WE503D Modbus RTU

Dynamic Integrator for Belt Scales, Screw Weighers, Bulkslide Flowmeters and Impact Flowmeters

Program Version: From v3.06

Communication Protocol Manual

RICE LAKE®
WEIGHING SYSTEMS

November 13, 2020

PN 203226 Rev A

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

This manual provides Modbus RTU communication protocol information for the WE503D Dynamic Integrator.

Ensure the WE503D Dynamic Integrator is fully installed by following the instructions of the WE503D Dynamic Integrator Installation Manual (PN 200479).

Where Modbus RTU is indicated, it also fulfills Modbus TCP. See Section 10.0 on page 15 to set the Modbus TCP parameters.

Manuals and additional resources are available from the Rice Lake Weighing Systems website at www.ricelake.com

Warranty information can be found on the website at www.ricelake.com/warranties

1.1 Additional Resources

For additional resources, see the following information:

WE503D Dynamic Integrator Installation Manual
The WE503D Dynamic Integrator Installation Manual (PN 200479) provides an overview of the installation instructions for the WE503D Dynamic Integrator.

WE503D Fieldbus Communication Protocol Manual
The WE503D Fieldbus Communication Protocol Manual (PN 203227) provides an overview of the Fieldbus communication protocol.

1.2 Safety

Safety Signal Definitions:

DANGER — ***Indicates an imminently hazardous situation that, if not avoided, will result in death or serious injury. Includes hazards that are exposed when guards are removed.***

WARNING — ***Indicates a potentially hazardous situation that, if not avoided, could result in serious injury or death. Includes hazards that are exposed when guards are removed.***

CAUTION — ***Indicates a potentially hazardous situation that, if not avoided, could result in minor or moderate injury.***

IMPORTANT — ***Indicates information about procedures that, if not observed, could result in damage to equipment or corruption to and loss of data.***

General Safety

Do not operate or work on this equipment unless this manual has been read and all instructions are understood. Failure to follow the instructions or heed the warnings could result in injury or death. Contact any Rice Lake Weighing Systems dealer for replacement manuals.

WARNING

Failure to heed could result in serious injury or death.

1.3 General

The Modbus communication protocol defines the structure of the messages and the communication mode between a master device which manages the system and one or more slave devices which respond to the interrogations of the master (master-slave technique).
This protocol defines how the transmitter and receiver are identified, the data switch mode, the communication launch and interruption modes and the error detection mode.
Only the master can start a transaction (***Query***) while the other devices (the slaves) respond by supplying the data requested to the master or carrying out the actions requested in the query. The master can either address single slaves or transmit a broadcast message to all. The slaves respond with a message (***Response***) to the queries which are individually addressed, but do not transmit any answer to the master if there are broadcast messages.
A transaction can therefore have the following structures:

- Single question to a slave / single answer from the slave
- Single broadcast message to all the slaves / no answer from the slaves

2.0 Symbols

In the manual:

msb= most significant bit

MSB= most significant byte

lsb= least significant bit

LSB= least significant byte

3.0 Selection of the Modbus Serial Communication Mode

Enter the technical setup environment to select the Fieldbus communication protocol.

To enter the setup environment, turn off the instrument by pressing the **CLR** key for 3-seconds, the instrument will startup again, when the firmware version is displayed press the **TARE** key once and again when ***USER – PRESS KEY*** is displayed. Password: 41042

- Select ***Serial Printouts, ETC*** by using the **F6** and **F7** key, press **Enter**.
- Select ***Serial Ports Configuration*** and press **Enter**.
- Select a selection with COM3: PC and press **Enter**.
- Select ***PC Serial Configuration*** and press **Enter**.
- Select ***Address 485*** and press **Enter**.
- Enter the slave address and press **Enter**.
- Select ***Selection Protocol*** and press **Enter**.
- Select ***Modbus RTU*** and press **Enter**.
- Set the desired serial parameters.
- Press ***CLR*** until the display show ***Exiting Setup: Save?*** and press **Enter** to save the settings.

