# Technical Reference

# rinWIRE Serial Protocol

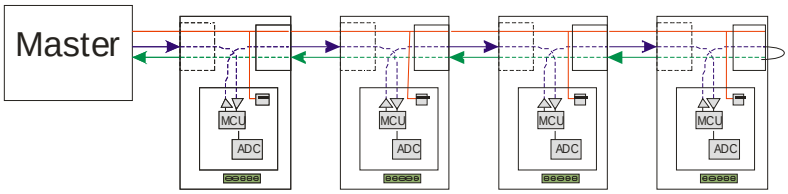
# 1. Overview

1.1. The rinWIRE digital interface has the following characteristics:

- Uses standard RS485 or RS232 serial communications signals. All devices on a rinWIRE network must use the same serial communications signals.
- Each sensor regenerates the communication signals so there is no need for network termination devices to balance the network as with standard RS485 or for any special hardware to support networking with RS232.
- rinWIRE supports both individually addressed communications and broadcast messages. Due to the network architecture devices do not need to have unique addresses when first added to the network.
- Up to 31 devices can be connected to the rinWIRE network.
- The rinWIRE network presents to the host controller as a single multi-channel device. Broadcast queries can be issued which collect responses from all units in a single transaction.

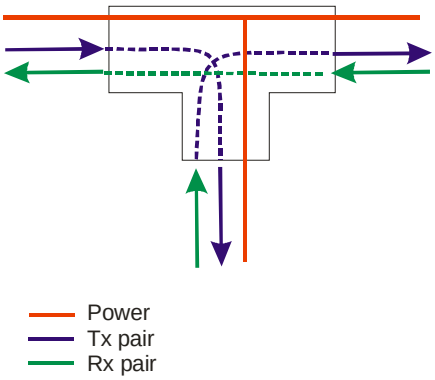
# 2. rinWIRE Communications Cabling System

2.1. Modules can be connected together to form a simple ring structure with the return path built into the cabling as follows:

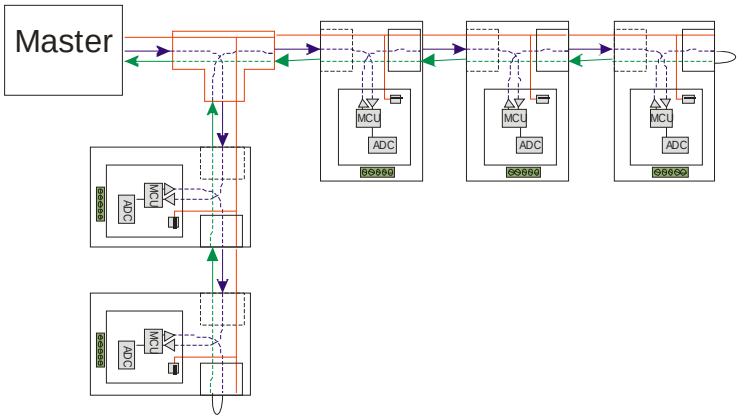



Note that the loopback connection on the last connection in the chain.

2.2. Cabling can also be split into a number of connection chains using a rinWIRE T-junction:



2.3. Various network configurations can be created using a combination of T-junctions:



# 3. Protocol Overview

The protocol uses ASCII characters with a single master POLL / RESPONSE message structure. All information and services are provided by registers each of which has its own register address.

## 3.1. Basic Message Format

The basic message format is as follows:

| **ADDR** | **CMD** | **REG** | **:** | **DATA** |
|---|---|---|---|---|
| *Header* | | | | *Data* |

### ADDR

ADDR is a two character hexadecimal field corresponding with the following:

| ADDR | Field Name | Description |
|---|---|---|
| $80_H$ | Response | '0' for messages sent from the master (POLL).<br>'1' for messages received from a module (RESPONSE) |
| $40_H$ | Error | Set to indicate that the data in this message is an error code and not a normal response. |
| $20_H$ | Reply Required | Set by the master to indicate that a reply to this message is required by any slave that it is addressed to. If not set, the slave should silently perform the command. |
| $00_H$ .. $1F_H$ | Module Address | Valid addresses are $01_H$ to $1F_H$ (1 .. 31).<br>$00_H$ is the broadcast address. All sensors must process broadcast commands. When replying to broadcasts, sensors reply with their own address in this field. |

**Note:** The hexadecimal codes are combined in the fields described above when multiple options are active at the same time. For example an error response message from sensor address 5 would have an ADDR code of $C5_H$ ($80_H + 40_H + 05_H$).

