Modbus RTU ASCII

Communication Protocol

SCT-2200 and SCT-1100

Interface and Setup Manual

RICE LAKE®
WEIGHING SYSTEMS

July 30, 2018

PN 190960

An ISO 9001 registered company

All information contained within this publication is, to the best of our knowledge, complete and accurate at the time of publication. Rice Lake Weighing Systems reserves the right to make changes to the technology, features, specifications and design of the equipment without notice.

The most current version of this publication, software, firmware and all other product updates can be found on our website:

www.ricelake.com

Contents

1.0 Introduction

The Modbus communication protocol defines the structure of messages and the communication mode between a primary device, which manages the system, and one or more secondary devices. It defines how the primary and the secondary establish and interrupt the communication, how the transmitter and receiver are identified, how the messages are exchanged and how the errors are detected.

Only the primary can start a transaction (Query) while the secondary devices respond by supplying the data requested to the primary or carrying out the actions requested in the query. The primary can either address one secondary unit or transmit a broadcast message to all. The secondary respond with a message (Response) to the queries which are individually addressed, but do not transmit any answer to the primary if there are broadcast messages.

A transaction can therefore have the following structures:

- Single question to a secondary / Single answer from the secondary
- Single broadcast message to all the secondary / No answer from the secondary

1.1 Modbus Serial Communication Mode

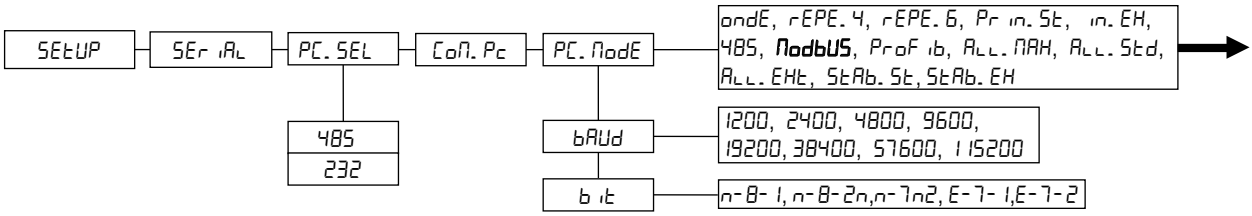

Figure 1-1. Communications Menu

To select the Modbus communication protocol mode:

1. Turn on the indicator, press ZERO or TARE during the countdown. tYPE displays.
2. Press ZERO or TARE to select SEtUP. Press PRINT.
3. Press ZERO or TARE to select SEr iAL. Press PRINT.
4. Press ZERO or TARE to select PC. SEL. Press PRINT. 485 displays.
5. Press ZERO or TARE to select 485 or 232. Press PRINT.
6. Press ZERO or TARE to select CoM. PC. Press PRINT. PCModE displays.
7. Press ZERO or TARE to select ModbUS.
8. Press PRINT to confirm. Mod. tYP displays.
9. Press PRINT to enter menu.
10. Select the type of protocol: ASCII or Binary (RTU). See Section 1.2 on page 2.
11. Press PRINT to confirm. Mod. Add displays.
12. Press PRINT to until Add. 485 displays.
13. Type in the serial address of the instrument.
14. Press PRINT to confirm.
15. Set *Baud Rate* and the *Serial Word Format*, in the bAUd and b it steps. See Section 1.1.

16. Press C several times until the display shows the message SAVEP.
17. Press PRINT to confirm the changes made or another key to not save.

1.2 Modbus Transmission Modes

1.2.1 ASCII or RTU (Binary)

Select the serial transmission mode as either ASCII or RTU. This determines how the information is packaged inside the message fields and how it is decoded.

American Standard Code for Information Interchange (ASCII)

The main advantage of this mode is that it allows for time intervals up to a second between a character and another during the transmission without provoking an error.

Each byte (8 bit) in a message is transmitted as two ASCII characters.

Remote Terminal Unit (RTU)

The main advantage of this mode, is its greater density of characters which allows for the transmission of higher volume of data equal to the baud rate.

Each byte (8 bit) in a message has 2 hexadecimal characters of 4 bits.

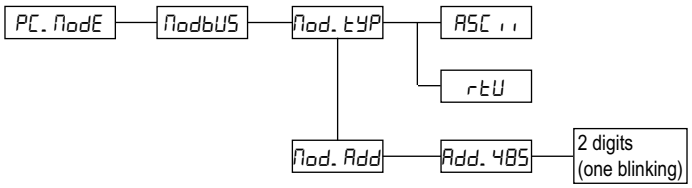

Figure 1-2. Modbus Menu

To select transmission mode:

1. Turn on the indicator, press ZERO or TARE during the countdown. tYPE displays.
2. Press ZERO or TARE to select SEtUP. Press PRINT.
3. Press ZERO or TARE to select SEr iAL. Press PRINT.
4. Press ZERO or TARE to select CoM.PC.
5. Press PRINT. PCModE displays.
6. Press ZERO or TARE to select ModbUS.
7. Press PRINT to confirm. Mod.tYP displays. Press PRINT.
8. Press ZERO or TARE to select serial transmission desired.
9. Press PRINT to confirm.

1.2.2 Serial Transmission Parameters

Set the Baud Rate (transmission speed) and Data Format (serial word format).

Suggested Formats:

- ASCII mode: n-7-2, E-7-1
- RTU mode: n-8-2, E-8-1

in which:

n = no parity (none)

E = even parity (Even)

Example

to use 8 data bits, the format is: n-8-2

n = No parity

2 = Stop Bit

To set serial transmission parameters:

1. Turn on the indicator, press ZERO or TARE during the countdown. TYPE displays.
2. Press ZERO or TARE to select SETUP. Press PRINT.
3. Press ZERO or TARE to select SER 1AL. Press PRINT.
4. Press ZERO or TARE to select COM. PC.
5. Press PRINT. PCMODE displays.
6. Press ZERO to select BAUD.
7. Press ZERO or TARE to select the desired baud rate. Press PRINT to save.
8. Press ZERO. bit displays. Press PRINT to enter settings.
9. Press ZERO or TARE to select the desired date format. Press PRINT to save.
10. Press PRINT to confirm.

IMPORTANT *The type of serial transmission (ASCII or RTU) and the communication parameters of the serial port (Baud Rate and Data Format) must be the same for each device connected to the MODBUS network.*

1.3 Component and Message Format

In both serial transmission modes (ASCII or RTU), a Modbus message is put by the transmitting device inside a frame, which has a known beginning and end point. This allows for the receiving devices to locate the beginning of the message, read the address part and determine which device it is addressed to (or all the devices, if the message is broadcast) and to know when the message is complete. In this way the incomplete messages can be detected and consequently indicated as errors.

The format of the messages, for the primary and secondary, includes:

- Address of the Device – that the primary has established the transaction
  The address 0 corresponds to a broadcast message transmitted to all the secondary devices.
- Function Code – defines the requested action
- Data – which must be transmitted
- Error Check – made up according to the CRC or LRC algorithm

These fields are described in detail in the following paragraphs.

