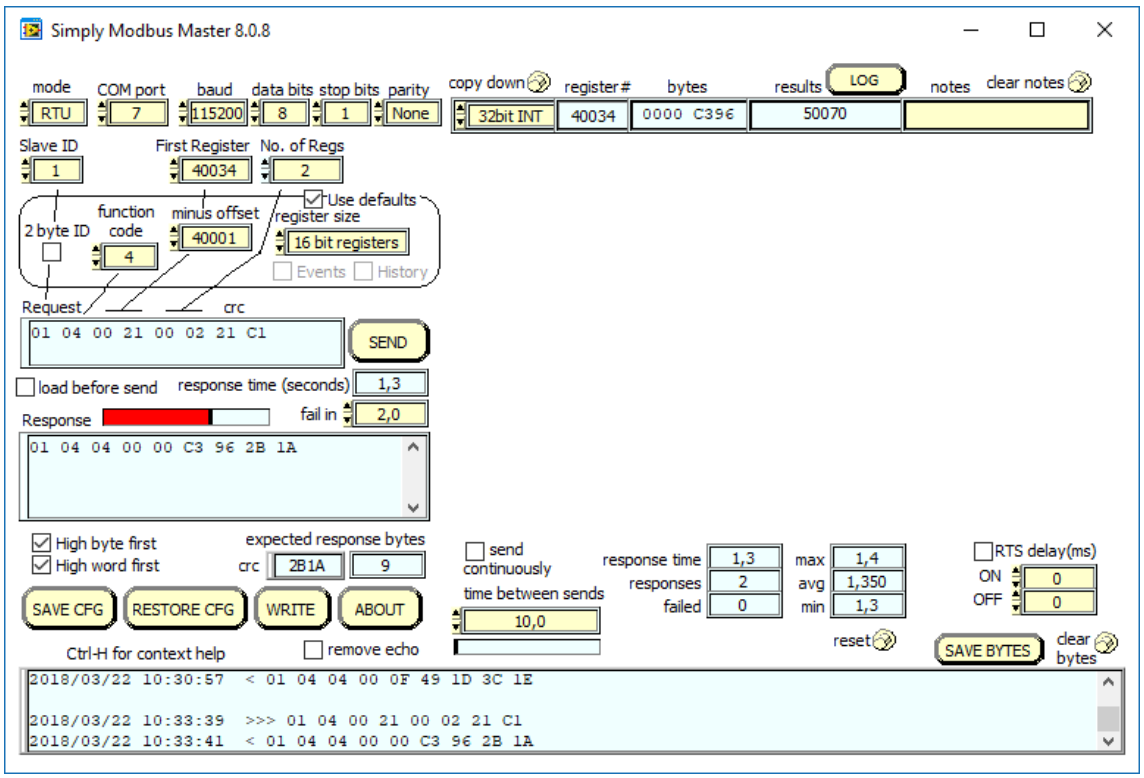

Figure 15: Example function code 0x04, Conductivity measurement, medium range

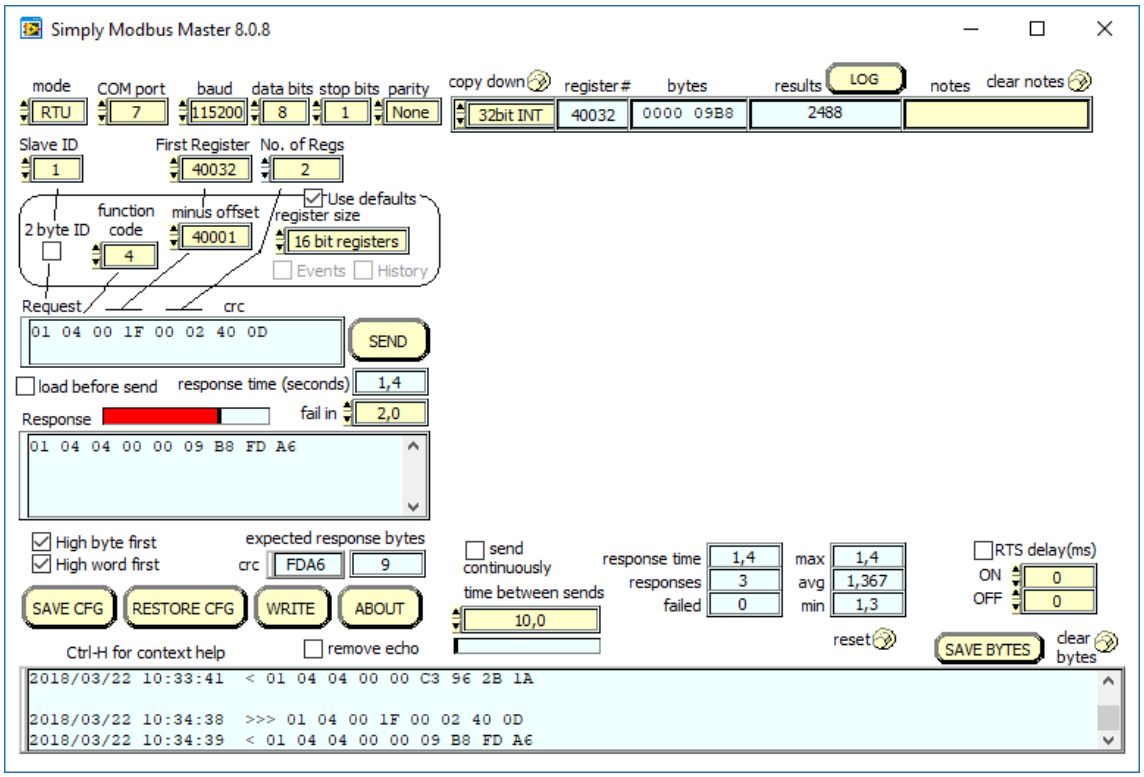

Figure 16: Example function code 0x04, Conductivity measurement, low range

