Eilersen
The Weighing Experts

# 5024G Options Guide Serial Modbus

**5024G LCD weighing terminal**

**Serial Modbus option: Direct RS485 connection**



Software: StdLim.140630.6v6s

Doc. no.: StdLim-140630-6v6-OG-SerialModbus-eng

Date: 2023-08-29

Rev.: 6v6



**Contact:**
**Eilersen Electric A/S**
**Kokkedal Industripark 4**
**DK-2980 Kokkedal**
**Denmark**
**www.eilersen.com**
**info@eilersen.com**
**Tel: +45 49 180 100**
**Fax: +45 49 180 200**

# Contents

# Introduction

This document describes the use of the RS485 Serial Modbus option on the 5024G Weighing Terminal from Eilersen Electric. With the software version stated on the front page and with the Serial Modbus option enabled the system can communicate on RS485 with an external controller/PLC using Modbus (RTU or ASCII) protocol, where the 5024G terminal acts as a Modbus slave.

With the stated software version installed the 5024G terminal can transfer 7 output word registers (14 output bytes) from the Modbus master to the 5024G terminal and transfer 7 input word registers (14 input bytes) from the 5024G terminal to the Modbus master.

Exchange of data between 5024G terminal and the external controller/PLC is made according to the profile/protocol described later (see ***Protocol description***).

This manual only describes the Serial Modbus option. For general information on the operation of the 5024G please see the 'Users guide'.

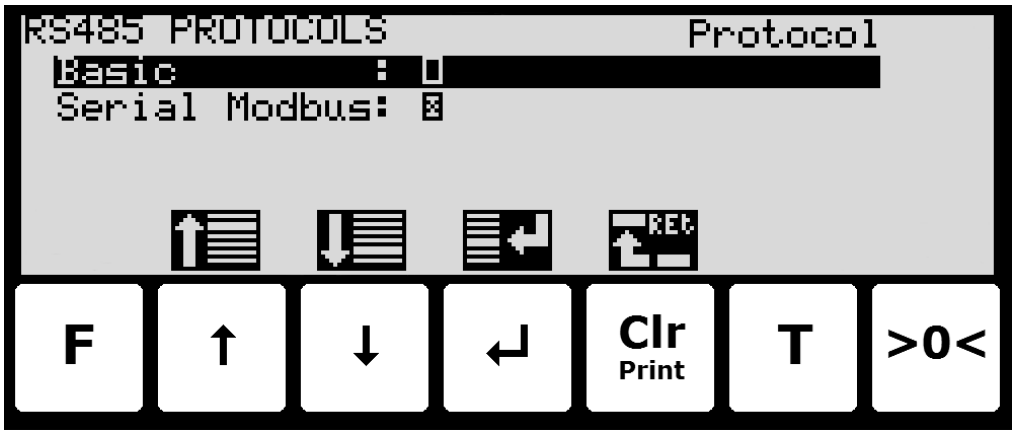
# How to

## – Configure RS485 Serial Modbus communication protocol

The terminal can perform Modbus communication on its RS485 serial communication connector (J4) if the Serial Modbus protocol is enabled.

### Enable Serial Modbus protocol

The Serial Modbus protocol must be enabled. This can be done in the **RS485 PROTOCOLS** screen shown below.

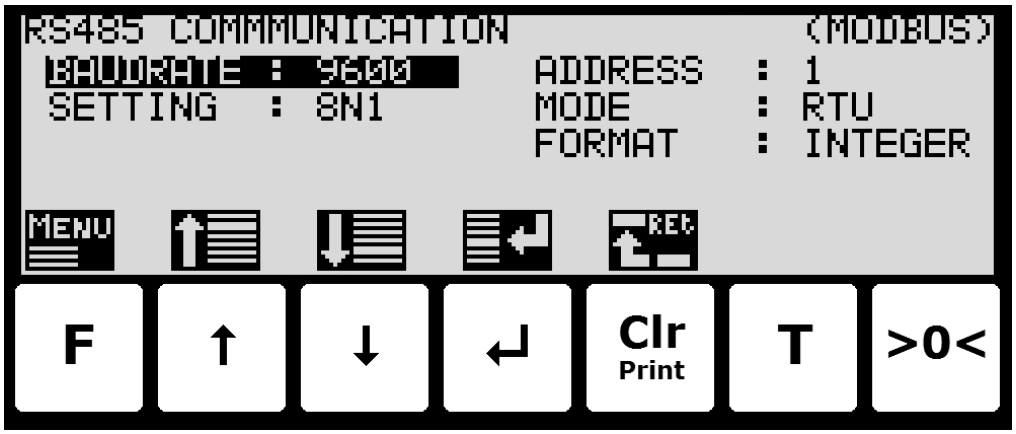



In the **RS485 PROTOCOLS** screen the '**Serial Modbus**' parameter is selected using the cursor, and ↵ is pressed to enable or disable the Serial Modbus protocol. An **X** in the corresponding box indicates the protocol is enabled.