4.0 RTU (Binary) Modbus Transmission Mode

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

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

5.0 Component Description and Message Format

In the RTU transmission modes, 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 master as well as the slave, includes:

- The address of the device with which the master has established the transaction (the address 0 corresponds to a broadcast message transmitted to all the slave devices)
- The function code which defines the requested action
- The data which must be transmitted
- The error check made up according to the CRC algorithm

These fields are described in detail in the following paragraphs:

Query

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

Response

- If the slave gives a normal answer:
  The function code is the echo of the query function code; The data bytes contain the data retrieved from the slave, like the registers' values or the states
- If a slave locates an error (format, parity, or in the CRC) or it is unable to execute the requested action:
  The master 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

5.1 Frame Format in RTU Mode

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, see Table 5-1). 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 times the time period of a character, if 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.

A typical message frame is shown in the following table:

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 5-1. Typical Message Frame

5.2 Device Address

As mentioned above, the Modbus transactions always involve the master, which manages the line, and a slave at a time (except for the broadcast messages). In order to identify the message consignee, the numeric address of the selected slave device (1-byte: 8-bits for the RTU) is transmitted as the first field of the frame. Each slave will therefore be assigned a different address number so that it can be identified. When the slave transmits its answer, its address is entered in the response's field address in order that the master knows which slave is responding.

Valid addresses for the slave devices are within a range from 0 to 247, in particular:

0 ⇨ broadcast address (all the slave devices)

1 ⇨ minimum possible address for the slave

247 ⇨ maximum possible address for the slave

5.3 Function Code

The field of the frame function code of a message contains 8-bits (RTU). Valid codes are within the range from 1 to 255 decimals. When a message is transmitted from a master to a slave device the function code field indicates to the slave what kind of action should be executed (for example the reading of the ***Input Registers***, etc.).

When a slave responds to the master, 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 slave 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 slave enters a single code within the data field of the response message, in order to tell the master which type of error has taken place or the reason for the exception.

5.4 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 RTU character, in accordance with the network's serial transmission mode.

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

5.5 Error Check

The contents of the error check field depend on the Modbus transmission mode because there are two distinct error check methods. In particular:

- The communication strings are checked by a CRC (Cyclical Redundancy Check) algorithm (Section 7.0 on page 11)
- 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

5.6 Example of the Message Components in 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 RTU mode.

Query

Read Input Registers to the ***01 Slave Address***, for reading of the contents of three registers starting from register n°8

Field Name	Example (Hex)	RTU: 8-bit Field
Heading		None
Slave Address	01	0000 0001
Function	04	0000 0100
Start Address (High)	00	0000 0000
Start Address (Low)	08	0000 1000
Number of Registers (High)	00	0000 0000
Number of Registers (Low)	03	0000 0011
Error Check		CRC (16-bits)
Terminator		None
Nr. of total bytes		**8**

Table 5-2. Example Query Message Components

Response

Field Name	Example (Hex)	RTU: 8-bit Field
Heading		None
Slave Address	01	0000 0001
Function	04	0000 0100
Number of Bytes	06	0000 0110
Data (HIGH)	02	0000 0010
Data (LOW)	2B	0010 1011
Data (HIGH)	00	0000 0000
Data (LOW)	00	0000 0000
Data (HIGH)	00	0000 0000
Data (LOW)	63	0110 0011
Error Check		CRC (16-bits)
Terminator		None
Nr. of total bytes		**11**

Table 5-3. Example Response Message Components

In the ***Response*** of the slave there is the ***Function Echo*** indicating that it's a normal type of answer (Table 5-3).

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 RTU.

Example: The 63 Hex value is transmitted as a 8-bit byte (01100011).

6.0 Modbus Functions

Each function is exposed in detail in the following pages and is made up of a ***QUERY*** (master request → instrument) and a ***RESPONSE*** (instrument response → master).

Note ***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.

6.1 List of the Supported Functions

In the following table there are the active Modbus functions for the instrument.

Function Code	Description
04 (0x04)	READ INPUT REGISTERS
06 (0x06)	PRESET SINGLE REGISTER
16 (0x10)	PRESET MULTIPLE REGISTERS

Table 6-1. Supported Functions

In the parentheses there are the hexadecimal values.

6.2 List of the Transmission Strings

In the following paragraphs the functions (shown in Table 5-2 on page 7 and Table 5-3 on page 7) supported by the instrument are described in detail; for this purpose one uses the following classification for the message fields:

- **Address:** 1-byte for the instrument address (slave)
- **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):** For the error check, in the RTU and in the ASCII transmission modes it is always 2-bytes See Section 7.0 on page 11 for further details

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

Note ***The following registers are defined, on which the functions operate:***

- N°16 Input Registers (***Input Registers***): written by the instrument → read by the master
- N°16 Output Registers (***Holding Registers***): written by the master → read by the instrument

The input and output registers are fully described in Section 9.0 on page 14.