4.5 Example function code 0x04: pH channel

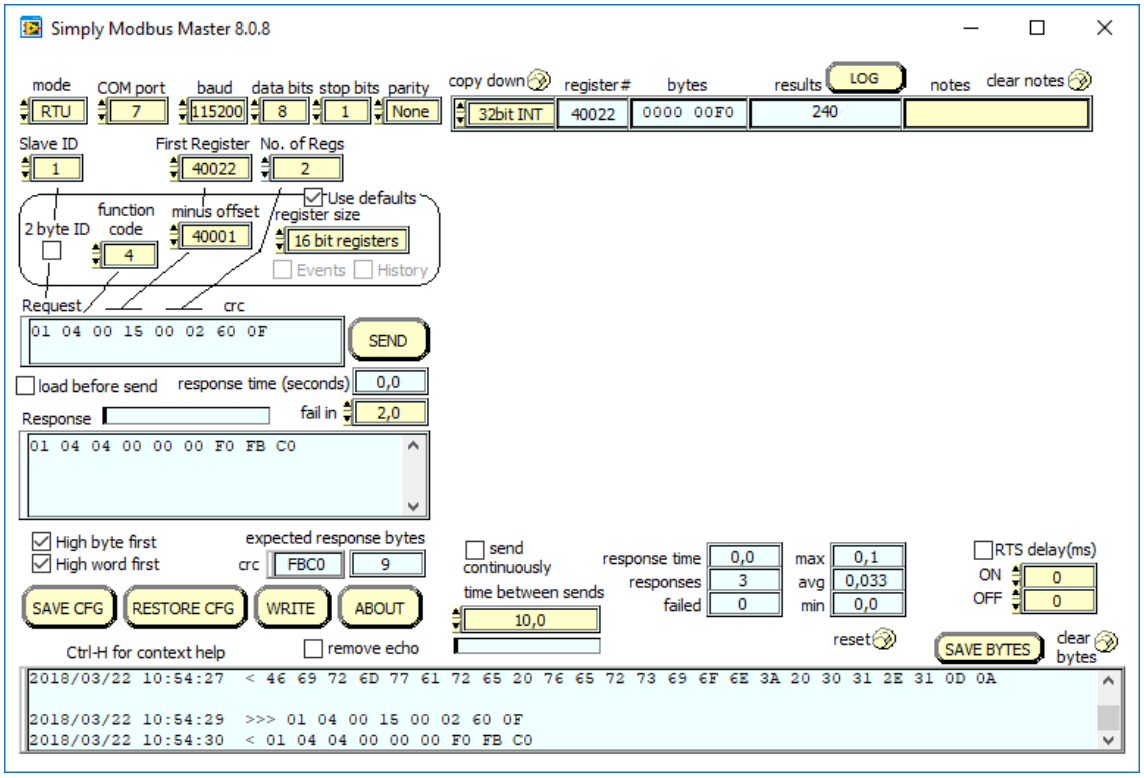

Figure 17: Example function code 0x04, pH measurement

This is another example but measuring negative voltage at the input