**NOTE:** It is **ONLY** the shown **Serial Modbus** protocol that may be enabled.

## – Configure Serial Modbus settings

Modbus communication settings such as Baudrate, Databits, Stopbits, Parity, Address, Mode and Format are shown and can be set in the **RS485 COMMUNICATION** screen:




### Modbus Baudrate

The Modbus **BAUDRATE** is used to specify the actually used baudrate on the RS485 Modbus network. The Modbus baudrate can be set to a wide variety of commonly used baud rates in the interval 1200 bps to 115200 bps.

### Modbus serial Setting

The Modbus **SETTING** is used to specify the actual serial settings used on the RS485 Modbus network. The Modus settings can be set to a wide range of combinations all defined and indicated by 3 characters with the following format:

**<D> <P> <S>**

where:

**<D>** indicates the number of Databits and can be set to '**7**' or '**8**'.

**<P>** indicates the Parity and can be set to '**N**' (None), '**E**' (Even) or '**O**' (Odd).

**<S>** indicates the number of Stopbits and can be set to '**1**' or '**2**'.

Hence '**8N1**' would indicate 8 Databits, No Parity and 1 Stopbit.

Please notice that depending on the selected Modbus mode described below, this puts certain restraints on the used Modbus setting as follows:

In **ASCII** mode the following should be used:

7 databits.

1 stopbit if Parity is used (Even or Odd).

2 stopbit if Parity is NOT used (None).

In **RTU** mode the following should be used:

8 databits.

1 stopbit if Parity is used (Even or Odd).

2 stopbit if Parity is NOT used (None).

### Modbus Address

The Modbus **ADDRESS** is used to identify the 5024 weighing terminal on the RS485 Modbus network. Please notice that the address must be unique and different for each unit on the RS485 Modbus network. The address can be set from 0 to 31.

### Modbus Mode

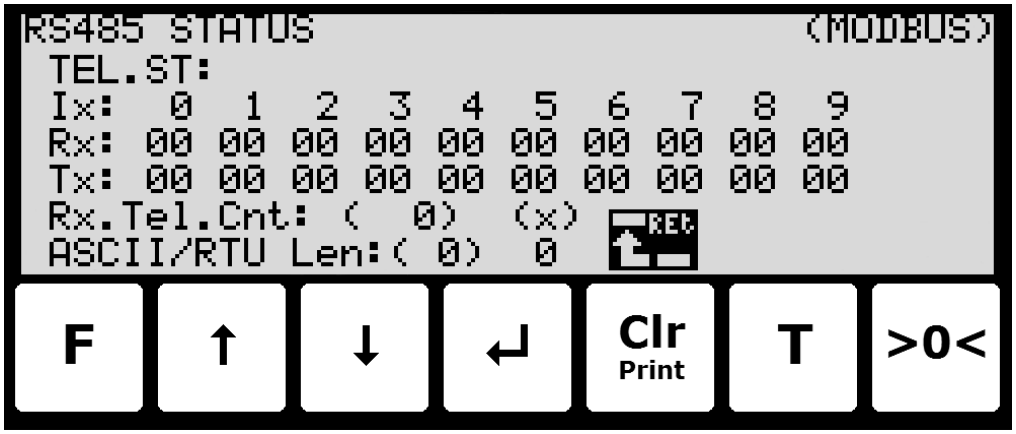
The Modbus **MODE** is used to determine the communication form on the RS485 Modbus network. The Modbus mode can be set to **ASCII** mode or **RTU** (binary) mode depending on the used Modbus communication form. Please refer to separate literature for a description of the two modes. Please notice that the selected Modbus mode sets limits on what serial settings should be used as described above.

### Modbus Format

The Modbus **FORMAT** is used to specify the representation of the data registers transferred on the Modbus. The Modbus format can be set to '**INTEGER**' format (32 bit signed integer) or '**FLOAT**' format (32 bit IEEE754 floating point). Normally '**INTEGER**' format is selected. If '**FLOAT**' is selected only 4 byte data registers (32 bit registers) are represented as floating point; i.e. 2 byte data registers (16 bit registers) are still represented as integers.

## – Monitor Modbus communication status

In the **RS485 STATUS** screen, the status of the Modbus communication can be read:




The shown Modbus communication data are solely for internal use during trouble shooting.