1.3.1 Query and Response

Query

The function code tells the addressed secondary device which action must be made. The data bytes contain some additional information which the secondary needs in order to execute the function. The error check field gives the secondary a method in order to confirm the integrity of the message contents.

Response

If the secondary device gives a normal answer:

The function code is the echo of the query function code. The data bytes contain the data retrieved from the secondary device, like the registers' values or the states.

If a secondary device locates an error (format, parity, in CRC or LRC) or it is unable to execute the requested action:

The primary message is considered non valid and rejected and consequently the action will not be executed; a Response in which the function code is changed in order to indicate that is an error response and the data bytes contain a code which describes the error.

1.3.2 ASCII Frame Format

In the ASCII mode the messages start with the (:) character (ASCII 3A Hex) and end with CRLF (Carriage Return Line-Feed), of two characters (ASCII 0D and 0A Hex).

For all the other fields it is possible to transmit the 0..9 and A..F hexadecimal characters; the devices continuously monitor the network to locate the (:) character; when one of these is received, each device decodes the next field (field address) in order to verify whether the device is addressed.

Between one character and another of the message there may be various time intervals of up to one second; if there is a longer interval, the receiving device assumes that an error has taken place.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	N CHARS	2 CHARS	2 CHARS CRLF

Table 1-1. Typical ASCII Message Frame

1.3.3 RTU Frame Format

In the RTU mode the messages start with a silent interval that lasts at least a period equal to 3.5 times the time period of a character (T1-T2-T3-T4 time interval). The devices monitor continuously the transmission bus, also during the silent intervals. When the first field (the address) is received, each device decodes it in order to verify whether the device is addressed.

For each field the characters which may be transmitted are all the decimal values from 0 to 255.

After the last transmitted character there will be a silent interval equal to at least 3.5 times the time period of a character, indicating the end of the message; after this a new message can start.

The entire frame must be transmitted as a continuous stream. If there is a silent interval greater than the time period of 1.5 characters before the completion of the frame, the receiving device considers the incomplete message as ended and assumes that the next byte is the address field of a new message.

In the same way, if a new message starts before a time period equal to 3.5 characters following a previous message, the receiving device considers it a continuation of the previous message. This causes an error, and consequently the value in the final field of the CRC will not be valid, due to the combination of the two messages.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8 BITS	8 BITS	N * 8 BITS	16 BITS	T1-T2-T3-T4

Table 1-2. Typical RTU Message Frame

1.3.4 Device Address

The Modbus transactions always involve the primary device, which manages the line, and a secondary device at a time (except for the broadcast messages). In order to identify the message consignee, the numeric address of the selected secondary device (one byte: two characters for the ASCII, eight bits for the RTU) is transmitted as the first field of the frame. Each secondary device is assigned a different address number so that it can be identified. When the secondary device transmits its answer, its address is entered in the response's field address in order that the primary device knows which secondary device is responding.

Valid addresses for the secondary devices are within a range from 0 to 247.

Example

- *0 .....broadcast address (all the secondary devices)*
- *1 .....minimum possible address for the secondary*
- *247 ....maximum possible address for the secondary*

1.3.5 Function Code

The field of the frame function code of a message contains two characters (ASCII) or eight bits (RTU). Valid codes are within the range from 1 to 255 decimals.

When a message is transmitted from a primary to a secondary device the function code field indicates to the secondary what kind of action should be executed (for example the reading of the Input Registers, etc.).

When a secondary responds to the primary, it uses the function code field in order to indicate either a normal response (without errors) or a type of error which has already taken place (called exception responses). For a normal response, the secondary simply echoes the original function code, while for an exception response it gives back a code which is equivalent to the original function code, but with the most significant bit set at the 1 logic value.

Besides the modification of the function code for an exception response, the secondary enters a single code within the data field of the response message, in order to tell the primary which type of error has taken place or the reason for the exception.

1.3.6 Data

The data field is made by using groups of two hexadecimal digits, in the range from 00 to FF Hex. This can be made by a pair of ASCII characters, or by RTU characters, in accordance with the network's serial transmission mode.

The field data of the messages transmitted from the primary to the secondary devices contains additional information which the secondary must use in order to carry out the action defined by the function code.

1.3.7 Error Check

The contents of the error check field depend on the used Modbus transmission mode (ASCII or RTU) because there are two distinct error check methods. In particular:

ASCII Mode

The communication strings are checked by an LRC (Longitudinal Redundancy Check) algorithm, see Section 3.3.

The error check field contains two ASCII characters, which are the result of the calculation of an LRC algorithm executed on the contents of the message, excluding the initial character (:) and the CRLF terminator.

RTU Mode

The communication strings are checked by a CRC (Cyclical Redundancy Check) algorithm, see Section 3.2.

The error check field contains 16 bits (implemented as 2 bytes of 8 bits), which are the result of the calculation of a CRC algorithm executed on the contents of the message.

This field is the last of the message and the first byte is the one of the low order and is followed by one of the high order, which is the last one of the frame.

One may find further details regarding the error check and the creation of the LRC and CRC algorithms in chapter 3.

1.3.8 Message Components in ASCII and RTU

The following tables show an example of the fields inside a Modbus message, for a Query as well as for a normal Response; in both cases the fields' content is shown in hexadecimals and how the message is organized (framed) in ASCII or RTU mode.

Query: Read Input Registers to the 01 Secondary Device address, for the reading of the contents of 3 registers starting from register n°8.

Field Name	Example (Hex)	ASCII: characters	RTU: 8-bit field
Heading		: (colon)	None
Secondary Address	01	0 1	0000 0001
Function	04	0 4	0000 0100
Start Address (HIGH)	00	0 0	0000 0000
Start Address (LOW)	08	0 8	0000 1000
Number of Registers (HIGH)	00	0 0	0000 0000
Number of Registers (LOW)	03	0 3	0000 0011
Error Check		LRC (2 characters)	CRC (16 bits)
Terminator		CR LF	None
Nr. of total bytes		17	8

Response

Field Name	Example (Hex)	ASCII: characters	RTU: 8-bit field
Heading		: (colon)	None
Secondary Address	01	0 1	0000 0001
Function	04	0 4	0000 0100
Number of bytes	06	0 6	0000 0110
Data (HIGH)	02	0 2	0000 0010
Data (LOW)	2B	2 B	0010 1011
Data (HIGH)	00	0 0	0000 0000
Data (LOW)	00	0 0	0000 0000
Data (HIGH)	00	0 0	0000 0000
Data (LOW)	63	6 3	0110 0011
Error Check		LRC (2 characters)	CRC (16 bits)
Terminator		CR LF	None
Nr. of total bytes		23	11

Table 1-3. Response

In the Response of the Secondary there is the Function Echo indicating that it's a normal type of answer.

The *Number of Bytes* field specifies how many groups of 8-bit data are given back, in other words, the number of bytes of the *Data* fields is shown, for the ASCII as well as for the RTU: in the ASCII mode this value is half of the total number of the ASCII characters in the data (each hexadecimal value of 4 bits requires an ASCII character, therefore two ASCII characters must be adjacent in the message in order to contain each 8-bit data item).