6.3 Function 1: Read Input Registers (04 Hex)

Function 1: Read Input Registers reads the contents of the slave instrument's input registers (or ***Input Registers***, which the instrument will write); the broadcast communication is not supported.

Query

Initial Register (***1st Register Address***) is specified from which reading starts and ***Number of Registers*** (***Nr. of Registers***) which must be read. 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	04	00	08	00	01	CRC

Table 6-2. Function 1 Query Example

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	04	02		00	0A	CRC

Table 6-3. Function 1 Response Example

Example: A = 01;

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

- In the Response: 1st Register = 00 0A

6.4 Function 2: Preset Single Register (06 Hex)

It allows to set a single output register (***Holding Register***, which the instrument or slave goes to read) to a determined value.

The broadcast transmission of this command is allowed and in which one can set the same register in all the connected slaves.

Query

One specifies the ***Register Address*** which must be set (***Register Address***) and the relative balue (***Register Value***).

The registers are addressed starting 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	06	00	01	00	03	CRC

Table 6-4. Function 2 Query Example

Response

It is the echo of the ***Query***.

Address	Function	Address 1st Register		Nr. of Registers		Error Check
		High	Low	High	Low	
A	06	00	01	00	03	CRC

Table 6-5. Function 2 Response Example

Example: A = 01;

- in the Query: Register Address = 00 01; Register Value = 00 03

- in the Response: Register Address = 00 01; Register Value = 00 03

6.5 Function 3: Preset Multiple Registers (16 Hex)

Allows to set various output registers (Holding Register, which the instrument or slave goes to read) to a determined value.

Query

Here is specified the address of the First output Register which must be set (***1st Register address***), the Number of Registers to be written (***Nr. of Registers***) starting from 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		Nr. of Registers		Nr. of Bytes	1st Register Value		2nd Register Value		Error Check
		High	Low	High	Low		High	Low	High	Low	
A	10	00	00	00	03	10	00	03	00	03	CRC

Table 6-6. Function 3 Query Example

Response

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

Address	Function	1st Register Address		Nr. of Registers		Error Check
		High	Low	High	Low	
A	10	00	00	00	02	CRC

Table 6-7. Function 3 Response Example

Example: A = 01;

- In the 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;

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

7.0 Error Check Methods

The Modbus serial communication uses two error check types:

Check on the character or parity (even or uneven), can be applied optionally to each character.

Check on the frame (CRC algorithm), applied to the entire message.

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

7.1 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.

7.2 CRC Algorithm: Cyclical Redundancy Check (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.

7.3 Procedure for Creating the CRC is the Following

1. Loading a 16-bit register with FFFF Hex (all 1). This register is called ***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 of 1-bit to the right (towards 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 (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. Step 3 and Step 4. are repeated until eight shifts have been made. After this a byte of 8-bits have been completely processed.
6. Step 2 to Step 5 are repeated for the next byte of 8-bits of the message. One continues until all 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 (Section 7.4).

7.4 Placing of the CRC in the 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	CRC High
							41	12

Table 7-1. Sequence of the CRC Bytes

7.5 Example in C Language in Generating the CRC

A functioning example for the creation of the CRC in the C language is shown below.

The function creates internally the swapping of high and low bytes of the CRC. 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 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(unsigned char *puchMsg,
                         unsigned short usDataLen
                         )