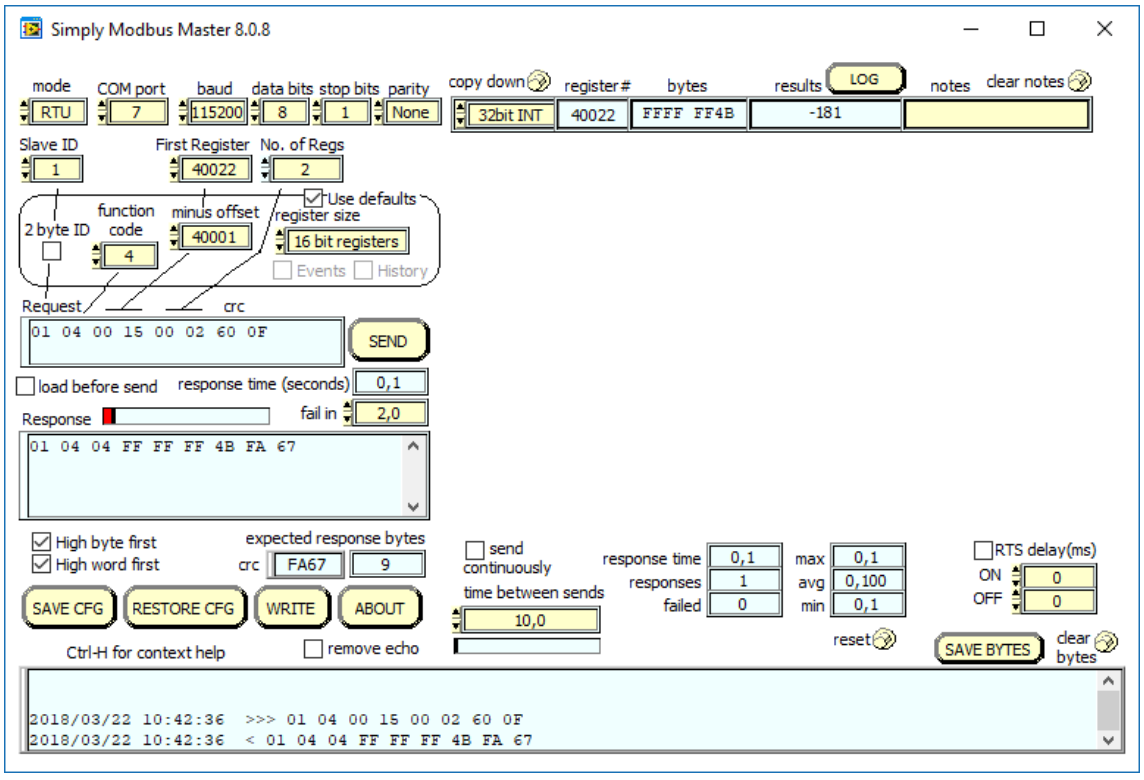

Figure 18: pH measuring with a negative input voltage

4.6 Example function code 0x04: Counter channel

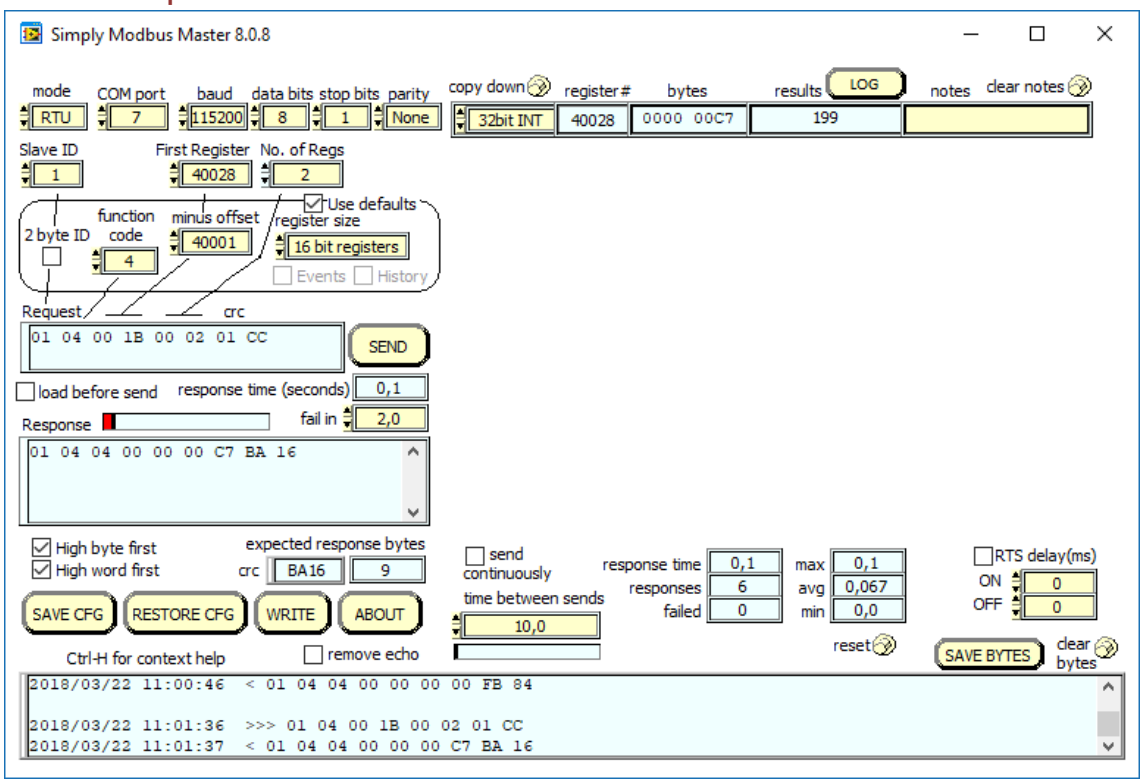