# Protocol description

## – Modbus communication using PPO

The Modbus communication is made using a *'parameter-process data object'* (PPO) consisting of 7 output word registers followed by 7 input word registers. This object is used during both reception and transmission of data. The structure consists of the following registers:

| Starting Address | Holding Register | Number of Points/Registers | Byte Count | Register Type | Register Content |
|---|---|---|---|---|---|
| ***Write data (From Modbus master to 5024G):*** | | | | | |
| 0 | 40001 | 1 | 2 | 2, Rd/Wr | MDS_PCA |
| 1 | 40002 | 1 | 2 | 2, Rd/Wr | PNU |
| 2 | 40003 | 2 | 4 | 4, Rd/Wr | PVA |
| 4 | 40005 | 1 | 2 | 2, Rd/Wr | CTW |
| 5 | 40006 | 2 | 4 | 4, Rd/Wr | MRV |
| ***Read data (From 5024G to Modbus master):*** | | | | | |
| 7 | 40008 | 1 | 2 | 2, Rd | MDS_PCA |
| 8 | 40009 | 1 | 2 | 2, Rd | PNU |
| 9 | 40010 | 2 | 4 | 4, Rd | PVA |
| 11 | 40012 | 1 | 2 | 2, Rd | STW |
| 12 | 40013 | 2 | 4 | 4, Rd | MAV |

where:

MDS_PCA stands for Mode Selector and Parameter Characteristics

PNU stands for Parameter Number

PVA stands for Parameter Value

CTW stands for Control Word

MRV stands for Main Reference Value

STW stands for Status Word

MAV stands for Main Actual Value

In the following the meaning of the individual registers of the object is explained further.

**IMPORTANT:** During transfer/reception of data (i.e. the MAV) it is up to the master (the PLC) to ensure consistent data, when a parameter consisting of several word registers is read/updated and when AS/MAV or RS/MRV is read/set.

## MDS_PCA

The MDS part is the most significant byte (MSB) of the MDS_PCA register, and indicates which value is to be transferred as **Main Reference Value** (MRV) and as **Main Actual Value** (MAV).

The PCA part is the least significant byte (LSB) of the MDS_PCA register and determines (along with the PNU and PVA registers) what is to happen with a given parameter.

| **MDS** | | | | | | | | **PCA** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RS | | | | AS | | | | RC | | | | | | | |

**RS:** Reference value selector (Values: 0..15)

**AS:** Actual value selector (Values: 0..15)

**RC:** Request/Response Characteristics (Values: 0..255)

MDS contains an RS part for selection of **Main Reference Value** (MRV) and an AS part for selection of **Main Actual Value** (MAV).

The PCA part contains an RC part for 'request' and 'response' indication during write and read.

RC is used by the master to tell the slave which ’request’ is desired. Similarly, the slave uses RC to inform the master the status of the received 'request' (’response’).

The contents of RC has the following function during ’request:

| REQUEST | FUNCTION |
|---|---|
| 0 | No request |
| 1 | Request parameter value |
| 2 | Change parameter value in word register (2 bytes) |
| 3 | Change parameter value in double word register (4 bytes) |
| *Others* | *Reserved for future use* |

The contents of RC has the following function during response:

| RESPONSE | FUNCTION |
|---|---|
| 0 | No response |
| 1 | Transfer parameter value in word register (2 bytes) |
| 2 | Transfer parameter value in double word register (4 bytes) |
| 3 | Request rejected (incl. Error#, see later) |
| 4 | Cannot be serviced by PCA, PNU and PVA interface |
| *Others* | *Reserved for future use* |

### PNU

The PNU part indicates the parameter number of the parameter to be read/changed. The parameters and their function are described below.

### PVA

The PVA part contains a double word for reception and transmission of parameter values. The PVA part transfers single word parameters in the least significant word (LSW).

If the slave rejects a request from the master the RC part assumes the value 3 (see above) and the error number itself is transferred in the least significant word (LSW) of the PVA register. The following error indications are possible:

| ERROR # | CAUSE |
|---|---|
| 0 | Illegal command for this PNU or PNU not used. |
| *1* | *Reserved for future use* |
| 2 | Upper or lower limit exceeded |

### CTW/STW

During communication from the master to the slave a Control Word (CTW) is used. Using the Control Word (CTW) it is possible to tell the slave how to react as different commands can be transferred to the slave.

During communication from the slave to the master a Status Word (STW) is used. Using the Status Word (STW) it is possible for the master to gain information on the status of the slave.

### MRV/MAV

During communication from the master to the slave a **Main Reference Value** (MRV) is used; a setpoint. Using the RS defines exactly which value is transferred as MRV.

During communication from the slave to the master a **Main Actual Value** (MAV) is used; the actual value. Using the AS defines exactly which value is transferred as MAV.

