MODBUS

Communication Protocol

for

DGT4 /DGTQ

and

DGT4 AN/DGTQ AN

WEIGH INDICATORS

DGT-DGTQ_APPENDIX_MODBUS_06.00_15.11_UK

INDEX

1. GENERALITIES

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). It defines how the master and the slaves 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 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

1.1. Selection of the MODBUS serial communication mode

To select the Modbus communication protocol mode one should enter in the *SET-UP ENVIRONMENT* of the instrument (see Figures 1 and 2):

Input in the Set-up Environment

- Turn on the indicator, press the ZERO or the TARE key during the countdown (the display shows the "tYPE" menu).
- Select "SEtuP" (with the ZERO or TARE keys) ⇨ press PRINT to confirm the step.
- Select "SEr 1AL" (with the ZERO or TARE keys) ⇨ press PRINT to confirm the step.
- Select "CoM_PC" (with the ZERO or TARE keys) ⇨ press PRINT to enter in the:

Menu for Setting the Communication parameters of the PC port:

➢ The "PCModE" item appears ⇨ press PRINT to enter in this submenu and select the "Modbus" item ⇨ press PRINT again to confirm.

Now one should select the type of protocol: Ascii or Binary (RTU) (see Section 1.2) and the setting of the instrument's serial address (see Section 1.4.3).

ASCII or RTU

➢ The first item which appears is "Mod_tYP" ⇨ press PRINT to enter in the submenu:
choose either the ASCII or the RTU communication ⇨ confirm the choice with PRINT.

Instrument serial address

➢ The second item which appears is "Mod_Add" ⇨ press PRINT ⇨ for a few instants "Ad485" is displayed ⇨ now type in the serial address of the instrument (or slave) ⇨ confirm the entered value with PRINT.

Baud Rate – Data Format

➢ Now set the other communication parameters of the serial port, in other words, the Baud Rate and the Serial Word Format, in the "bAud" and "b 1t" steps, respectively *(see Section 1.3)*.

- Press the C key various times until the display shows the message "SAVEP".
- Press PRINT to confirm the changes made or another key to not save.

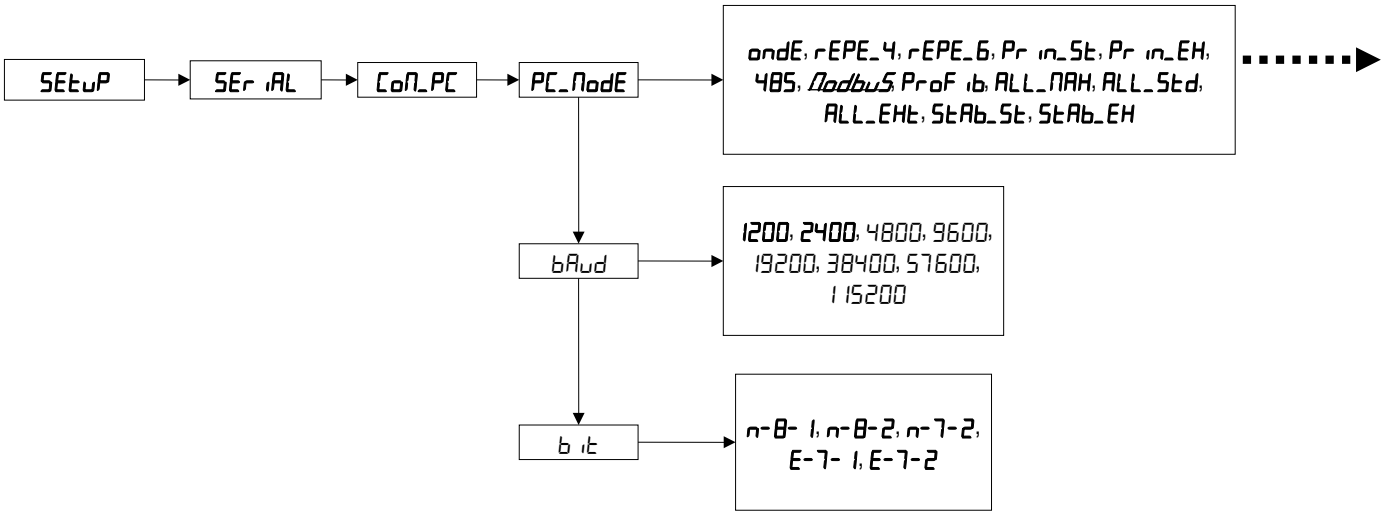

Figure 1*:* Selection from SET-UP ENVIRONMENT of the MODBUS communication, setting of the Baud Rate and the serial word format.