    {
        unsigned short CRC;
        int i, n;
        unsigned short usPolynomial = 0xA001;
        unsigned short usInitialReminder = 0xFFFF;
        CRC = usInitialReminder;//initialisation CRC
        for (i = 0; i < usDataLen; i++)//for each byte of the message it executes the division module-2 for the
        polynomial
        {
            CRC = CRC ^ puchMsg[i];//XOR byte low CRC with byte message under exam
            for (n = 0; n < 8; n++)// division module-2 at bit
            {
                if (CRC & 0x0001) //least significant bit CRC 1
                {
                    CRC = CRC >> 1; //shift to the right of the CRC
                    CRC = CRC ^ usPolynomial; //XOR CRC with polynomial
                }
                else//bit least significant CRC 0
                CRC = CRC >> 1; //shift to the right of the CRC
            }
        }
    CRC = (CRC << 8) | (CRC >> 8); //switch of least significant and most significant byte
    return CRC;
}
```

Figure 7-1. CRC Generation Function Example

8.0 Modbus Exceptions

In a ***Normal Response*** the slave 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*** the slave 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.

8.1 List of the Detected Exceptions

Active Modbus 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 slave 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 slave device or is not valid
06	Slave Device Busy	The slave is busy in processing a command which requires a lot of time; The master can transmit again the message later, when the slave is free

Table 8-1. Active Modbus Exceptions

9.0 Notes

There are two data areas, input data and output data area. The input area is read by the master and the output area is written by the master. Both areas organized in registers. Each register consists of 16-words (32-bytes). See Section 10.0 on page 15 for input data information. The input register consists of multiple pages. It's possible to change the displayed page by writing a command in the output data area.

All data are expressed in big endian order (first byte is most significant).

Wx: Word rank in a multi-word data.

Example: (W2), (W1), (W0): W2 is the most significant word and W0 is the least significant word.

10.0 Input Register

Word Addr.	Byte Nr.	Word Contents	Word Type	Bit Addr.	Contents
IW0	0	Number of processed commands	Integer number	0.0 ... 0.3	Number of processed commands
				0.4 ... 0.7	Result of last command: 0= Ok 1= Not allowed command 2= Wrong command data 3= Unknown command
	1	Displayed page	Integer number		
IW1	1	Belt status	Integer number	0.0 ... 0.3	Operation mode: 0= Reader (PID disabled) 1= Regulator (PID enabled)
				0.4 ... 0.7	Belt status: 0= Error (Alarm) 1= Manual functioning 2= Wait for start 3= Pause 5= Running (Dosage) 6= Start phase 7= Stop (Dosage end) 8= Locked 9= Zero calibration active
	2	Error code	Integer number		0= No error 1= Off track 2= Underflow (alarm) 3= Overflow (alarm) 4= Zero flow rate 5= Weight (alarm) 7= Underflow (locked) 8= Overflow (locked) 9= Total dosed weight (locked) 10= Dosage disabled by input 11= Weight (locked) 12= External alarm 13= External alarm lock 14= Flow rate (locked)
IW2		Belt load (W1)	Integer number		Kg/m or lb/ft with three decimals
IW3		Belt load (W0)	Integer number		Kg/m or lb/ft with three decimals
IW4		Flow rate	Integer number		
IW5		PID value %	Integer number		With two decimals
IW6		Analog output %	Integer number		With two decimals
IW7		Partial total (W2)	Integer number		
IW8		Partial total (W1)	Integer number		
IW9		Partial total (W0)	Integer number		
IW10		General total (W2)	Integer number		
IW11		General total (W1)	Integer number		
IW12		General total (W0)	Integer number		
IW13		Input status	Discr. Bits	13.0	Load polarity: 0= +, 1=--
				13.1	Weight stability: 0= Unstable, 1= Stable
				13.2	Underload: 0= No, 1= Yes
				13.3	Overload: 0= No, 1= Yes
				13.4	Gross zero zone: 0= Out, 1= In zone
				13.5 ... 13.6	Total units of measure: 1=kg, 2= t, 3= lb