## – Communication overview (Modbus data registers)

Below is a complete list of data registers available on 5024G weighing terminal. The list specifies the *Starting Address*, the *Holding Register*, the *Number of Points/Registers*, the *Byte Count*, the *Register Type* (register size in bytes and if it's a Read/Write register) as well as the *Register Content*.

| Starting Address | Holding Register | Number of Points/Registers | Byte Count | Register Type | Register Content |
|---|---|---|---|---|---|
| ***Write data (From Modbus master to 5024G):*** | | | | | |
| 0 | 40001 | 1 | 2 | 2, Rd/Wr | MDS_PCA |
| 1 | 40002 | 1 | 2 | 2, Rd/Wr | PNU |
| 2 | 40003 | 2 | 4 | 4, Rd/Wr | PVA |
| 4 | 40005 | 1 | 2 | 2, Rd/Wr | CTW |
| 5 | 40006 | 2 | 4 | 4, Rd/Wr | MRV |
| ***Read data (From 5024G to Modbus master):*** | | | | | |
| 7 | 40008 | 1 | 2 | 2, Rd | MDS_PCA |
| 8 | 40009 | 1 | 2 | 2, Rd | PNU |
| 9 | 40010 | 2 | 4 | 4, Rd | PVA |
| 11 | 40012 | 1 | 2 | 2, Rd | STW |
| 12 | 40013 | 2 | 4 | 4, Rd | MAV |

**Please note the following:**

1. All weights are transferred as shown in the display without a decimal point (i.e. 300.0 kg is transferred as 3000 and 67.2 kg is transferred as 672).
2. All negative numbers are transferred as 2-complement numbers.
3. Actual unit and decimal point position can be read from the appropriate parameter.
4. In Modbus communication the **M**ost **S**ignificant **B**yte (MSB) of a word is first.
5. In Modbus communication the **L**east **S**ignificant **W**ord (LSW) of a double word is first.
6. When reading/transferring data consisting of multiple holding registers (such as gross and net weight in the MAV part) it is up to the Modbus master to ensure consistent data (data originate from the same telegram). This is typically done by performing a "Read Holding Register" command requesting the read of 2 points (for a double word such as gross or net weight) and then determining the actual result from the response telegram. In similar way the transfer of a double word parameter should be done using a single "Preset Multiple Registers" command requesting the write/update of 2 registers.

## – RS – Reference Value Selector, MRV – Main Reference Value

| RS<br><br>Reference Value Selector | MRV<br><br>Main Reference Value |
|---|---|
| *0* | *Not used* |
| *Others* | *Not used* |

## – AS – Actual Value Selector, MAV – Main Actual Value

| AS<br><br>Actual Value Selector | MAV<br><br>Main Actual Value |
|---|---|
| *0* | *Not used* |
| 1 | Actual gross weight |
| 2 | Actual net weight |
| *Others* | *Not used* |

**Actual gross weight** is the actual gross weight on the 5024G terminal.

**Actual net weight** is the actual net weight on the 5024G terminal.

## – CTW – Control Word

| Bit | Function |
|---|---|
| 0 | Zero |
| 1 | Autotare (zero of net weight) |
| 2 | Start dosing |
| 3 | Stop dosing |
| 4 | Registration |
| *Others* | *Not used* |

**Zero** must be activated if a zero of the gross weight is desired.

**Autotare** must be activated if a zero of the net weight is desired.

**Start dosing** must be activated if a start of dosing is desired.

**Stop dosing** must be activated if a dosing is to be stopped before the fine limit is reached. If the terminal is set to perform automatic registration on time this will take place afterwards.

**Registration** must be activated if a registration of the actual net weight is desired. Any dosing in progress will be aborted before registration.

## – STW – Status Word

| Bit | Function |
|---|---|
| 0 | Weight reading not possible |
| 1 | Zero OK |
| 2 | Zero not possible |
| 3 | Autotare OK |
| 4 | Autotare not possible |
| 5 | Start dosing OK |
| 6 | Start dosing not possible |
| 7 | Stop dosing OK |
| 8 | Stop dosing not possible |
| 9 | Registration OK |
| 10 | Registration not possible |
| 11 | Fine dosing |
| 12 | Coarse dosing |
| *13* | *Not used* |
| 14 | Registration ready |
| 15 | OK – always ON |

**Weight reading not possible** is active when the 5024G terminal is unable to determine weight.

**Zero OK** is active when zero was possible.*)

**Zero not possible** is active when zero was NOT possible.*)

**Autotare OK** is active when autotare was possible.*)

**Autotare not possible** is active when autotare was NOT possible.*)

**Start dosing OK** is active when start of dosing was possible.*)

**Start dosing not possible** is active when start of dosing was NOT possible.*)

**Stop dosing OK** is active when stop of dosing was possible.*)

**Stop dosing not possible** is active when stop of dosing was NOT possible.*)

**Registration OK** is active when registration of net weight was possible.*)

**Registration not possible** is active when registration of net weight was NOT possible.*)

**Fine dosing** is active during dosing until the fine limit (pos. adjusted for afterflow) is reached.

**Coarse dosing** is active during dosing when the net weight is below the coarse limit.

**Registration ready** is active when a registration is ready. The bit is cleared when a new dosing is started.

**OK – always ON** is always activated. Can be used as a control of the communication.

Bits marked with *) are cleared again when the corresponding request bit is cleared.