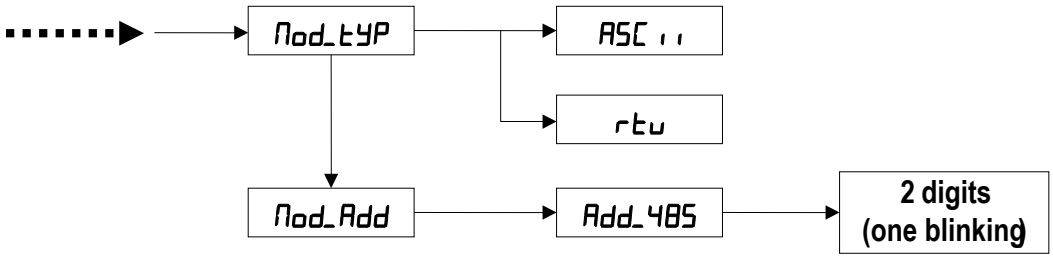

Figure 2*:* Selection from SET-UP ENVIRONMENT of the MODBUS communication (Ascii or Rtu) and setting of the serial address of the instrument (or slave).

1.2. Modbus transmission modes: ASCII or RTU (binary)

In the selection from the SET-UP ENVIRONMENT of the requested Modbus protocol one is also asked to choose the serial transmission mode which may either be ASCII or RTU (see Figure 2). This choice determines how the information is packaged inside the message fields and how it is decoded.

- **ASCII mode** (American Standard Code for Information Interchange):
  Each byte (8 bit) in a message is transmitted as 2 ASCII characters.
  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.

- **RTU mode** (Remote Terminal Unit):
  Each byte (8 bit) in a message has 2 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.

1.3. Setting Parameters of the serial transmission:Baud Rate and Data Format

In the SET-UP ENVIRONMENT, after having selected the type of serial transmission (ASCII or RTU) for the Modbus protocol, one is asked also to choose the communication parameters of the serial port, in other words, the Baud Rate and the Data Format (see Figure 1).

- **Baud Rate** (or transmission speed):
  SEtuP ⇨ SEr iAL ⇨ CoM_PC ⇨bAud⇨ 1200 ... 115200

- **Data Format** (or serial word format):
  SEtuP ⇨ SEr iAL ⇨ CoM_PC ⇨ bAud ⇨ n-8-1, n-8-2, n-7-2, E-7-1, E-7-2
  (n-8-1) (n-8-2) (n-7-2) (e-7-1) (e-7-2)

  NOTE: the advisable formats are:
  - **In ASCII mode:** n-7-2, E-7-1
  - **In RTU mode:** n-8-2, E-8-1

  In which:
  n ⇨ no parity (none)
  E ⇨ even parity (Even)

  Example*:* if one uses n-8-2 the data format will be:
  8 Data Bits
  n = No parity
  2 Stop Bit

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.

1.4. Description of the Components 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 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 or LRC algorithm.

These fields are described in detail in the following paragraphs. For the Query and Response there is:
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, in the CRC on in the LRC) 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.

1.4.1. Frame Format in ASCII mode

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 1 second; if there is a longer interval, the receiving device assumes that an error has taken place.

A typical message frame is shown in the following figure:

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

Figure 3*:* ASCII message frame*.*

1.4.2. 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 Figure 4). 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.

A typical message frame is shown in the following figure:

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

Figure 4*:* Frame of the RTU message*.*

1.4.3. The 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 (one byte: two characters for the ASCII, eight 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

1.4.4. The 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 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.

1.4.5. The 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 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.

1.4.6. The 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:

- **In 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.

- **In 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.

1.4.7. Example of the message components in ASCII and 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 organised (framed) in ASCII or RTU mode.