Figure 19: Example function code 0x04, counter measurement

4.7 Example function code 0x04: Firmware version

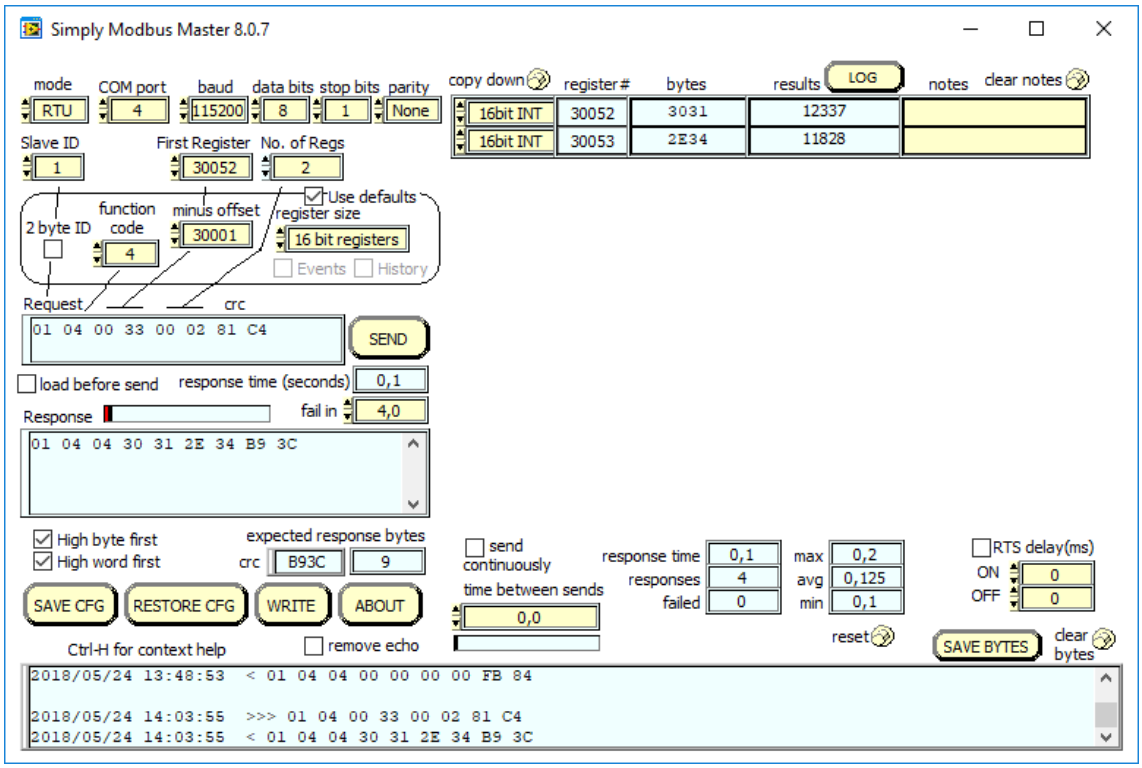

Figure 21: Example function code 0x04, Firmware version

4.8 Example function code 0x04: PCB model