## – Parameters

| NO | TYPE | PARAMETER |
|---|---|---|
| 1 | 4, R | Actual gross weight |
| 2 | 4, R | Actual net weight |
| 3 | 4, RW | Fine limit |
| 4 | 4, RW | Coarse limit |
| *5* | - | *Not used* |
| 6 | 4, R | Last registered amount |
| 7 | 4, R | Total dosed amount |
| 8 | 4, R | Total number of weighings |
| 10 | 2, R | Unit<br>*0: kg*<br>*1: lbs*<br>*2: gram* |
| 11 | 2, R | Decimal point position |
| 20 – 35 | 2, R | Load cell-Status[x] |
| 40 – 55 | 4, R | Load cell-Gross[x] |
| *Others* | | *Not used* |

**Actual gross weight** is the actual gross weight on the 5024G terminal.

**Actual net weight** is the actual net weight on the 5024G terminal.

**Fine limit** contains the fine limit used during dosing.

**Coarse limit** contains the coarse limit used during dosing.

**Last registered amount** contains the result (registration) of the last dosing.

**Total dosed amount** contains the total dosed amount.

**Total number of weighings** contains the total number of weighings.

**Unit** indicates the unit used in the display reading. It should be used to scale weight indications received/transmitted using the Modbus communication.

**Decimal point position** indicates the number of digits after the decimal point in the display reading. It should be used to scale weight indications received/transmitted using the Modbus communication.