Query: "Read Input Registers" to the 01 Slave 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
Slave 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
Slave 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

In the *Response* of the Slave 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).
For *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. 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:

- *In the ASCII transmission mode*:
  Each character is an ASCII type character (made up of 8 bits).
- *In the RTU transmission mode*:
  Each character is a 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. List of the supported Functions

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

Table 2.1: Active Modbus functions

Function Code	Description
01 (0x01)	READ COILS STATUS
03 (0x03)	READ HOLDING REGISTERS
04 (0x04)	READ INPUT REGISTERS
05 (0x02)	PRESET SINGLE COIL
06 (0x06)	PRESET SINGLE REGISTER
15 (0x0F)	PRESET MULTIPLE COILS
16 (0x10)	PRESET MULTIPLE REGISTERS

In the parentheses there are the hexadecimal values.

2.2. List of the 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 (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 / 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 Chapter 3)

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 Master → read by the Instrument and the Master
  - "Holding Registers": written by the Master → read by the Instrument and the Master
  - "Input Registers": written by the Instrument → read by the Master
- 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. Functions 1,3 and 4: READ COILS STATUS / HOLDING / INPUT REGISTERS (01,03 and 04 Hex)

They read the contents of the slave instrument's registers (which the instrument or the master will write); 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	Nr. of read bytes		Value of registers		Error Check
		High	Low	High	Low	
A	**XX**	02		*00*	*0A*	CRC / LRC

Example: A = *01*;
- *in the Query*: 1st Register address = *00 08*; Number of Registers = *00 01*
- *in the Response*: 1st Register = *00 0A*

NOTE:
Maximum number of registers readable with one request: 49

2.2.2. Functions 5 and 6: PRESET COIL / SINGLE REGISTER (05 and 06 Hex)

It allows to set a single 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 specify the register / coil data area to write (*Function*), the Register Address which must be set (*Register Address*) and the relative Value (*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		Register Value		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		Register Value		Error Check
		High	Low	High	Low	
A	**XX**	*00*	*01*	*00*	*03*	CRC / LRC

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*

2.2.3. Function 15 and 16: PRESET MULTIPLE COILS / REGISTERS (0F and 16 Hex)

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

Query:
One specify the registers / coils data area to write (*Function*), Here is 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 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 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*	*02*	CRC / LRC

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;*

NOTE:
Maximum number of registers writable with one request:
- RTU mode: 45
- ASCII mode: 20

2.3 Example of function

For that examples we'll have a DGT-Q; capacity = 3000 kg; division = 1 kg; gross = 1000 kg; tare = 1500 kg; net = -500 kg.

- **Read net weight:**
  Query:

Slave address	Active function	Address 1st register	Number of registers	Error check
0x01	0x04	0x0002	0x0002	0xD00B

Complete string: "**00000001000001000000000000000100000000000000010110100000001011**$_2$";
"**010400020002D00B**$_{16}$";

Response:

Slave Address	Active function	Nr. bytes	Value 1st register	Value 2nd register	Error check
0x01	0x04	0x04	0x0000	0x01F4	0xFB93

Value 1st register Value 2nd register
0000 0000 0000 0000 | 0000 0001 1111 0100$_2$ = 01F4$_{16}$ = 500$_{10}$

- **Read gross weight:**

Query:

Slave address	Active function	Address 1st register	Number of registers	Error check
0x01	0x04	0x0000	0x0002	0x71CB

Complete string: "**00000001000001000000000000000000000000000000010011100011100101 1**$_2$";
"**0104000000271CB**$_{16}$";

Response:

Slave Address	Active function	Nr. bytes	Value 1st register	Value 2nd register	Error check
0x01	0x04	0x04	0x0000	0x03E8	0xFB3A

Value 1st register Value 2nd register
0000 0000 0000 0000 | 0000 0011 1110 1000$_2$ = 03E8$_{16}$ = 1000$_{10}$

- **Read input status register:**

Query:

Slave address	Active function	Address 1st register	Number of registers	Error check
0x01	0x04	0x0004	0x0001	0x700B

Complete string: "**00000001000001000000000000000100000000000000000101110000000010 1 1**$_2$";
"**010400040001700B**$_{16}$"

Response:

Slave Address	Active function	Nr. bytes	Value 1st register	Error check
0x01	0x04	0x02	0x0025	0x78EB

Value 1st register = 0025$_{16}$ = 37$_{10}$ = 0000 0000 0010 0101$_2$

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

(1) : ***high Bit, low Bit:*** *0 0* → Channel 1 *0 1* → Channel 2
(15) (14) *1 0* → Channel 3 *1 1* → Channel 4

- **Execution of Scale Zero:**

Query:

Slave address	Active function	Register address	Register value	Error check
0x01	0x06	0x0000	0x0001	0x480A

Complete string: "**00000001000001100000000000000000000000000000010100100000001010**$_2$";
"**010600000001480A**$_{16}$";

If the command will be executed correctly, the indicator will return the **Query** as **Response**.