Example

The 63 Hex value is transmitted as a 8-bit byte in RTU mode (01100011); the same value transmitted in ASCII mode requires 2 bytes: for ASCII 6 (0110110) and 3 (0110011). The Number of Bytes field contains this data an 8-bit item, without taking into consideration the packing mode of the characters (ASCII or RTU).

2.0 Modbus Functions

Each Modbus function detailed in this section and is made up of the following:

- QUERY (Primary Device request to Instrument)
- RESPONSE (Instrument response to Primary Device)

Note

ASCII transmission mode: each character is an ASCII type character (made up of 8 bits).

RTU transmission mode: each character is an Hexadecimal type of character (made up of 4 bits).

With 0x or Hex before a number it means that it has to do with a hexadecimal value.

2.1 Supported Functions

Table 2-1 contains active Modbus functions for the SCT instrument.

Function Code	Description
01 (0x01)	Read Coils Status
03 (0x03)	Read Holding Registers
04 (0x04)	Read Input Registers
05 (0x02)	Write Single Coil
06 (0x06)	Write Single Register
15 (0x0F)	Write Multiple Coils
16 (0x10)	Write Multiple Registers
Parentheses indicate the hexadecimal values	

Table 2-1. Active Modbus functions

2.2 Transmission Strings

In the following paragraphs the functions (shown in Table 2.1) supported by the instrument are described in detail; for this purpose one uses the following classification for the message fields:

- Address: A = 1 byte for the instrument address (secondary).
- Function: Code or Number of the function to be executed.
- Number of bytes: represents the number of bytes which make up a datum.
- Error Check (CRC / LRC): for the error check, in the RTU and in the ASCII transmission modes it's always 2 bytes. (CRC = Cyclical Redundancy Check, LRC = Longitudinal Redundancy Check; see Section 3.0 on page 10)

Other fields for the message frames are described in detail in the various single functions.

Note

The following data are defined, on which the functions operate:

- *Coils Status: written by the Primary → read by the Instrument and the Primary*
- *Holding Registers: written by the Primary → read by the Instrument and the Primary*
- *Input Registers: written by the Instrument → read by the Primary*

The Registers are described in detail in Chapter 5.

The switch buffer is made of 100 bytes, therefore it is not possible to read a registers number that exceed the transmission buffer capacity and it is not possible to write a registers number that exceed the reception buffer capacity.

2.2.1 Read Coils Status / Holding / Input Registers

Functions 1,3 and 4: (01,03 And 04 Hex)

They read the contents of the secondary instrument's registers (the instrument writes); the broadcast communication is not supported.

Query

One specify the registers / coils data area to read (Function), the Initial Register (1st Register Address) from which the reading starts and the Number of Registers which must be read (Nr. of Registers). The registers are addressed from 0: in this way the registers from 1 to 16 are addressed as 0 to 15.

Address	Function	Address 1st Register		Nr. of Registers		Error Check
		High	Low	High	Low	
A	XX	00	08	00	01	CRC / LRC

Response

The response message is made up of 2 bytes for each read register, with the binary content aligned on the right in each byte. For each register the first byte contains the most significant bits and the second byte contains the least significant bits.

Address	Function	Address 1st Register		Nr. of Registers		Error Check
		High	Low	High	Low	
A	XX	02	02	00	0A	CRC / LRC

Example: A = 01

Query: 1st Register address = 00 08; Number of Registers = 00 01

Response: 1st Register = 00 0A

Note Maximum number of registers readable with one request: 49

2.2.2 Preset Coil / Single Register

Functions 5 and 6: (05 and 06 Hex)

It allows a single register (which the instrument or secondary goes to read) to be set to a determined value.

The broadcast transmission of this command is allowed and the same register can be set in all the connected secondary devices.

Query

Specify the register / coil data area to write (*Function*), the Register Address and the relative value (*Register Value*). The registers are addressed starting from 0: so registers 1 to 16 are addressed as 0 to 15.

Address	Function	Address 1st Register		Nr. of Registers		Error Check
		High	Low	High	Low	
A	XX	00	01	00	03	CRC / LRC

Response

It is the echo of the Query.

Address	Function	Address 1st Register		Nr. of Registers		Error Check
		High	Low	High	Low	
A	XX	00	01	00	03	CRC / LRC

Example: A = 01

Query: Register Address = 00 01; Register Value = 00 03

Response: Register Address = 00 01; Register Value = 00 03

2.2.3 Preset Multiple Coils / Registers

Function 15 and 16: (0F and 16 Hex)

Allows various registers (which the instrument or secondary goes to read) to be set to a determined value.

Query

Specify the coils/registers data area to write (*Function*), specified the address of the First Register which must be set (*1st Register address*), the Number of Registers to be written (*Nr. of Registers*) starting with the first, the number of bytes transmitted for the values of the registers (2 bytes for each register) or *Nr. of Bytes* and then the values to be assigned to the registers (*1st Register value* of 2 bytes, *2nd Register Value, etc.*) starting from the first one addressed.

Address	Function	1st Register Address		Nr. of Registers		Nr. of bytes	1st Register Value		2nd Register Value		Error Check
		High	Low	High	Low		High	Low	High	Low	
A	XX	00	00	00	02	04	00	00	00	00	CRC / LRC

Response

The response includes the identification of the modified registers (1st Register address and Nr. of Registers).

Address	Function	Address 1st Register		Nr. of Registers		Error Check
		High	Low	High	Low	
A	XX	00	00	00	03	CRC / LRC

Example: A = 01

Query: 1st Register Address = 00 00; Nr. of Registers = 00 02; Nr. of bytes = 04;
1st Register Value = 00 00; 2nd Register Value = 00 00;

Response: 1st Register Address = 00 00; Nr. or registers = 00 02;

Note *Maximum number of registers write-able with one request:*

* *RTU mode: 45*
* *ASCII mode: 20*

3.0 Error Check Methods

The Modbus serial communication uses two error check types:

- The character or parity (even or uneven), can be applied optionally to each character (see Par. 1.3, Data Format).
- The frame (LRC or CRC algorithms), applied to the entire message.

The communication strings are checked by a Cyclical Redundancy Check (CRC) type algorithm in the case of binary (RTU) and the Longitudinal Redundancy Check type (LRC) for the ASCII communication.

Both the check on the character as well as the one on the frame of the message is created in the Primary device and applied to the contents of the message before the transmission. The secondary device checks each character and the entire frame of the message during the reception.

3.1 Parity Check

It is possible to configure the parity check in the following way:

- n – no parity (none)
- E – even parity (Even)

This determines how the parity is set in each character.

3.2 Cyclical Redundancy Check (CRC) Algorithm

(RTU mode)

In the RTU transmission mode, the messages include an error check field based on a CRC method, which checks the contents of the entire message and is applied without any regard to any parity method used for the single characters. The CRC field is made up of 2 bytes (containing a binary value of 16 bits) and is calculated from the transmitting device, which puts the CRC at the end of the message. The receiving device calculates again the CRC during the reception of the message, and compares the calculated value with the actual value received in the CRC field. If the two values are not the same an error has taken place.