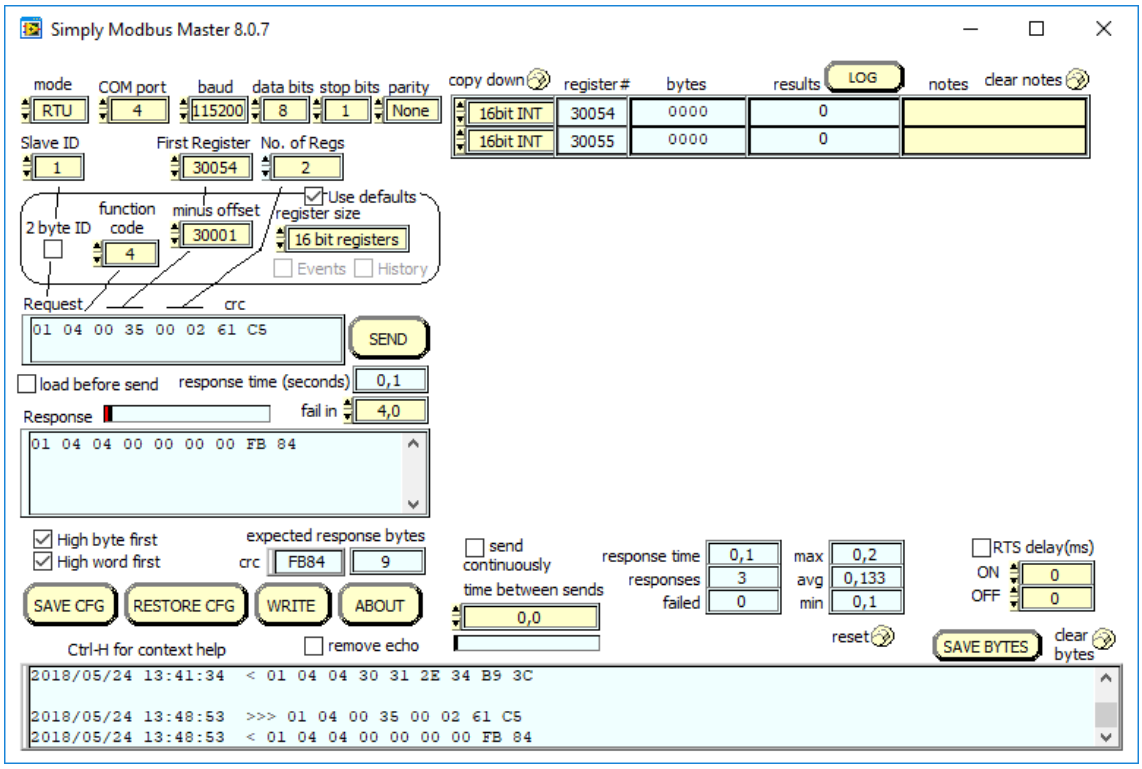

Figure 2220: Example function code 0x04, PCB model

4.9 Example of function 0x06

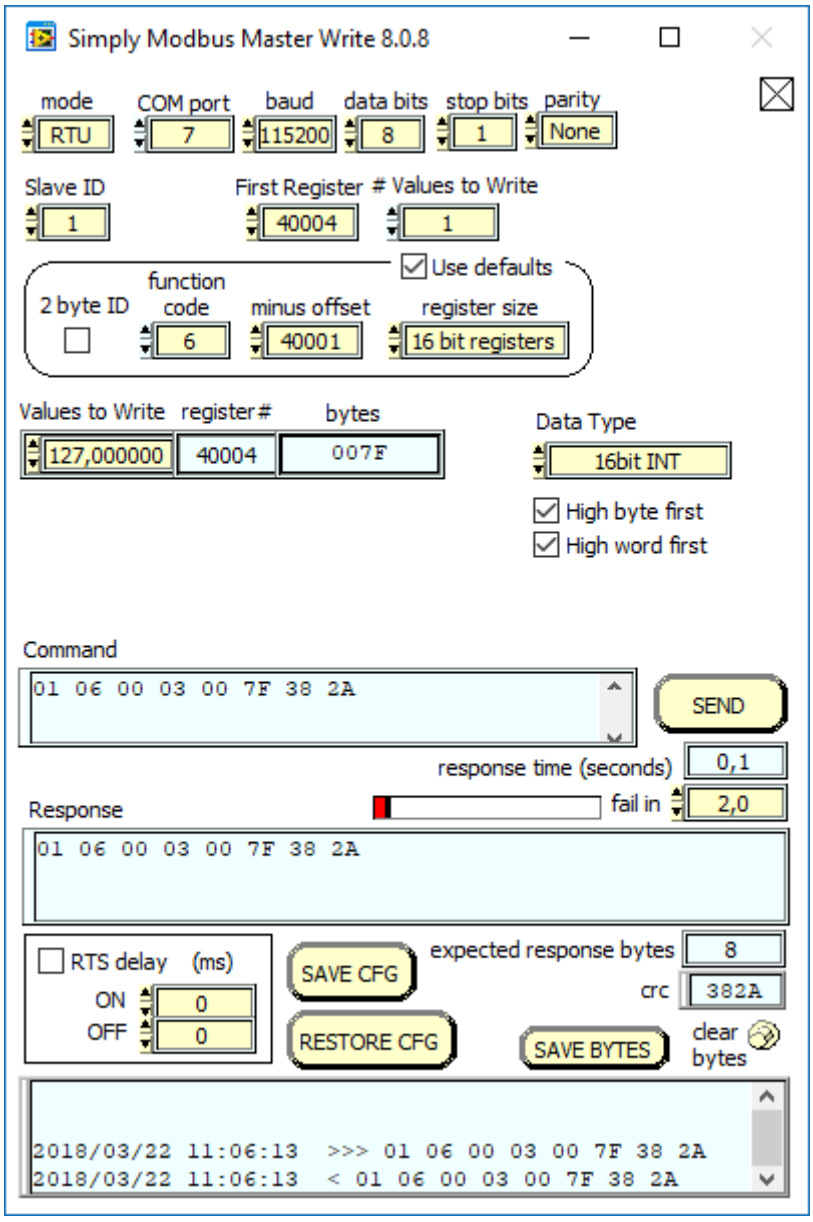

Figure 21: Example of function 0x06, channel AO_2, output 5V.

4.10 Examples of function 0x0F: Write Digital Outputs

The first example shows how to set all output to On state. Please note the final command sent by the master, not the tool configuration.