- **Execution of automatic Tare:**

Query:

Slave address	Active function	Register address	Register value	Error check
0x01	0x06	0x0000	0x0002	0x080B

Complete string: "**00000001000001100000000000000000000000000000010000100000001011**$_2$";
"**010600000002080B**$_{16}$";

If the command will be executed correctly, the indicator will return the **Query** as **Response**.

- **Execution of manual Tare:**

Query:

Slave address	Active function	First register address	Nr. Reg.	Nr. bytes	1st Reg. value	2nd Reg. value	3rd Reg. value	Error check
0x01	0x10	0x0000	0x0003	0x06	0x0003	0x0000	0x01F4	0xA297

Complete string:
"**00000001000100000000000000000000000000000000000011000001100000000000000011000000000000000000000111110100101000101001011 1**$_2$";
"**011000000003060003000001F4A297**$_{16}$";

2nd Reg. value 3rd Reg. value
0000 0000 0000 0000 | 0000 0001 1111 0100$_2$ = 01F4$_{16}$ = 500$_{10}$

Response:

Slave address	Active function	1st Register address	Nr. Registers	Error check
0x01	0x10	0x0000	0x0003	0x8008

For zero, automatic tare and manual tare command, ref. **Command Register**

- **Set setpoint 1 ON permanent:**
  Query:

Slave address	Active function	First register address	Nr. Registers	Nr. bytes	1st Reg. value	2nd Reg. value	Error check
0x01	0x10	0x0083	0x0002	0x04	0x0000	0x0866	0x03C30

1st Reg. value 2nd Reg. value
0000 0000 0000 0000 | 0000 1000 0110 $0110_2 = 0866_{16} = 2150_{10}$

Complete string:
"**0000000100010000000000001000001100000000000000100000010000000000000000000001000011001100011110000110000**$_2$";
"**011000830002040000086 63C30**$_{16}$";

Response:

Slave address	Active function	1st Register address	Nr. Registers	Error check
0x01	0x10	0x0083	0x0002	0xB020

3. ERROR CHECK METHODS

The Modbus serial communication uses two error check types:

- **Check <u>on the character</u> or parity (even or uneven),** can be applied *optionally* to each character (see Par. 1.3, Data Format).
- **Check <u>on the frame</u> (LRC or CRC algorithms),** applied to the entire message.
  The communication strings are checked by a CRC (Cyclical Redundancy Check) type algorithm in the case of binary (rtu) and the LRC type (Longitudinal Redundancy Check) 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 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.

3.1. Parity Check

It is possible to configure the parity check in the following way (see Par. 1.3):

n ⇨ no parity (none)
E ⇨ even parity (Even)

This determines how the parity is set in each character.

3.2. 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.

3.2.1. A procedure for creating the CRC is the following:

1. Loading a 16-bit register with FFFF Hex (all 1). This register is called *Register CRC*
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 → 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. Steps 3. and 4. are repeated until 8 shifts have been made; after this a byte of 8 bits have been completely processed.
6. Steps 2 to 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. 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.

For 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**

Fig. 5*: Sequence of the CRC bytes.*

3.2.3. Example in C Language of the CRC creation

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

NOTE: *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

`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. LRC algorithm: Longitudinal Redundancy Check (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 *colours*) 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. A procedure for creating the LRC is the following:

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. Placing of the 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.

For example, if the LRC value is 61 Hex (0110 0001):

Colon (**:**)	Addr	Func	Data Count	Data	Data	Data	Data	**LRC HIGH**	**LRC LOW**	CR	LF
								6	**1**		

Fig. 6*: Sequence of the LRC bytes.*

3.3.3. 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. MODBUS EXCEPTIONS

In a normal response (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 (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.