3.2.1 CRC Procedure

Use the following steps to create a CRC procedure:

1. Load a 16-bit register with FFFF Hex (all 1) to create a CRC Register.
2. OR-exclusive with the first byte of the message and the least significant byte of the CRC Register at 16 bit. The result is inserted in the CRC Register.
3. The CRC Register is shifted by 1 bit to the right (toward the LSB), a 0 is inserted in the place of the MSB. The LSB is extracted and examined.
4. If LSB = 0 – repeat Step 3 is to be repeated. (another shift)
   If LSB = 1 – the OR-ex is made between the CRC Register and the A001 Hex (1010 0000 0000 0001) polynomial value.
5. Repeat Steps 3 and 4, until 8 shifts have been made; after this a byte of 8 bits has been completely processed.
6. Steps 2-5 are repeated for the next byte of 8 bits of the message. One continues until all the bytes are processed.
7. The final contents of the CRC Register are the CRC Value.
8. When the CRC is inserted within the message, its bytes (high and low) must be exchanged as described below.

3.2.2 Place CRC in a Message

When the 16 bits of the CRC (2 bytes) are transmitted in the message, the least significant byte must be transmitted first, followed by the most significant byte.

Example: if the CRC value is 1241 Hex (0001 0010 0100 0001)

Addr	Func	Data Count	Data	Data	Data	Data	CRC LOW 41	CRC HIGH 12

Sequence of the CRC bytes

Example: C Language of the CRC creation:

The function creates internally the swapping of the high and low bytes of the CRC. The bytes are already exchanged in the CRC value which is given back by the function, which can then be placed directly in the message for the transmission.

The function uses two arguments:

```
unsigned char *puchMsg;          A pointer to the message buffer which contains the
                                 binary data to be used for creating the CRC for
                                 generating the CRC
unsigned short usDataLen;        The quantity of bytes in the message buffer
```

The function gives back the CRC value as an unsigned short.

CRC Generation Function

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg;              //message to calculate CRC upon
unsigned short usDataLen;            //quantity of bytes in message
{
    unsigned char uchCRCHi = 0xFF;   //high CRC byte initialized
    unsigned char uchCRCLo = 0xFF;   //low CRC byte initialized
    unsigned uIndex;                 //will index into CRC lookup table

while (usDataLen--)                  //pass through message buffer
{
uIndex = uchCRCHi ^ *puchMsg++;       //calculate the CRC
uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex];
uchCRCLo = auchCRCLo[uIndex];
}
return (uchCRCHi << 8 | uchCRCLo);
}
```

3.3 Longitudinal Redundancy Check (LRC) Algorithm

(ASCII mode)

In the ASCII transmission mode the messages include an error check field based on an LRC method which checks the contents of the message, except for the initial character (: or colors) and the two CRLF characters. The algorithm is applied without taking into consideration of any parity check method used for the single characters of the message.

The LRC field is of one byte and contains a binary value of 8 bits which is calculated by the transmitting device, which puts the LRC at the end of the message. The receiving device calculates a LRC during the reception and compares the calculated value with the actual value received in the LRC field. If the two values are not the same an error is shown.

The LRC is calculated by then summing together the bytes (8 bit) of the message, discarding the carry values and then making the complement at 2 of the result. It uses the contents of the ASCII message fields, with the exception of the (:) character which starts the message and the two CRLF characters which end it.

3.3.1 Create LRC Procedure

1. Sum all the bytes of the message, excluding the (:) character and the CRLF, within the 8-bit field. In this way the carry values are discarded.
2. Subtract the resulting value from step 1. to FF Hex (8 bits all at 1), obtaining in this way the *Complement to 1*.
3. Add 1 to obtain the *Complement to 2*.

3.3.2 Place LRC in the message

When the 8 bits of the LRC (2 ASCII characters) are transmitted in the message, the most significant character must be transmitted first, followed by the least significant one.

Example: if the LRC value is 61 Hex (0110 0001)

Colon (:)	Addr	Func	Data Count	Data	Data	Data	Data	LRC High 6	LRC Low 1	Data	Data

Sequence of the LRC bytes

Example in the C language for creating the LRC

A functioning example for creating the LRC in the C language is shown below. The function uses two arguments:

```
unsigned char *auchMsg;          A pointer to the message buffer which contains the binary
                                 data to be used for creating the LRC
unsigned short usDataLen;        Quantity of bytes in the message buffer
```

The function gives back the LRC value as an unsigned char.

LRC Creation Function