**Load cell-Status[x]** contains the actual status for load cell x.

**Load cell-Gross[x]** contains the actual gross signal (not zeroed) on load cell x.

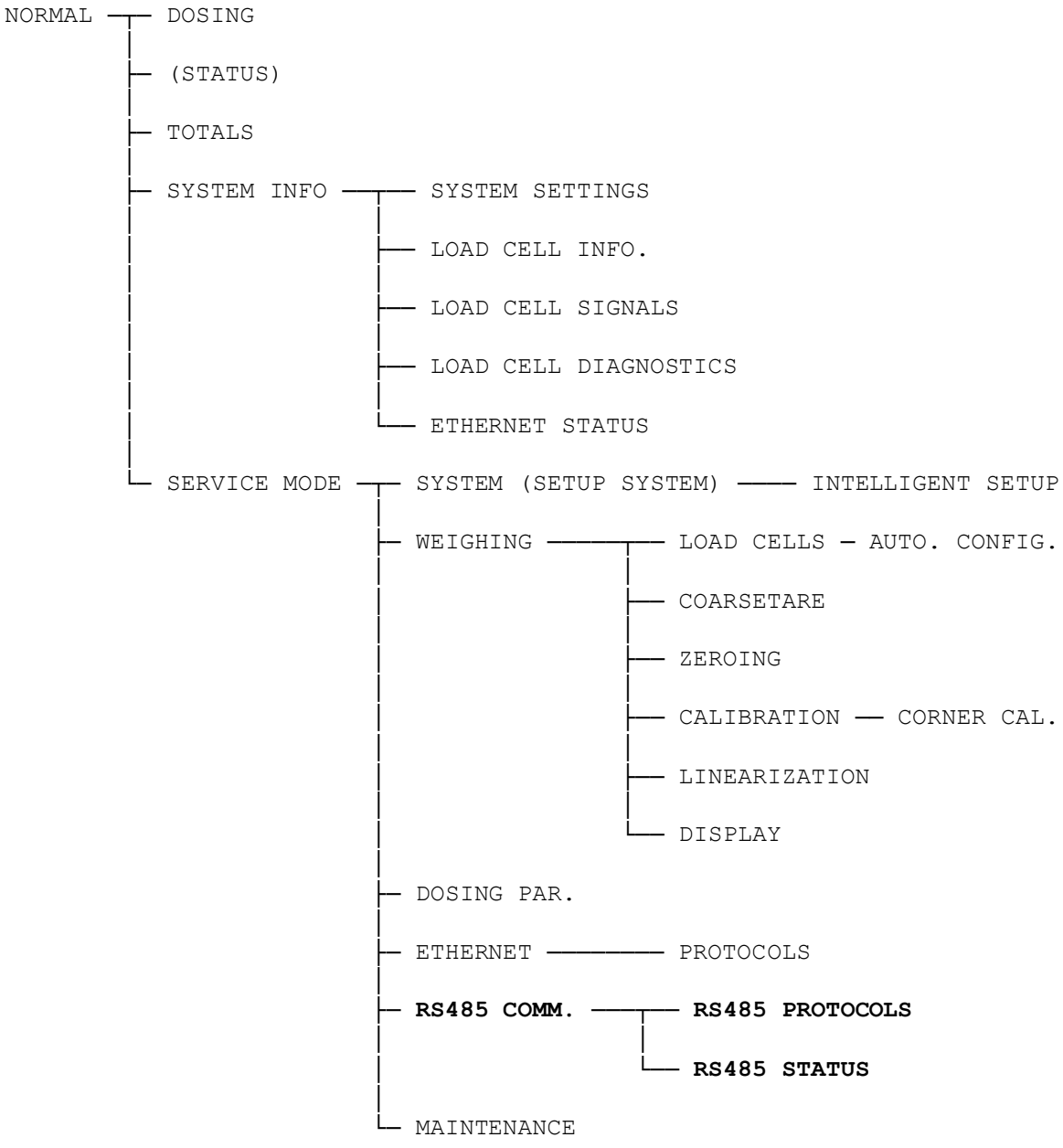
# Trouble shooting

| ***Problem*** | **Explanation and possible solutions** |
|---|---|
| *PLC unable to receive/transmit data from/to 5024G* | Check connection between PLC and 5024G terminal is ok (see below).<br>Check configuration of 5024G terminal is ok (see below).<br>Check configuration of Serial Modbus master (PLC) is ok (see below). |
| *No connection between PLC and 5024G terminal* | Check the RS485 cabling between 5024G and PLC is made correctly.<br>Check the RS485 cable is not damaged.<br>Check the 5024G (in the **RS485 COMMUNICATION** screen) and the PLC are configured correctly to the same serial settings.<br>Check the correct protocol has been enabled in the **RS485 PROTOCOLS** screen. |
| *Wrong configuration of 5024G terminal* | Check parameters in the **RS485 COMMUNICATION** screen are configured correctly:<br>- Check "***BAUDRATE***" and "***SETTING***" parameters are set correct.<br>- Check "***ADDRESS***" parameter matches the desired Modbus address.<br>- Check "***MODE***" parameter is set correct (RTU / ASCII).<br>- Check "***FORMAT***" parameter is set correct (INTEGER / FLOAT).<br>Check the **Serial Modbus** protocol has been enabled in the **RS485 PROTOCOLS** screen. |
| *Wrong configuration of PLC* | Check the PLC is configured correctly. In this application 7 output word registers and 7 input word registers are used.<br>Check the PLC is configured with correct parameters matching the configuration on 5024G terminal. |
| *Values change rapidly between random values* | Check the Modbus master (PLC) uses correct data format:<br>- in word registers **M**ost **S**ignificant **B**yte (MSB) is first, and **L**east **S**ignificant **B**yte (LSB) is last.<br>- in double words **L**east **S**ignificant **W**ord (LSW) is first, and **M**ost **S**ignificant **W**ord (MSW) is last.<br>- is correct "***FORMAT***" selected (INTEGER / FLOAT) ? |
| *Implemented protocol does not behave as expected* | Compare implemented Modbus protocol (PLC program) with the Modbus protocol description above.<br>For instance, check if the **OK – always ON** bit in **STW** is ON as expected. |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

# Appendices

## Appendix A – Screens overview

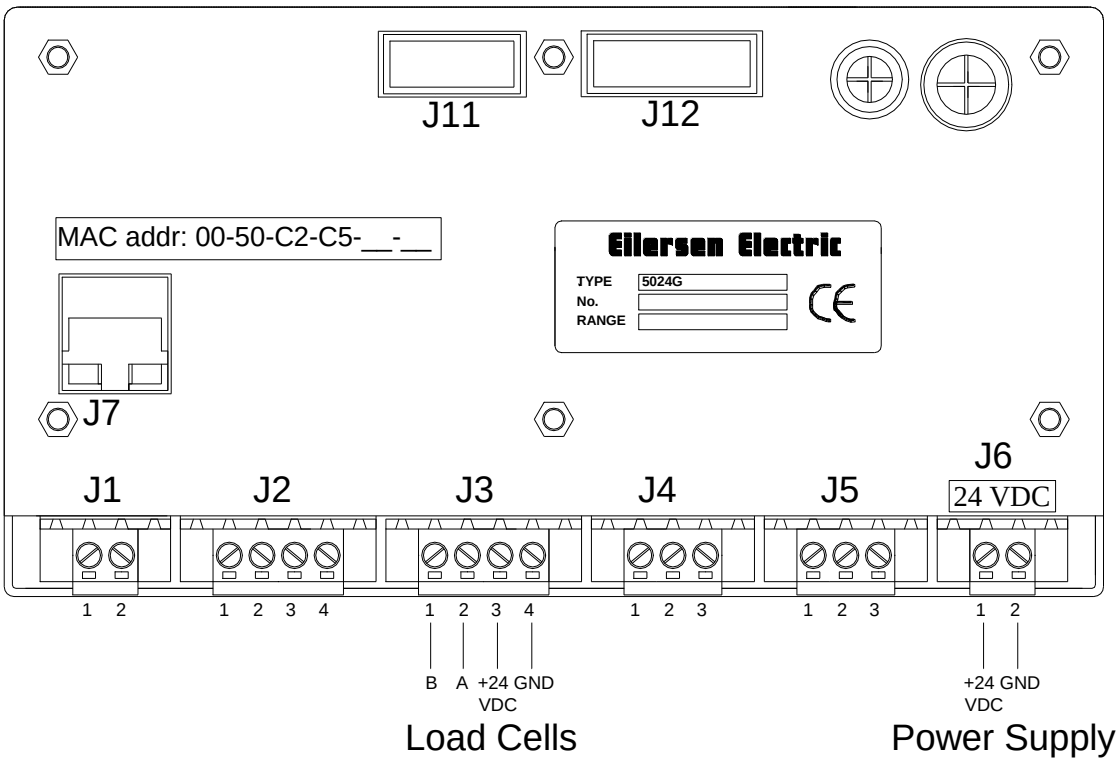
The system has the following screens, which are selected using the menu system.