4.1. List of the detected Exceptions

Table 4.1: 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

5. DATA AREAS

There are 3 data areas, "Input", "Holding" and "Coils", defined this way due to the master's point of view: while the "Input" area is read by this device, the "Holding" and "Coils" ones are written.
The first 2 areas are organised 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 master (is therefore written by the instrument) and is made up of registers (input registers), of 2 bytes.

Table 5.1: Modbus Input Registers

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 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

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 5.1.1 section

- Format of the Command Status Register value
See 5.1.3 section

- Format of the Output Status Register value
See 5.1.2 section

30008 (7)	Nr. of last page read or written	(MSB)	
	Nr. of last page read or written	(LSB)	
30009 (8)	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. (sub release)	(byte 1)	
	Software Vers. (bug release)	(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

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)	Analogic output value	(byte 1)	
	Analogic output value	(byte 0)	

- Available 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 (Table 5.1.1)

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

Bit	Description	Bit meaning	
		0	1
(LSB)			
0	Net Weight Polarity	+	--
1	Gross Weight Polarity	+	--
2	Weight Stability	NO	YES
3	Underload 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)**		

(1) : ***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 (Table 5.1.2)

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

Bit	Description	Bit meaning	
		0	1
(LSB)			
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*** (see Table 5.2.1)
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 master (is therefore read by the instrument) and is made up of registers (holding registers), of 2 bytes.

Table 5.2: Modbus Holding Registers

Reg. Nr.	Holding 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 5.2.1 section

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

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 5.1.1 section

- Format of the Output Status Register value
See 5.1.2 section

Setpoint DGT 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.

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 Channel X Status Register value
See 5.2.2 section

- 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

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 5.1.3 section

- Format of the Command Register value
See 5.2.1 section

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 5.2.1.1 section

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 5.2.3 section

Weight in one word (less significant word) and status repetition

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

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 5.1.1 section

- Format of the Output Status Register value
See 5.1.2 section

Setpoint DGTQ Family in one word (less significant word)

N°Reg.	Holding Registers		
41106 (1105)	Setpoint 1 ON temporary	(byte 1)	
	Setpoint 1 ON temporary	(byte 0)	
– – – – – – – – – – – – – – – –			
41111 (1110)	Setpoint 6 ON temporary	(byte 1)	
	Setpoint 6 ON temporary	(byte 0)	
41112 (1111)	Setpoint 1 OFF temporary	(byte 1)	
	Setpoint 1 OFF temporary	(byte 0)	
– – – – – – – – – – – – – – – –			
41117 (1116)	Setpoint 6 OFF temporary	(byte 1)	
	Setpoint 6 OFF temporary	(byte 0)	
41118 (1117)	Setpoint 1 ON permanent	(byte 1)	
	Setpoint 1 ON permanent	(byte 0)	
– – – – – – – – – – – – – – – –			
41123 (1122)	Setpoint 6 ON permanent	(byte 1)	
	Setpoint 6 ON permanent	(byte 0)	
41124 (1123)	Setpoint 1 OFF permanent	(byte 1)	
	Setpoint 1 OFF permanent	(byte 0)	
– – – – – – – – – – – – – – – –			
41129 (1128)	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.

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 Channel X Status Register value
See 5.2.2 section

- 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

5.2.1. 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).

MODES OF OPERATION:

Read only if the instrument is legal for trade **(only for DGT 4)**

Register	Data
40971	Instrument type 0 = independent channel 1 = dependent channel 2 = Transm
40972	Number of channel

When the register 40972 is written the instrument stores the instrument type and the set number of channels and restarts to start to work in the new selected mode.

Table 5.2.1: Command Register

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 **(2)**)
NET_SWITCH_REQUEST	4 (0004 Hex)	Display switching onto the NET WEIGHT **(3)**
GROSS_SWITCH_REQUEST	5 (0005 Hex)	Display switching on the GROSS WEIGHT **(3)**
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 **(4)**
READ_ALIBI	30 (001E Hex)	WEIGH READING ON ALIBI TROUGH SETUP PAGE (HOLDING REGISTERS 8-15) **(5)**
WRITE_ALIBI	31 (001F Hex)	STORAGE OF WEIGH ON ALIBI