				13.7	Flow rate polarity: 0= +, 1=--
				13.8 ... 13.9	Flow rate units of measure: 1= kg/h, 2= t/h, 3= lb/h
				13.10	Status if input 1: 0= Disabled, 1= Enabled
				13.11	Status if input 2: 0= Belt speed = 0, 1= Belt speed > 0
				13.12	Status if input 3: 0= Disabled, 1= Enabled
				13.13	Status if input 4: 0= Disabled, 1= Enabled
				13.14	Status if input 5: 0= Disabled, 1= Enabled
				13.15	Status if input 6: 0= Disabled, 1= Enabled
IW14		Output status	Discr. Bits	14.0 ... 14.15	Status of output: 0= Disabled, 1= Enabled
IW15		Other data	Integer number	15.0 ... 15.3	Weight decimals
			Integer number	15.4 ... 15.5	Flow rate decimals
			Discr. Bits	15.6	Status if input 7: 0= Disabled, 1= Enabled
			Discr. Bits	15.7	Status if input 8: 0= Disabled, 1= Enabled
			Integer number	15.8 ... 15.15	Number of dosages (batch)

Table 10-1. Page 0

Word Addr.	Byte Nr.	Word Contents	Word Type	Bit Addr.	Contents
IW0	0	Number of processed commands	Integer number	0.0 ... 0.3	Number of processed commands
				0.4 ... 0.7	Result of last command: 0= Ok 1= Not allowed command 2= Wrong command data 3= Unknown command
	1	Displayed page	Integer number		
IW1		Target flow rate	Integer number		
IW2		Scale capacity (W1)	Integer number		
IW3		Scale capacity (W0)	Integer number		
IW4		Max flow rate	Integer number		
IW5		Min flow rate	Integer number		
IW6		Dosage time (W1)	Integer number		1/100 sec
IW7		Dosage time (W0)	Integer number		1/100 sec
IW8		Target batch weight (W2)	Integer number		
IW9		Target batch weight (W1)	Integer number		
IW10		Target batch weight (W0)	Integer number		
IW11		Digital output function	Integer number	11.0 ... 11.7	Digital output function: 0= None 1= Run / batch active 2= Batch finished 3= Pause 4= Off track 5= > Upper flow 6= < Lower flow 7= Alarm 8= Lock 9= Totalizer pulse 10= Material enable 11= Slow flow 12= Belt speed > 0 13= Zero calibration active 14= Flow in dead band 15= Test weight 16= Air purge
				11.8 ... 11.15	Digital output index
IW12		Digital output ON value (W1)	Integer number		Used only with digital output function 9, ON and OFF values are set with the same value
IW13		Digital output ON value (W0)	Integer number		Used only with digital output function 9, ON and OFF values are set with the same value
IW14		Digital output OFF value (W1)	Integer number		Used only with digital output function 9, ON and OFF values are set with the same value
IW15		Digital output OFF value (W0)	Integer number		Used only with digital output function 9, ON and OFF values are set with the same value

Table 10-2. Page 1

Word Addr.	Byte Nr.	Word Contents	Word Type	Bit Addr.	Contents
IW0	0	Number of processed commands	Integer number	0.0 ... 0.3	Number of processed commands
				0.4 ... 0.7	Result of last command: 0= Ok 1= Not allowed command 2= Wrong command data 3= Unknown command
	1	Displayed page	Integer number		
IW1		PID P value	Integer number		With two decimals
IW2		PID I value	Integer number		With two decimals
IW3		PID D value	Integer number		With two decimals
IW4		PID interval time	Integer number		In seconds with one decimal

Table 10-3. Page 2

Word Addr.	Byte Nr.	Word Contents	Word Type	Bit Addr.	Contents
IW0	0	Number of processed commands	Integer number	0.0 ... 0.3	Number of processed commands
				0.4 ... 0.7	Result of last command: 0= Ok 1= Not allowed command 2= Wrong command data 3= Unknown command
	1	Displayed page	Integer number		
IW1	1	Belt status	Integer number	0.0 ... 0.3	Operation mode: 0= Reader (PID disabled) 1= Regulator (PID enabled)
				0.4 ... 0.7	Belt status: 0= Error (Alarm) 1= Manual functioning 2= Wait for start 3= Pause 5= Running (Dosage) 6= Start phase 7= Stop (Dosage end) 8= Locked 9= Zero calibration active
	2	Error code	Integer number		0= No error 1= Off track 2= Underflow (alarm) 3= Overflow (alarm) 4= Zero flow rate 5= Weight (alarm) 7= Underflow (locked) 8= Overflow (locked) 9= Total dosed weight (locked) 10= Dosage disabled by input 11= Weight (locked) 12= External alarm 13= External alarm lock 14= Flow rate (locked)