```
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg;          //message to calculate
unsigned short usDataLen;        //LRC upon quantity of bytes in message
{
unsigned char uchLRC = 0;        //LRC char initialized
while (usDataLen--)              //pass through message
uchLRC += *auchMsg++;            //buffer add buffer byte without carry
return((unsigned char)(-((char_uchLRC)));      //return twos complement
```

4.0 Exceptions

In a normal response (Normal Response) the secondary device echoes the Function Code of the Query, putting it in the Response Function field. All the function codes have their own most significant bit (MSB) set at 0 (values less than 80 Hex).

In an exception response (Exception Response) the secondary sets the MSB of the Function Code at 1 (equivalent to summing the value 80 Hex to the normal response code) in order to signal that an anomaly has taken place, and the Exception Code is given back in the Data Field, in order to indicate the type of exception.

4.1 Detected Exceptions

Code	Exception	Description
01	Illegal Function	The *Function Code* received by the Query is not supported or not valid
02	Illegal Data Address	The *Data Address* received in the Query is not an address supported by the secondary device or is not valid
03	Illegal data Value	A *Value in the Data field* of the Query is not a value acceptable by the secondary device or is not valid
06	Secondary Device Busy	Processing a command; Primary transmits again the message later, when secondary is free

Table 4-1. Active Modbus Exceptions

5.0 Data Areas

There are three data areas, *Input*, *Holding* and *Coils*, defined this way due to the primary's point of view: while the *Input* area is read by this device, the *Holding* and *Coils* ones are written.

The first two areas are organized in registers on which the Modbus protocol functions perform.

All the numeric values have the Big Endian format (the 1st byte is the most significant one) for the Data Input Area and the Data Output Area, while these have the Little Endian format (the 1st byte is the least significant one) for the setup area.

5.1 Input Registers Data Area

The input data area is read by the primary device (is therefore written by the instrument) and is made up of registers (input registers), of 2 bytes.

Reg. Nr.		Input Registers	
30001	(0)	Gross Weight Value	(byte 3)
		Gross Weight Value	(byte 2)
30002	(1)	Gross Weight Value	(byte 1)
		Gross Weight Value	(byte 0)
30003	(2)	Net Weight Value	(byte 3)
		Net Weight Value	(byte 2)
30004	(3)	Net Weight Value	(byte 1)
		Net Weight Value	(byte 0)

Format of the *GROSS WEIGHT* and *NET WEIGHT* values

Whole in absolute value (without decimals)

Example*:*

if 3 decimals are set, the value 3,000 is read 3000

if 2 decimals are set, the value 3,00 is read 300

Reg. Nr.		Input Registers	
30005	(4)	Input Status Register	(MSB)
		Input Status Register	(LSB)
30006	(5)	Command Status Register	(MSB)
		Command Status Register	(LSB)
30007	(6)	Output Status Register	(MSB)
		Output Status Register	(LSB)

Format of the Input Status Register value

See Section 5.1.1 on page 16.

Format of the Output Status Register value

See Section 5.1.2 on page 16.

Format of the Command Status Register value

See Section 5.1.3 on page 17.

Reg. Nr.		Input Registers	
30008	(7)	Nr. of last page read or written	(MSB)
		Nr. of last page read or written	(LSB)
30006	(5)	1st set-up page word	
		1st set-up page word	
– – – – – – – – – – – – – – – –			
30016	(15)	8th set-up page word	
		8th set-up page word	
30101	(100)	Software Vers. ("00")	byte 3
		Software Vers. (release)	byte 2
30102	(101)	Software Vers. (subrelease)	byte 1
		Software Vers. (bugrelease)	byte 0

Format of the Software Version (registers 100÷101)

Whole in absolute value (without points)

Example: the software version 05.06.00 is read 00050600

Reg. Nr.		Input Registers	
30103	(102)	ADC points channel 1	(byte 3)
		ADC points channel 1	(byte 2)
30104	(103)	ADC points channel 1	(byte 1)
		ADC points channel 1	(byte 0)
30105	(104)	ADC points channel 2	(byte 3)
		ADC points channel 2	(byte 2)
30106	(105)	ADC points channel 2	(byte 1)
		ADC points channel 2	(byte 0)
30107	(106)	ADC points channel 3	(byte 3)
		ADC points channel 3	(byte 2)
30108	(107)	ADC points channel 3	(byte 1)
		ADC points channel 3	(byte 0)
30109	(108)	ADC points channel 4	(byte 3)
		ADC points channel 4	(byte 2)
30110	(109)	ADC points channel 4	(byte 1)
		ADC points channel 4	(byte 0)
30111	(110)	Microvolts channel 1	(byte 1)
		Microvolts channel 1	(byte 0)
30112	(111)	Microvolts channel 2	(byte 1)
		Microvolts channel 2	(byte 0)
30113	(112)	Microvolts channel 3	(byte 1)
		Microvolts channel 3	(byte 0)
30114	(113)	Microvolts channel 4	(byte 1)
		Microvolts channel 4	(byte 0)
30115	(114)	Analog output value	(byte 1)
		Analog output value	(byte 0)

ADC Points and Microvolts (Registers 102÷114):

In Dependent channels and Transm modes ADC and µV values for more channels are available.

In Independent channels mode the values of one channel only are available, other registers are equal to zero.

The registers related to non configured channels are equal to zero.

5.1.1 Input Status Register

It is Input Register number 4; two bytes defined in the following way:

Bit	Description	Bit Meaning	
(LSB)		0	1
0	Net Weight Polarity	+	--
1	Gross Weight Polarity	+	--
2	Weight Stability	NO	YES
3	Under load Condition	NO	YES
4	Overload Condition	NO	YES
5	Entered Tare Condition	NO	YES
6	Manual Tare Condition	NO	YES
7	Gross ZERO zone	Out of Zone 0	In Zone 0
(MSB)			
8	Input 1	DISABLED	ENABLED
9	Input 2	DISABLED	ENABLED
10	Not used		
11	Not used		
12	Not used		
13	Not used		
14	Displayed channel (low bit) ([1])		
15	Displayed channel (high bit) (from 0 to 3) ([1])		

Note *Table Note*

High Bit, Low Bit: 0 0 ® Channel 1 0 1 ® Channel 2

(15) (14) 1 0 ® Channel 3 1 1 ® Channel 4

5.1.2 Output Status Register

It is Input Register number 6; two bytes defined in the following way:

Bit	Description	Bit meaning	
(LSB)		0	1
0	RELAY 1	NOT EXCITED	EXCITED
1	RELAY 2	NOT EXCITED	EXCITED
2	RELAY 3	NOT EXCITED	EXCITED
3	RELAY 4	NOT EXCITED	EXCITED
4	RELAY 5	NOT EXCITED	EXCITED
5	RELAY 6	NOT EXCITED	EXCITED
6	Not used		
7	Not used		
(MSB)			
8	Not used		
9	Not used		
10	Not used		
11	Not used		
12	Not used		
13	Not used		
14	Not used		
15	Not used		

5.1.3 Command Status Register

It is Input Register number 5; two bytes defined in the following way:

High Byte –		Last received command
Low Byte –	low nibble	Counting of processed commands (module 16)
	high nibble	Result of last received command

In which the *Result of the last received command* can assume the following values:

OK = 0			Corrected and executed command
ExceptionCommandWrong	=	1	Incorrect command
ExceptionCommandData	=	2	Data in the incorrect command
ExceptionCommandNotAllowed	=	3	Command not allowed
ExceptionNoCommand	=	4	Inexistent command

5.2 Holding Registers Data Area

The *Holding* data area is written by the Primary (is therefore read by the instrument) and is made up of registers (holding registers), of 2 bytes.

Reg. Nr.		Input Registers	
40001	(0)	Command Register	(MSB)
		Command Register	(LSB)
40002	(1)	Parameter 1	(byte 3)
		Parameter 1	(byte 2)
40003	(2)	Parameter 1	(byte 1)
		Parameter 1	(byte 0)
40004	(3)	Parameter 2	(byte 3)
		Parameter 2	(byte 2)
40005	(4)	Parameter 2	(byte 1)
		Parameter 2	(byte 0)
40006	(5)	Not used	
		Not used	
40007	(6)	Not used	
		Not used	
40008	(7)	Not used	
		Not used	

Format of the Command Register value

See Section 5.2.8 on page 23

Reg. Nr.		Holding Registers	
40009	(8)	1st set-up page word	
		1st set-up page word	
– – – – – – – – – – – – – – – –			
40016	(15)	8th set-up page word	
		8th set-up page word	
40101	(100)	Gross Weight Value	(byte 3)
		Gross Weight Value	(byte 2)
40102	(101)	Gross Weight Value	(byte 1)
		Gross Weight Value	(byte 0)
40103	(102)	Net Weight Value	(byte 3)
		Net Weight Value	(byte 2)
40104	(103)	Net Weight Value	(byte 1)
		Net Weight Value	(byte 0)
40105	(104)	Tare Weight Value	(byte 3)
		Tare Weight Value	(byte 2)
40106	(105)	Tare Weight Value	(byte 1)
		Tare Weight Value	(byte 0)

Format of the GROSS WEIGHT, NET WEIGHT and TARE WEIGHT value
Whole in absolute value (without decimals)

Example: *if 3 decimals are set, the value 3,000 is read 3000*
if 2 decimals are set, the value 3,00 is read 300

Reg. Nr.		Holding Registers	
40107	(106)	Input Status Register	(MSB)
		Input Status Register	(LSB)
40108	(107)	Output Status Register	(MSB)
		Output Status Register	(LSB)

Format of the Input Status Register value
See Section 5.1.1 on page 16.

Format of the Output Status Register value
See Section 5.1.2 on page 16.

5.2.1 Setpoint SCT Family

N°Reg.		Holding Registers	
40109	(108)	Setpoint 1 ON temporary	(byte 3)
		Setpoint 1 ON temporary	(byte 2)
40110	(109)	Setpoint 1 ON temporary	(byte 1)
		Setpoint 1 ON temporary	(byte 0)
– – – – – – – – – – – – – – – –			
40119	(118)	Setpoint 6 ON temporary	(byte 3)
		Setpoint 6 ON temporary	(byte 2)
40120	(119)	Setpoint 6 ON temporary	(byte 1)
		Setpoint 6 ON temporary	(byte 0)
40121	(120)	Setpoint 1 OFF temporary	(byte 3)
		Setpoint 1 OFF temporary	(byte 2)
40122	(121)	Setpoint 1 OFF temporary	(byte 1)
		Setpoint 1 OFF temporary	(byte 0)
– – – – – – – – – – – – – – – –			
40131	(130)	Setpoint 6 OFF temporary	(byte 3)
		Setpoint 6 OFF temporary	(byte 2)
40132	(131)	Setpoint 6 OFF temporary	(byte 1)
		Setpoint 6 OFF temporary	(byte 0)
40133	(132)	Setpoint 1 ON permanent	(byte 3)
		Setpoint 1 ON permanent	(byte 2)
40134	(133)	Setpoint 1 ON permanent	(byte 1)
		Setpoint 1 ON permanent	(byte 0)
– – – – – – – – – – – – – – – –			
40143	(142)	Setpoint 6 ON permanent	(byte 3)
		Setpoint 6 ON permanent	(byte 2)
40144	(143)	Setpoint 6 ON permanent	(byte 1)
		Setpoint 6 ON permanent	(byte 0)
40145	(144)	Setpoint 1 OFF permanent	(byte 3)
		Setpoint 1 OFF permanent	(byte 2)
40146	(145)	Setpoint 1 OFF permanent	(byte 1)
		Setpoint 1 OFF permanent	(byte 0)
– – – – – – – – – – – – – – – –			
40155	(154)	Setpoint 6 OFF permanent	(byte 3)
		Setpoint 6 OFF permanent	(byte 2)
40156	(155)	Setpoint 6 OFF permanent	(byte 1)
		Setpoint 6 OFF permanent	(byte 0)

Note *no controls are executed:*

* *if the value is <= capacity*
* *off value <= on value*

The less significant word value is cut to the minimum scale division.

5.2.2 Weight in Transm Mode

N°Reg.		Holding Registers	
40201	(200)	Configured channels number	(byte 1)
		Configured channels number	(byte 0)
40202	(201)	Channel 1 Status Register	(MSB)
		Channel 4 Status Register	(LSB)
40203	(202)	Weight value Channel 1	(byte 3)
		Weight value Channel 1	(byte 2)
40204	(203)	Weight value Channel 1	(byte 1)
		Weight value Channel 1	(byte 0)
– – – – – – – – – – – – – – – –			
40211	(210)	Channel 4 Status Register	(MSB)
		Channel 4 Status Register	(LSB)
40212	(211)	Weight value Channel 4	(byte 3)
		Weight value Channel 4	(byte 2)
40213	(212)	Weight value Channel 4	(byte 1)
		Weight value Channel 4	(byte 0)

Format of the Canale X Status Register value
See Section 5.2.11 on page 24.

Format of the WEIGHT value
Whole in absolute value (without decimals)

Example: *if 3 decimals are set, the value 3,000 is read 3000*
if 2 decimals are set, the value 3,00 is read 300

5.2.3 Commands

N°Reg.		Holding Registers	
40231	(230)	Command Status Register	(MSB)
		Command Status Register	(LSB)
40232	(231)	Command Register	(MSB)
		Command Register	(LSB)
40233	(232)	Commands Parameters	
		Commands Parameters	
– – – – – – – – – – – – – – – –			
40236	(235)	Commands Parameters	
		Commands Parameters	

Format of the Command Status Register value
See Section 5.1.3 on page 17.

Format of the Command Register value
See Section 5.2.8 on page 23.

5.2.4 Alibi Memory

N°Reg.		Holding Registers	
40251	(250)	Last stored gross weigh	(byte 3)
		Last stored gross weigh	(byte 2)
40252	(251)	Last stored gross weigh	(byte 1)
		Last stored gross weigh	(byte 0)
40253	(252)	Last stored tare weigh	(byte 3)
		Last stored tare weigh	(byte 2)
40254	(253)	Last stored tare weigh	(byte 1)
		Last stored tare weigh	(byte 0)
40255	(254)	Last weigh id value	(byte 3)
		Last weigh id value	(byte 2)
40256	(255)	Last weigh id value	(byte 1)
		Last weigh id value	(byte 0)
40257	(256)	Alibi status register	(MSB)
		Alibi status register	(LSB)

Format of the Alibi status register value:
See Section 5.2.10 on page 24.

5.2.5 Setup

N°Reg.		Holding Registers	
40301	(300)	1st set-up word	(page 0)
		1st set-up word	(page 0)
– – – – – – – – – – – – – – – –			
40308	(307)	8th set-up word	(page 0)
		8th set-up word	(page 0)
40309	(308)	9th set-up word	(page 1)
		9th set-up word	(page 1)
– – – – – – – – – – – – – – – –			
40812	(811)	Last set-up word	(page 64)
		Last set-up word	(page 64)

Format of the words value:
See Section 5.2.12 on page 25.

5.2.6 Weight in One Word (Less Significant Word) and Status Repetition