## Appendix B – Electrical connection of 5024G to Serial Modbus

The following describes the electrical connection of the 5024G terminal to Serial Modbus using a RS485 connection.

### Rear view




### RS485 communication connector on 5024G

The 3 pin RS485 serial communication connector (J4) on the backside of the 5024G terminal is the connector that is used to connect the 5024G system to the serial Modbus master. This connector has the following connections:

| J4 pin | Function | Connection |
|---|---|---|
| 1 | **RS485-B** (negative line) | **Serial Modbus master:** RS485-B |
| 2 | **RS485-A** (positive line) | **Serial Modbus master:** RS485-A |
| 3 | **RS485-GND** | **Serial Modbus master:** RS485-GND |

**Please notice:** A and B line definitions may be switched on external equipment. Especially on Siemens equipment and a few other German manufacturers A and B lines definitions are different.

## Appendix C – Reading of Modbus data

Modbus data registers on the 5024G weighing terminal are read by the Modbus Master by performing a ”Read Holding Register” request (0x03) to the 5024G weighing terminal. The *Holding Register* as well as the *Number of Points* used in the request is specified for each data register in the Modbus data register table shown earlier.

### Telegram format (Read Holding Register – 1 point)

A ”Read Holding Register” command requesting the read of 1 point from start address 1 (Holding Register 40002) has the following telegram format:

| | HEX | ASCII (chars) | RTU (binary) |
|---|---|---|---|
| Header: | xx | : | none |
| Slave Adr.: | xx | x x | xxxxxxxx |
| Function: | 03 | 0 3 | 00000011 |
| Start Adr.(H): | 00 | 0 0 | 00000000 |
| Start Adr. (L): | 01 | 0 1 | 00000001 |
| # of points (H): | 00 | 0 0 | 00000000 |
| # of points (L): | 01 | 0 1 | 00000001 |
| Error Check: | xx | LRC(2 bytes) | CRC(2 bytes) |
| Trailer: | xx | CR LF | none |

To this request the 5024G terminal responds with the following telegram:

| | HEX | ASCII (chars) | RTU (binary) |
|---|---|---|---|
| Header: | xx | : | none |
| Slave Adr.: | xx | x x | xxxxxxxx |
| Function: | 03 | 0 3 | 00000011 |
| Byte Count: | 02 | 0 2 | 00000010 |
| Data0(H) | xx | x x | xxxxxxxx |
| Data0(L) | xx | x x | xxxxxxxx |
| Error Check: | xx | LRC(2 bytes) | CRC(2 bytes) |
| Trailer: | xx | CR LF | none |

### Telegram format (Read Holding Register – 2 points)

A ”Read Holding Register” command requesting the read of 2 points from start address 2 (Holding Register 40003) has the following telegram format:

| | HEX | ASCII (chars) | RTU (binary) |
|---|---|---|---|
| Header: | xx | : | none |
| Slave Adr.: | xx | x x | xxxxxxxx |
| Function: | 03 | 0 3 | 00000011 |
| Start Adr.(H): | 00 | 0 0 | 00000000 |
| Start Adr. (L): | 02 | 0 2 | 00000010 |
| # of points (H): | 00 | 0 0 | 00000000 |
| # of points (L): | 02 | 0 2 | 00000010 |
| Error Check: | xx | LRC(2 bytes) | CRC(2 bytes) |
| Trailer: | xx | CR LF | none |

To this request the 5024G terminal responds with the following telegram:

| | HEX | ASCII (chars) | | RTU (binary) |
|---|---|---|---|---|
| Header: | xx | : | | none |
| Slave Adr.: | xx | x | x | xxxxxxxx |
| Function: | 03 | 0 | 3 | 00000011 |
| Byte Count: | 04 | 0 | 4 | 00000100 |
| Data0(H)(LSW) | xx | x | x | xxxxxxxx |
| Data0(L)(LSW) | xx | x | x | xxxxxxxx |
| Data1(H)(MSW) | xx | x | x | xxxxxxxx |
| Data1(L)(MSW) | xx | x | x | xxxxxxxx |
| Error Check: | xx | LRC(2 bytes) | | CRC(2 bytes) |
| Trailer: | xx | CR | LF | none |

## Appendix D – Writing of Modbus data

Modbus data registers on the 5024G weighing terminal are written by the Modbus Master by performing a ”Preset Multiple Registers” request (0x10) to the 5024G weighing terminal. The *Holding Register*, the *Number of Registers* as well as the *Byte Count* used in the request is specified for each data register in the Modbus data register table shown earlier.