IW2		Belt load (W1)	Integer number		Kg/m or lb/ft with three decimals
IW3		Belt load (W0)	Integer number		Kg/m or lb/ft with three decimals
IW4		Flow rate	Integer number		
IW5		PID value %	Integer number		With two decimals
IW6		Analog output %	Integer number		With two decimals
IW7		Partial total (W1)	Integer number		
IW8		Partial total (W0)	Integer number		
IW9		General total (W1)	Integer number		
IW10		General total (W0)	Integer number		
IW11		Speed	Integer number		In m/s or ft/min with two decimals
IW12		Inclination	Integer number		In ° with one decimal
IW13		Input status	Discr. Bits	13.0	Load polarity: 0= +, 1=--
				13.1	Weight stability: 0= Unstable, 1= Stable
				13.2	Underload: 0= No, 1= Yes
				13.3	Overload: 0= No, 1= Yes
				13.4	Gross zero zone: 0= Out, 1= In zone
				13.5 ... 13.6	Total units of measure: 1=kg, 2= t, 3= lb
				13.7	Flow rate polarity: 0= +, 1=--
				13.8 ... 13.9	Flow rate units of measure: 1= kg/h, 2= t/h, 3= lb/h
				13.10	Status of input 1: 0= Disabled, 1= Enabled
				13.11	Status of input 2: 0= Belt speed = 0, 1= Belt speed > 0
				13.12	Status of input 3: 0= Disabled, 1= Enabled
				13.13	Status of input 4: 0= Disabled, 1= Enabled
				13.14	Status of input 5: 0= Disabled, 1= Enabled
				13.15	Status of input 6: 0= Disabled, 1= Enabled
IW14		Output status	Discr. Bits	14.0 ... 14.15	Status of output: 0= Disabled, 1= Enabled
IW15		Other data	Integer number	15.0 ... 15.3	Weight decimals
			Integer number	15.4 ... 15.5	Flow rate decimals
			Discr. Bits	15.6	Status of input 7: 0= Disabled, 1= Enabled
			Discr. Bits	15.7	Status of input 8: 0= Disabled, 1= Enabled
			Integer number	15.8 ... 15.15	Number of dosages (batch)

Table 10-4. Page 3

Word Addr.	Byte Nr.	Word Contents	Word Type	Bit Addr.	Contents
IW0	0	Number of processed commands	Integer number	0.0 ... 0.3	Number of processed commands
				0.4 ... 0.7	Result of last command: 0= Ok 1= Not allowed command 2= Wrong command data 3= Unknown command
	1	Displayed page	Integer number		
IW1		Selected article	Integer number		(65535 or FFFF hex if no article is selected)
IW2		PID start value %	Integer number		With two decimals
IW3		Target Flow rate	Integer number		
IW4		Target batch weight (W1)	Integer number		
IW5		Target batch weight (W0)	Integer number		
IW6		Correction factor (W1)	Integer number		
IW7		Correction factor (W0)	Integer number		With six decimals
IW8		Total batch weight (W1)	Integer number		
IW9		Total batch weight (W0)	Integer number		
IW10		Total number of batches (W1)	Integer number		
IW11		Total number of batches (W0)	Integer number		

Table 10-5. Page 4

11.0 Output Register

Word Addr.	Byte Nr.	Word Contents	Word Type	Bit Addr.	Contents
OW0	0	Command	Integer number		Command value: 0= None 1= Not used 2= Start 3= Pause 4= Stop 5= Set flow rate / % remote flow-rate Set value in OW1 6= Set batch target weight Set value in OW1 (W2), OW2 (W1), OW3 (W0) 7= Set batch dosage time Set value in OW1 (W1), OW2 (W0) 8= Print format Set value in OW1 9= Reset general total 10= Set digital output Set value in OW1 (bitmap of the outputs) 11= Store setup 12= Change digital output index (page 1) Set value in OW1 13= Set digital output value Set index in OW1, ON value in OW2 (W1) and OW3 (W0), OFF value in OW4 (W1) and OW5 (W0) 14= Change page Set value in OW1 15= Zero calibration 16= Set PID P parameter Set value in OW1 17= Set PID I parameter Set value in OW1 18= Set PID D parameter Set value in OW1 19= Set PID interval time Set value in OW1 20= Set digital input Set value in OW1 (bitmap of the inputs) 21= Select article Set value in OW1 (value 9999 is unselected)
	1	0 or 1	Integer number		Write 0 with Modbus
OW1		Parameter	Integer number		
OW2		Parameter	Integer number		
OW3		Parameter	Integer number		
OW4		Parameter	Integer number		
OW5		Parameter	Integer number		

Table 11-1. Output Register

RICE LAKE®
WEIGHING SYSTEMS

Rice Lake Weighing Systems Europe B.V. • Weiland 11 • NL-6666 MH Heteren • The Netherlands
Europe +31 (0)26 472 1319 • U.S. 800-472-6703 • Canada/Mexico 800-321-6703 • International 715-234-9171