N°Reg.		Holding Registers	
41101	(1100)	Gross Weight Value	(byte 1)
		Gross Weight Value	(byte 0)
41102	(1101)	Net Weight Value	(byte 1)
		Net Weight Value	(byte 0)
41103	(1102)	Tare Weight Value	(byte 1)
		Tare Weight Value	(byte 0)

Format of the GROSS WEIGHT, NET WEIGHT and TARE WEIGHT value
Whole in absolute value (without decimals)

Example: *if 3 decimals are set, the value 3,000 is read 3000*
if 2 decimals are set, the value 3,00 is read 300

N°Reg.		Holding Registers	
41104	(1103)	Input Status Register	(MSB)
		Input Status Register	(LSB)
41105	(1104)	Output Status Register	(MSB)
		Output Status Register	(LSB)

Format of the Input Status Register value
See Section 5.1.1 on page 16.

Format of the Output Status Register value
See Section 5.1.2 on page 16.

5.2.7 Weights in Transm Mode

N°Reg.		Holding Registers	
41201	(1200)	Configured channels number	(byte 1)
		Configured channels number	(byte 0)
41202	(1201)	Channel 1 Status Register	(MSB)
		Channel 1 Status Register	(LSB)
41203	(1202)	Weight value Channel 1	(byte 1)
		Weight value Channel 1	(byte 0)
41204	(1203)	Channel 2 Status Register	(MSB)
		Channel 2 Status Register	(LSB)
41205	(1204)	Weight value Channel 2	(byte 1)
		Weight value Channel 2	(byte 0)
41206	(1205)	Channel 3 Status Register	(MSB)
		Channel 3 Status Register	(LSB)
41207	(1206)	Weight value Channel 3	(byte 1)
		Weight value Channel 3	(byte 0)
41208	(1207)	Channel 4 Status Register	(MSB)
		Channel 4 Status Register	(LSB)
41209	(1208)	Weight value Channel 4	(byte 1)
		Weight value Channel 4	(byte 0)

Format of the Canale X Status Register value
See Section 5.2.11 on page 24.

Format of the WEIGHT value
Whole in absolute value (without decimals)

Example: *if 3 decimals are set, the value 3,000 is read 3000*
if 2 decimals are set, the value 3,00 is read 300

5.2.8 Command Register

It is the Output Register number 0. It is made up of two bytes and can take on the following values, which correspond to implemented commands described in table 5.2.1.

Execution of a Command

The execution of a command happens when the contents of the Command Register vary (therefore to repeat the last command one should first set the Command register at the NO COMMAND value, and then at the command value).

Implemented Command	Command Register Value	Description
NO_COMMAND	0 (0000 Hex)	No Command
ZERO_REQUEST	1 (0001 Hex)	Execution of SCALE ZERO
TARE_REQUEST	2 (0002 Hex)	Execution of AUTOMATIC TARE
TAREMAN_REQUEST	3 (0003 Hex)	Execution of MANUAL TARE; (the value is to be entered in Parameter 1); See Note 1
NET_SWITCH_REQUEST	4 (0004 Hex)	Display switching onto the NET WEIGHT; See Note 2
GROSS_SWITCH_REQUEST	5 (0005 Hex)	Display switching on the GROSS WEIGHT; See Note 2
CHANNEL_1_REQUEST	6 (0006 Hex)	Switching onto CHANNEL 1
CHANNEL_2_REQUEST	7 (0007 Hex)	Switching onto CHANNEL 2
CHANNEL_3_REQUEST	8 (0008 Hex)	Switching onto CHANNEL 3
CHANNEL_4_REQUEST	9 (0009 Hex)	Switching onto CHANNEL 4
SET_OUTPUT	25 (0019 Hex)	Relay setting; See Note 3
READ_ALIBI	30 (001E Hex)	Weigh reading on alibi trough setup page (holding registers 8-15); See Note 4
WRITE_ALIBI	31 (001F Hex)	Storage of weigh on alibi

Table 5-1. Commands

Note *Table Notes*

1. *Format of the Parameter 1 and Parameter 2 values; for the MANUAL TARE (only Param1)*

   Example:if 3 decimals are set, in order to enter the value 3,000 – write 3000
   if 2 decimals are set, in order to enter the value 3,00 – write 300

2. *Functions active only in NTGS mode (net/gross switch).*

3. *RELAY Setting*

   The status of the relays is settable through Parameter 1:

   Parameter 1:

   bit 0 RELAY 1 in which bit 0 = 1 • RELAY 1 CLOSED; bit 0 = 0 • RELAY 1 OPEN
   bit 1 RELAY 2 in which bit 1 = 1 • RELAY 2 CLOSED; bit 1 = 0 • RELAY 2 OPEN
   bit 2-15 (not used)

   Format of Parameter 1 and Parameter 2 Values for the RELAYS:

   → *Bit configuration*

   In the case a relay is linked to a set point, the command relative to that relay is ignored.

   The writing of the set point values does not cause the automatic saving in flash; these are only temporarily set. To save these in flash one should execute the WRITE_FLASH command.

4. *Weigh Reading on Alibi*

   To read a weight stored in the ALIBI trough setup page (Holding Registers 8-15), set the rewriting number in Parameter 1 and the weigh number (ID) in Parameter 2.

   Format of the Parameter 1 and Parameter 2 values:

   Whole in absolute value (without decimals)

5.2.9 Contents of Setup Page with Reading Alibi Command

	Input Data Area (N° Byte)	Description	
ALIBI PAGE (16 bytes)	16	Stored gross weight value	(byte 3)
	17	Stored gross weight value	(byte 2)
	18	Stored gross weight value	(byte 1)
	19	Stored gross weight value	(byte 0)
	20	Stored tare weight value	(byte 3)
	21	Stored tare weight value	(byte 2)
	22	Stored tare weight value	(byte 1)
	23	Stored tare weight value	(byte 0)
	24	ID: Weigh number	(byte 3)
	25	ID: Weigh number	(byte 2)
	26	ID: Weigh number	(byte 1)
	27	ID: Weigh number	(byte 0)
	28	Alibi status register	(MSB)
	29	Alibi status register	(LSB)
	30	*Not used*	
	31	*Not used*	

5.2.10 Alibi Status Register

It is the Holding Register number 7 after the READING ALIBI command execution; two bytes defined in the following way.

BIT	MEANING
bit from 7 to 0	Number of rewritings (from 0 to 255).
bit from 10 to 8	Number of scale (from 1 to 4).
bit 11	Type of tare; bit 11 = 1 → manual tare; bit 1 = 0 → null or semi-automatic tare
bit 12	Not used
bit 13	Not used
bit 14	Not used
bit 15	Not used

Note *It is possible to read the last stored weigh through the Holding registers from 250 to 256.*

5.2.11 Channel X Status Register

Bit	Description	Bit meaning	
(LSB)		0	1
0	Weight Polarity	+	--
1	Weight Stability	NO	YES
2	Underload Condition	NO	YES
3	Overload Condition	NO	YES
4	Gross ZERO zone	Out of Zone 0	In Zone 0
5	Not used		
6	Not used		
7	Not used		
(MSB)			
8-15	Not used		

5.2.12 Setup Area

The setup area is memorized in flash (1024 bytes) and consists of 64 pages (from 0 to 63). With an approved instrument, it's not possible to write the metric parameters, between page 0 and the first half of page 38. It is possible to write only the data between the second half of page 38 and page 63.

By writing one of the pages between 0 and 37 when the instrument is approved the result of the command is *ExceptionCommandNotAllowed*, by writing instead the others one obtains the *CommandOk*. Page 38, in any case is not completely copied, but only the second half of it.