### Telegram format (Preset Multiple Registers – 1 register)

A ”Preset Multiple Registers” command requesting the write/update of 1 register from start address 0 (Holding Register 40001) has the following telegram format:

| | HEX | ASCII (chars) | | RTU (binary) |
|---|---|---|---|---|
| Header: | xx | : | | none |
| Slave Adr.: | xx | x | x | xxxxxxxx |
| Function: | 10 | 1 | 0 | 00010000 |
| Start Adr.(H): | 00 | 0 | 0 | 00000000 |
| Start Adr. (L): | 00 | 0 | 0 | 00000000 |
| # of registers (H): | 00 | 0 | 0 | 00000000 |
| # of registers (L): | 01 | 0 | 1 | 00000001 |
| Byte Count: | 02 | 0 | 2 | 00000010 |
| Data0(H) | xx | x | x | xxxxxxxx |
| Data0(L) | xx | x | x | xxxxxxxx |
| Error Check: | xx | LRC(2 bytes) | | CRC(2 bytes) |
| Trailer: | xx | CR | LF | none |

To this request the 5024G terminal responds with the following telegram:

| | HEX | ASCII (chars) | | RTU (binary) |
|---|---|---|---|---|
| Header: | xx | : | | none |
| Slave Adr.: | xx | x | x | xxxxxxxx |
| Function: | 10 | 1 | 0 | 00010000 |
| Start Adr.(H): | 00 | 0 | 0 | 00000000 |
| Start Adr. (L): | 00 | 0 | 0 | 00000000 |
| # of registers (H): | 00 | 0 | 0 | 00000000 |
| # of registers (L): | 01 | 0 | 1 | 00000001 |
| Error Check: | xx | LRC(2 bytes) | | CRC(2 bytes) |
| Trailer: | xx | CR | LF | none |

## Telegram format (Preset Multiple Registers – 2 registers)

A ”Preset Multiple Registers” command requesting the write/update of 2 registers from start address 6 (Holding Register 40007) has the following telegram format:

| | HEX | ASCII (chars) | | RTU (binary) |
|---|---|---|---|---|
| Header: | xx | : | | none |
| Slave Adr.: | xx | x | x | xxxxxxxx |
| Function: | 10 | 1 | 0 | 00010000 |
| Start Adr.(H): | 00 | 0 | 0 | 00000000 |
| Start Adr. (L): | 06 | 0 | 6 | 00000110 |
| # of registers (H): | 00 | 0 | 0 | 00000000 |
| # of registers (L): | 02 | 0 | 2 | 00000010 |
| Byte Count: | 04 | 0 | 4 | 00000100 |
| Data0(H)(LSW) | xx | x | x | xxxxxxxx |
| Data0(L)(LSW) | xx | x | x | xxxxxxxx |
| Data1(H)(MSW) | xx | x | x | xxxxxxxx |
| Data1(L)(MSW) | xx | x | x | xxxxxxxx |
| Error Check: | xx | LRC(2 bytes) | | CRC(2 bytes) |
| Trailer: | xx | CR | LF | none |

To this request the 5024G terminal responds with the following telegram:

| | HEX | ASCII (chars) | | RTU (binary) |
|---|---|---|---|---|
| Header: | xx | : | | none |
| Slave Adr.: | xx | x | x | xxxxxxxx |
| Function: | 10 | 1 | 0 | 00010000 |
| Start Adr.(H): | 00 | 0 | 0 | 00000000 |
| Start Adr. (L): | 06 | 0 | 6 | 00000110 |
| # of registers (H): | 00 | 0 | 0 | 00000000 |
| # of registers (L): | 02 | 0 | 2 | 00000010 |
| Error Check: | xx | LRC(2 bytes) | | CRC(2 bytes) |
| Trailer: | xx | CR | LF | none |

# Revision History

| Date | Author | Rev. | *Update* |
|---|---|---|---|
| 2023-08-29 | HJA | 6v6 | *Initial document created and adapted.*<br>*(based on StdLim-140630-6v6-OG-ModbusTCP-eng)* |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Contact

With further questions or improvement suggestions please contact us:

**Eilersen Electric A/S**
**Kokkedal Industripark 4**
**DK-2980 Kokkedal**
**Denmark**
**www.eilersen.com**
**info@eilersen.com**
**Tel: +45 49 180 100**
**Fax: +45 49 180 200**