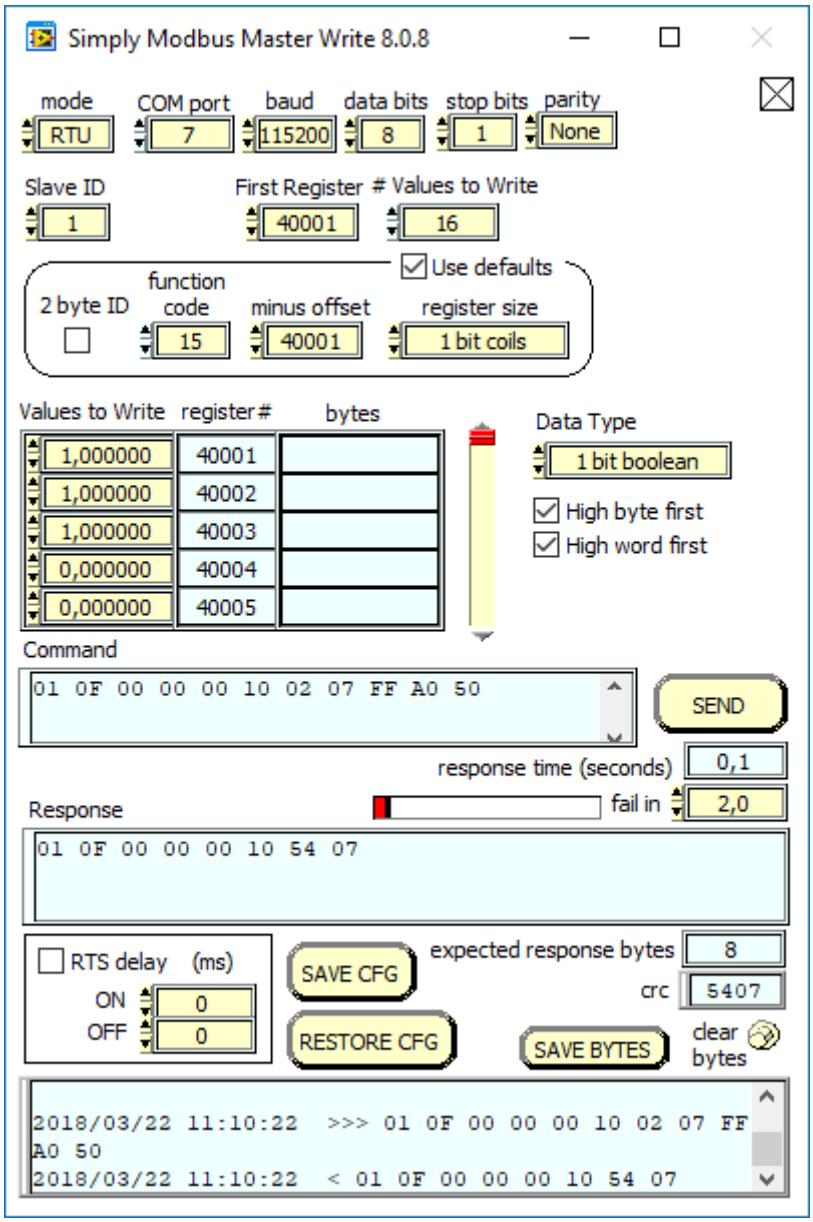

Figure 22: 0x0F function code. All outputs set to On state

The next example shows all outputs to off. Please note the final command sent by the master, not the tool configuration.