	N°Reg.	Holding Registers	Description	
Area Setup: PAGE 5 (16 bytes)	340			
	341			
	342			
		Byte 3	RANGE 1 channel 1	(LSB)
	343	Byte 2	RANGE 1 channel 1	
		Byte 1	RANGE 1 channel 1	
	344	Byte 0	RANGE 1 channel 1	(MSB)
		Byte 3	RANGE 2 channel 1	(LSB)
	345	Byte 2	RANGE 2 channel 1	
		Byte 1	RANGE 2 channel 1	
	346	Byte 0	RANGE 2 channel 1	(MSB)
	347			
Area Setup: PAGE 6 (16 bytes)	348			
		Byte 1	RANGE 1 channel 1 Division	(LSB)
	349	Byte 0	RANGE 1 channel 1 Division	(MSB)
		Byte 1	RANGE 2 channel 1 Division	(LSB)
	350	Byte 0	RANGE 2 channel 1 Division	(MSB)
	351			
		Byte 0	Channel 1 Decimals	
	352	Byte 0	Channel 1 Unit of Measure; See Note 1	
	353			
	354			
	355			

	N°Reg.	Holding Registers	Description	
Area Setup: PAGE 14 (16 bytes)	412	Byte 3	RANGE 1 channel 2	(LSB)
		Byte 2	RANGE 1 channel 2	
	413	Byte 1	RANGE 1 channel 2	
		Byte 0	RANGE 1 channel 2	(MSB)
	414	Byte 3	RANGE 2 channel 2	(LSB)
		Byte 2	RANGE 2 channel 2	
	415	Byte 1	RANGE 2 channel 2	
		Byte 0	RANGE 2 channel 2	(MSB)
	416			
	417			
	418	Byte 1	RANGE 1 channel 2 Division	(LSB)
		Byte 0	RANGE 1 channel 2 Division	(MSB)
	419	Byte 1	RANGE 2 channel 2 Division	(LSB)
		Byte 0	RANGE 2 channel 2 Division	(MSB)
Area Setup: PAGE 15 (16 bytes)	420			
	421	Byte 0	Channel 2 Decimals	
		Byte 0	Channel 2 Unit of Measure; See Note 1	
	422			
	423			
	424			
	425			
	426			
	427			
Area Setup: PAGE 22 (16 bytes)	476			
	477			
	478			
	479			
	480			
	481			
		Byte 3	RANGE 1 channel 3	(LSB)
	482	Byte 2	RANGE 1 channel 3	
		Byte 1	RANGE 1 channel 3	
	483	Byte 0	RANGE 1 channel 3	(MSB)
		Byte 3	RANGE 2 channel 3	(LSB)

	N°Reg.	Holding Registers	Description	
Area Setup: PAGE 23 (16 bytes)	484	Byte 2	RANGE 2 channel 3	
		Byte 1	RANGE 2 channel 3	
	485	Byte 0	RANGE 2 channel 3	(MSB)
	486			
	487			
		Byte 1	RANGE 1 channel 3 Division	(LSB)
	488	Byte 0	RANGE 1 channel 3 Division	(MSB)
		Byte 1	RANGE 2 channel 3 Division	(LSB)
	489	Byte 0	RANGE 2 channel 3 Division	(MSB)
	490			
		Byte 0	Channel 3 Decimals	
	491	Byte 0	Channel 3 Unit of Measure; See Note 1	
Area Setup: PAGE 31 (16 bytes)	548			
	549			
	550			
	551	Byte 3	RANGE 1 channel 4	(LSB)
		Byte 2	RANGE 1 channel 4	
	552	Byte 1	RANGE 1 channel 4	
		Byte 0	RANGE 1 channel 4	(MSB)
	553	Byte 3	RANGE 2 channel 4	(LSB)
		Byte 2	RANGE 2 channel 4	
	554	Byte 1	RANGE 2 channel 4	
		Byte 0	RANGE 2 channel 4	(MSB)
	555			
Area Setup: PAGE 32 (16 bytes)	556			
	557	Byte 1	RANGE 1 channel 4 Division	(LSB)
		Byte 0	RANGE 1 channel 4 Division	(MSB)
	558	Byte 1	RANGE 2 channel 4 Division	(LSB)
		Byte 0	RANGE 2 channel 4 Division	(MSB)
	559			
	560	Byte 0	Channel 4 Decimals	
		Byte 0	Channel 4 Unit of Measure; See Note 1	
	561			
	562			
	563			

Note *Table Notes*

1. *Significance of the numeric value in the Unit of Measure field:*

   0 Grams *2 Tons*
   1 Kilograms *3 Pounds*

5.3 Coils Status data area

The *Coils status* data area is written by the primary device (is therefore read by the instrument) and is made up of coils of 1 bit.

N°Coil.	Coils Status	Significant bit	
		1	0
00001 (0)	Digital output 1	Not active	Active
00002 (1)	Digital output 2	Not active	Active

Note *Writing allowed if the related output has no associated function.*

6.0 Calibration

Calibration holding registers:

Register	Data
40901	number of calibration points
40902	1st calibration point weight (high)
40903	1st calibration point weight (low)
40904	2nd calibration point weight (high)
40905	2nd calibration point weight (low)
40906	3rd calibration point weight (high)
40907	3rd calibration point weight (low)
40908	zero calibration ADC value (high)
40909	zero calibration ADC value (low)
40910	1st calibration point ADC value (high)
40911	1st calibration point ADC value (low)
40912	2nd calibration point ADC value (high)
40913	2nd calibration point ADC value (low)
40914	3rd calibration point ADC value (high)
40915	3rd calibration point ADC value (low)

Register	Data
40951	unit of measure (g, kg, t, lb)
40952	1st range division
40953	2nd range division
40954	decimals
40955	1st range capacity (high)
40956	1st range capacity (low)
40957	2nd range capacity (high)
40958	2nd range capacity (low)

Input register: 30116 calibration status. Values:

Value	Description
0	MODBUS_CALIBRATION_NOT_STARTED
1	MODBUS_CALIBRATION_ACQUISTION_UNDERWAY
2	MODBUS_CALIBRATION__ACQUISTION_OK
3	MODBUS_CALIBRATION__ACQUISTION_ERROR
4	MODBUS_CALIBRATION_OK
5	MODBUS_CALIBRATION_ERROR

Specific Commands

Number	Command	Notes
35 (23H)	READ_CALIBRATION	Copy of the calibration data of the channel is equal to parameter 1 into temporary area
36 (24H	WRITE_CALIBRATION	Store of temporary data into calibration data (non volatile memory)
37 (25H)	POINT_ACQUISITION	Parameter 1 is the point to acquire
38 (26H)	ABORT_CALIBRATION	Abort the calibration under way

6.1 Calibration sequence:

1. Use command *READ CALIBRATION* with parameter 1 equal to the channel to calibrate (1st channel is zero). If type is equal to dependent channels parameter 1 can be equal to zero only.
2. Set metrologic values in the page 5000, if needed.
3. Set calibration points in the page 5001, bytes 17-18.
4. Set calibration weight(s) in the page 5001, bytes 19 to 30.
5. If a theoretical calibration is to be executed write directly the ADC values in the page 5002.

   If not switch to page 5001 to read the calibration status register (bytes 31-32), then unload the platform and use the command *POINT_ACQUISITION* with parameter equal to 0.
6. Wait for calibration status to equal *CALIBRATION_ACQUISTION_OK*.

   If *CALIBRATION_ACQUISTION_ERROR* displays, return to Step 5.
7. Load the platform with 1st calibration weight and use command *POINT_ACQUISITION* with parameter equal to 1.
8. Wait for calibration status to equal *CALIBRATION_ACQUISTION_OK.*

   If *CALIBRATION_ACQUISTION_ERROR* displays, return to Step .
9. Repeat Step for other calibration points (if any).
10. Use command *WRITE_CALIBRATION* with parameter 1 equal to zero to store the new calibration.
11. Wait for calibration status is equal to *CALIBRATION_OK*.

    If *CALIBRATION_ERROR* displays, return to Step 1.

While the command WRITE_CALIBRATION is in progress some Modbus reading timeout may happen because of the saving procedure.

RICE LAKE®
WEIGHING SYSTEMS

230 W. Coleman St. • Rice Lake, WI 54868 • USA
U.S. 800-472-6703 • Canada/Mexico 800-321-6703 • International 715-234-9171 • Europe +31 (0)26 472 1319