HOLD_PEAK_WEIGHT	32 (0020 Hex)	BLOCK THE WEIGHT ON THE DISPLAY
UNLOCK_WEIGHT	33 (0021 Hex)	AFTER SECOND HOLD_ PEAK_WEIGHT ALLOWS TO UNLOCK THE WEIGHT ON THE DISPLAY AND TO SEE THE EFFECTIVE
RESTART_INSTRUMENT	34 (0022 Hex)	RESTART THE INSTRUMENT
READ CALIBRATION	35 (0023 Hex)	COPY OF CALIBRATION DATA OF THE CHANNEL EQUAL TO PARAMETER 1 INTO TEMPORARY AREA
WRITE_CALIBRATION	36 (0024Hex)	STORE OF TEMPORARY DATA INTO CALIBRATION DATA (NON VOLATILE MEMORY)
POINT_ACQUISITION	37 (0025 Hex)	PARAMETER 1 IS THE POINT TO ACQUIRE
ABORT_CALIBRATION	38 (0026 Hex)	ABORT THE CALIBRATION UNDER WAY
NUMBER_OF_PIECES	41 (0029 Hex)	COUNTING WITH PARAMETER 1 THAT IS THE NUMBER OF PIECES
APW	42 (002A Hex)	INSERT INPUT APW
PMU	43 (002B Hex)	COMMAND PMU WITH PARAMETER 1 THAT IS THE AVERAGE PIECE WEIGHT

(²) NOTE: ***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 → one should write 3000
If 2 decimals are set, in order to enter the value 3,00 → one should write 300

(³): functions active only in NTGS mode (net/gross switch).

(⁴) 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

OPTIONAL RELAYS (ONLY DGTQ PB)

bit 2 → RELAY 3 in which bit 2 = 1 → RELAY 3 CLOSED; bit 2 = 0 → RELAY 3 OPEN
bit 3 → RELAY 4 in which bit 3 = 1 → RELAY 4 CLOSED; bit 3 = 0 → RELAY 4 OPEN
bit 4 → RELAY 5 in which bit 4 = 1 → RELAY 5 CLOSED; bit 4 = 0 → RELAY 5 OPEN
bit 5 → RELAY 6 in which bit 5 = 1 → RELAY 6 CLOSED; bit 5 = 0 → RELAY 6 OPEN
bit 6 ÷15 (not used)

NOTES:

- ***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.

(⁵) WEIGH READING ON ALIBI

To read a weigh 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)

Table 5.2.1.A: 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*	

5.2.1.1. 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 semiautomatic 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 trough the Holding registers from 250 to 256.

(7) TRANSM MODE

Ability to perform zero / tare on any channel remotely via Modbus / Profibus.

Modbus Holding Registers 40202 to 40213 have the gross weights now.

New Modbus Holding Registers:

REGISTER	DATA
40214	Ch 1 net weight (byte 3,2)
40215	Ch 1 net weight (byte 1,0)
40216	Ch 2 net weight (byte 3,2)
40217	Ch 2 net weight (byte 1,0)
40218	Ch 3 net weight (byte 3,2)
40219	Ch 3 net weight (byte 1,0)
40220	Ch 4 net weight (byte 3,2)
40221	Ch 4 net weight (byte 1,0)
40222	Ch 1 tare weight (byte 3,2)
40223	Ch 1 tare weight (byte 1,0)
40224	Ch 2 tare weight (byte 3,2)
40225	Ch 2 tare weight (byte 1,0)
40226	Ch 3 tare weight (byte 3,2)
40227	Ch 3 tare weight (byte 1,0)
40228	Ch 4 tare weight (byte 3,2)
40229	Ch 4 tare weight (byte 1,0)

1 word weights:

REGISTER	DATA
41210	Ch 1 net weight (byte 1,0)
41211	Ch 2 net weight (byte 1,0)
41212	Ch 3 net weight (byte 1,0)
41213	Ch 4 net weight (byte 1,0)
41214	Ch 1 tare weight (byte 1,0)
41215	Ch 2 tare weight (byte 1,0)
41216	Ch 3 tare weight (byte 1,0)
41217	Ch 4 tare weight (byte 1,0)

New bits in the channel x status register:

BIT	DESCRIPTION	0	1
0	Gross weight polarity	+	-
1	Stability	No	yes
2	Underload condition	No	yes
3	Overload condition	No	yes
4	Gross weight zero zone	No	yes
5	**Net weight polarity**	+	-
6	**Tare active**	No	yes
7	**Preset Tare active**	No	yes

(9) COUNTER MODE

Modbus Input Registers (valid in pcs counter mode):

REGISTRO	DATO
30117	Set APW decimals
30118	Set APW unit (0=g , 1=kg , 2=t , 3=lb)
30119	Pcs value (byte 3,2)
30120	Pcs value (byte 1,0)
30121	APW value (byte 3,2)
30122	APW value (byte 1,0)

Modbus Holding Registers (only in count mode) to set directly APW value:

REGISTRO	DATA
41301	APW value (byte 3,2)
41302	APW value (byte 1,0)

Value is to be inserted as a fixed point value with a number of decimals equal to dec.APW parameter and in the UM.APW unit.
Example: unit = g, decimals = 5, to set 15.125 g as APW set the value 1512500 (higher bytes = 23, lower bytes = 5172).

5.2.2. Channel X Status Register (Table 5.2.2)

Bit	Description	Bit meaning	
		0	1
(LSB)			
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	*Not used*		
9	*Not used*		
10	*Not used*		
11	*Not used*		
12	*Not used*		
13	*Not used*		
14	*Not used*		
15	*Not used*		

5.2.3. SET-UP AREA

The set-up area is that which is memorised in flash (1024 bytes) and is made up 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 0	RANGE 1 channel 1	(LSB)
	343	Byte 1	RANGE 1 channel 1	
		Byte 2	RANGE 1 channel 1	
	344	Byte 3	RANGE 1 channel 1	(MSB)
		Byte 0	RANGE 2 channel 1	(LSB)
	345	Byte 1	RANGE 2 channel 1	
		Byte 2	RANGE 2 channel 1	
	346	Byte 3	RANGE 2 channel 1	(MSB)
	347			
Area Setup: PAGE 6 (16 bytes)	348			
		Byte 0	RANGE 1 channel 1 Division	(LSB)
	349	Byte 1	RANGE 1 channel 1 Division	(MSB)
		Byte 0	RANGE 2 channel 1 Division	(LSB)
	350	Byte 1	RANGE 2 channel 1 Division	(MSB)
	351			
			Channel 1 Decimals	
	352		Channel 1 Unit of Measure	**([5])**
	353			
	354			
	355			

	N°Reg.	Holding Registers	Description	
Area Setup: PAGE 14 (16 bytes)	412	Byte 0	RANGE 1 channel 2	(LSB)
		Byte 1	RANGE 1 channel 2	
	413	Byte 2	RANGE 1 channel 2	
		Byte 3	RANGE 1 channel 2	(MSB)
	414	Byte 0	RANGE 2 channel 2	(LSB)
		Byte 1	RANGE 2 channel 2	
	415	Byte 2	RANGE 2 channel 2	
		Byte 3	RANGE 2 channel 2	(MSB)
	416			
	417			
	418	Byte 0	RANGE 1 channel 2 Division	(LSB)
		Byte 1	RANGE 1 channel 2 Division	(MSB)
	419	Byte 0	RANGE 2 channel 2 Division	(LSB)
		Byte 1	RANGE 2 channel 2 Division	(MSB)
Area Setup: PAGE 15 (16 bytes)	420			
	421		Channel 2 Decimals	
			Channel 2 Unit of Measure	**([5])**
	422			
	423			
	424			
	425			
	426			
	427			
Area Setup: PAGE 22 (16 bytes)	476			
	477			
	478			
	479			
	480			
	481			
		Byte 0	RANGE 1 channel 3	(LSB)
	482	Byte 1	RANGE 1 channel 3	
		Byte 2	RANGE 1 channel 3	
	483	Byte 3	RANGE 1 channel 3	(MSB)
		Byte 0	RANGE 2 channel 3	(LSB)