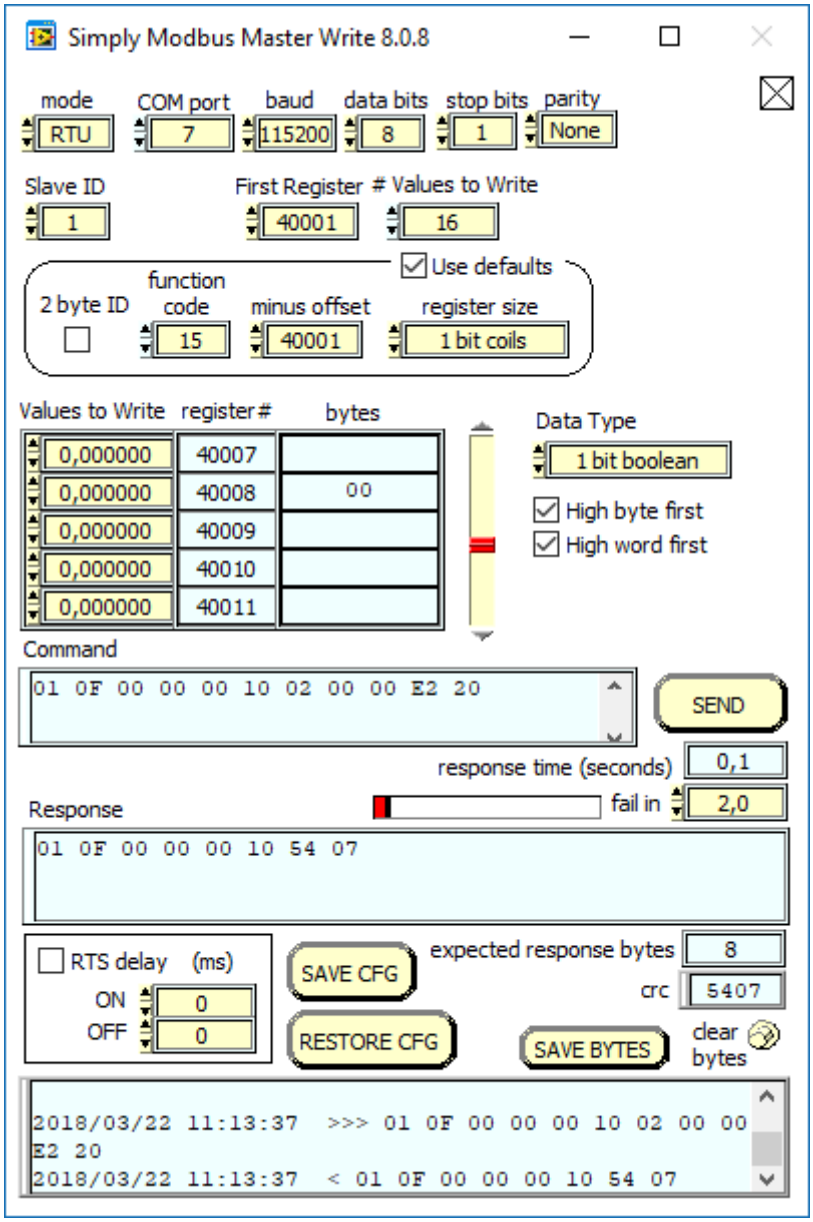

Figure 23: Example function code 0x0F, all outputs OFF state

The next example shows how the first output ON state, the second OFF, third ON, fourth OFF and so on.

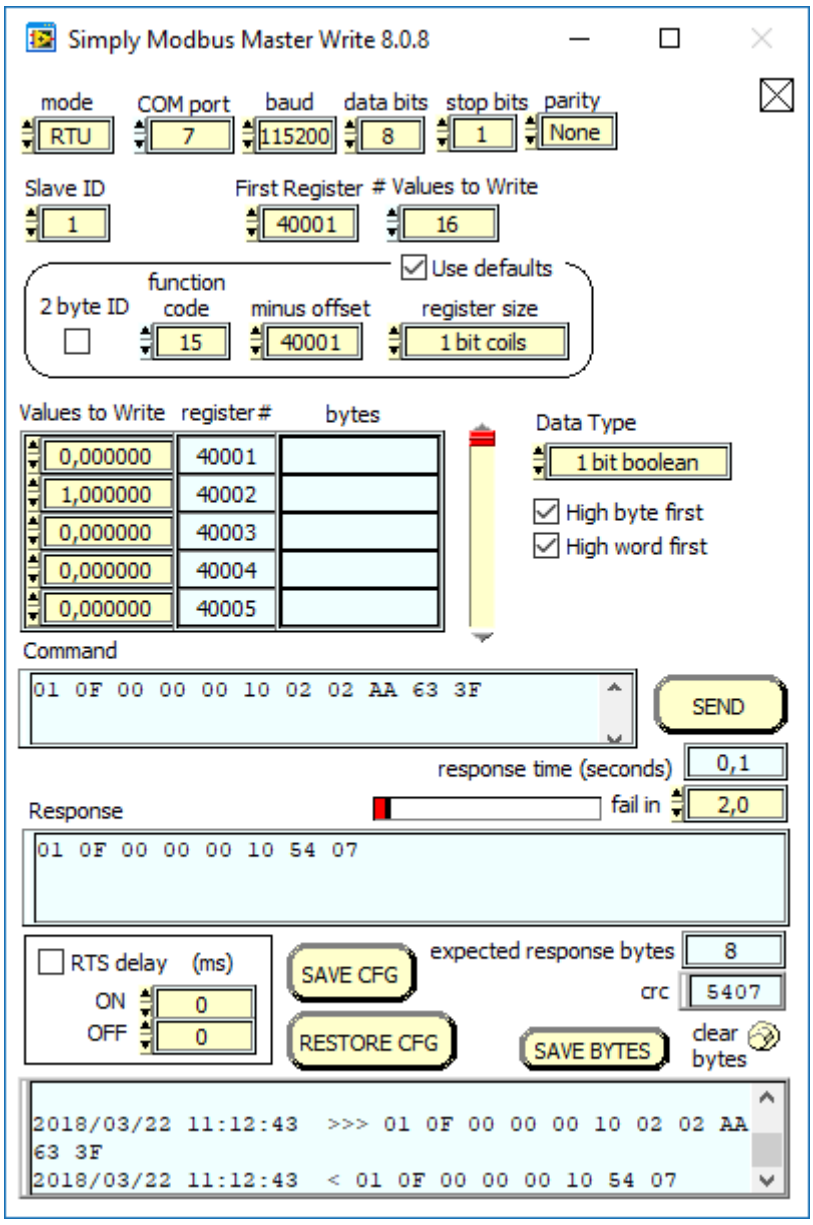

Figure 24: Example function code 0x0F, OFF-ON-OFF-ON-OFF....