	N°Reg.	Holding Registers	Description	
Area Setup: PAGE 23 (16 bytes)	484	Byte 1	RANGE 2 channel 3	
		Byte 2	RANGE 2 channel 3	
	485	Byte 3	RANGE 2 channel 3	(MSB)
	486			
	487			
		Byte 0	RANGE 1 channel 3 Division	(LSB)
	488	Byte 1	RANGE 1 channel 3 Division	(MSB)
		Byte 0	RANGE 2 channel 3 Division	(LSB)
	489	Byte 1	RANGE 2 channel 3 Division	(MSB)
	490			
			Channel 3 Decimals	
	491		Channel 3 Unit of Measure	**(5)**
Area Setup: PAGE 31 (16 bytes)	548			
	549			
	550			
	551	Byte 0	RANGE 1 channel 4	(LSB)
		Byte 1	RANGE 1 channel 4	
	552	Byte 2	RANGE 1 channel 4	
		Byte 3	RANGE 1 channel 4	(MSB)
	553	Byte 0	RANGE 2 channel 4	(LSB)
		Byte 1	RANGE 2 channel 4	
	554	Byte 2	RANGE 2 channel 4	
		Byte 3	RANGE 2 channel 4	(MSB)
	555			
Area Setup: PAGE 32 (16 bytes)	556			
	557	Byte 0	RANGE 1 channel 4 Division	(LSB)
		Byte 1	RANGE 1 channel 4 Division	(MSB)
	558	Byte 0	RANGE 2 channel 4 Division	(LSB)
		Byte 1	RANGE 2 channel 4 Division	(MSB)
	559			
	560		Channel 4 Decimals	
			Channel 4 Unit of Measure	**(5)**
	561			
	562			
	563			

(5) NOTE: ***Significance of the numeric value in the Unit of Measure field:***

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

5.3. Coils Status data area

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

Table 5.3: Modbus Coils Status

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

Only DGT-Q

N° Coil.	Coils Status	Bit meaning	
		0	1
00003 (2)	Digital output 3 (1)	Not active	Active
00004 (3)	Digital output 4 (1)	Not active	Active
00005 (4)	Digital output 5 (1)	Not active	Active
00006 (5)	Digital output 6 (1)	Not active	Active

(1) Writing allowed if the related output has no associated function.

6. CALIBRATION

Calibration holding registers:

REGISTER	DATA
40901	Number of calibration point
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 weigh (low)
40908	Zero calibration ADC value (low)
40909	Zero calibration ADC value (high)
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 (high)
40914	3rd calibration point ADC value (high)
40915	3rd calibration point ADC value (high)

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_ACQUISATION_UNDERWAY
2	MODBUS_CALIBRATION_ACQUISATION_OK
3	MODBUS_CALIBRATION_ACQUISATION_ERROR
4	MODBUS_CALIBRATION_OK
5	MODBUS_CALIBRATION_ERROR

SPECIFICAL COMMANDS:		
NUMBER	COMMAND	NOTES
35 (23 H)	READ_CALIBRATION	Copy of calibrations data of the cannel equal to parameter 1 into temporary area
36 (24 H)	WRITE_CALIBRATION	Store of temporary data into calibration data (non-volatile memory)
37 (25 H)	POINT_ACQUISITION	Parameter 1 is the point to acquire
38 (26 H)	ABORT_CALIBRATION	Abort the calibration under way

CALIBRATION SEQUENCE:	
A	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.
B	If necessary set metrological values in the registers 40951-40958
C	Set calibration points in register 40901
D	Set calibration weight(s) in registers 40902-40907
E	If a theoretical calibration is to be executed write directly registers 40908-40915
F	Otherwise unload the platform and use the command POINT_ACQUISITION with parameter equal to 0
G	Wait for calibration status (Input Register 30116) is equal to MODBUS_CALIBRATION_ACQUISTION_OK or MODBUS_CALIBRATION_ACQUISTION_ERROR
H	On error repeat from step f
I	On success load the platform with 1st calibration weight and use command POINT_ACQUISITION with parameter equal to 1
J	Wait for calibration status (Input Register 30116) is equal to MODBUS_CALIBRATION_OK or MODBUS_CALIBRATION_ERROR
K	On error repeat from step i
L	On success repeat step i for other calibration points (if any)
M	Use command WRITE_CALIBRATION to store the new calibration
N	Wait for calibration status (Input Register 30116) is equal to MODBUS_CALIBRATION_OK or MODBUS_CALIBRATION_ERROR
O	On error repeat from step a Note: while the command WRITE_CALIBRATION is in progress some Modbus reading timeout may happen because of the saving procedure