**CMD** is a two character hexadecimal field:

| CMD | Command | Description |
|---|---|---|
| $05_H$ | Read Literal | Read register contents in a 'human readable' format |
| $11_H$ | Read Final | Read register contents in a hexadecimal data format |
| $16_H$ | Read Final (Decimal) | Same as Read Final except numbers are decimal. |
| $12_H$ | Write Final | Write the DATA field to the register. |
| $17_H$ | Write Final (Decimal) | Same as Write Final except numbers are decimal. |
| $10_H$ | Execute | Execute function defined by the register using parameters supplied in the DATA field. |

**REG** is a four character hexadecimal field: Refer to specific reference documentation for the particular module in use for a full listing of the available registers.

| REG | Register | Description |
|---|---|---|
| $0005_H$ | Serial Number | Returns sensor serial number |
| $0020_H$ | ADC Sample Number | Read current sample number since last power on. (32 bit) |
| $0021_H$ | System Status | This register can be read to obtain the status of the instrument.<br>32 status bits sent as 8 hex chars, where:<br>$00020000_H$: Overload<br>$00010000_H$: Underload<br>$00008000_H$: Error (see System Error)<br>$00004000_H$: SETUP menus active<br>$00002000_H$: Calibration in progress<br>$00001000_H$: Motion<br>$00000800_H$: Centre of Zero<br>$00000400_H$: Zero<br>$00000200_H$: Net |
| $0022_H$ | System Error | Diagnostic Errors |
| $0023_H$ | Absolute mV/V | Absolute mV/V reading where 10000 = 1.0mV/V |
| $0025_H$ | Displayed Weight | Gross or Net weight depending on which is active |
| $0026_H$<br>$0027_H$<br>$0028_H$ | Gross,Net,Tare Weight | Gross Net or Tare weights |

**:DATA** carries the required information for the message

| | |
|---|---|
| **:** | ':' (COLON) character is used to separate the header (ADDR CMD REG) and DATA information. |
| **DATA** | Carries the information for the message. Some messages require no DATA (eg Read Commands) so the field is optional. |

## 3.2. Termination

Message termination is possible in two ways.

1. 8 : For normal communications that do not involve checksums use either a CRLF (ASCII 13, ASCII 10) as a terminator or a semicolon (';' ASCII ). There is no start-of-message delimiter:

   <Message> 8

2. To use a checksum the message is framed as:

   **SOH** <Message> ***CRC*** **EOT**

| | |
|---|---|
| **SOH** | ASCII 01 |
| **CRC** | a 4 character hexadecimal field comprising the 16 bit CRC checksum. The CRC uses the 16 bit CCITT polynomial calculation and includes only the contents of the <Message> section of the transmission. |
| **EOT** | ASCII 04 |

## 3.3. Error Handling

If a command cannot be processed, the sensor returns an error. The ERROR bit in the ADDR field is set and the DATA field contains the Error Code as follows:

| Error | DATA | Description |
|---|---|---|
| Unknown Error | $C000_H$ | Error is of unknown type |
| Not Implemented Error | $A000_H$ | Feature not implemented on this device |
| Access Denied | $9000_H$ | Passcode required to access this register |
| Data Under Range | $8800_H$ | Data too low for this register |
| Data Over Range | $8400_H$ | Data too high for this register |
| Illegal Value | $8200_H$ | Data not compatible with this register |
| Illegal Operation | $8100_H$ | CMD field unknown |
| Bad parameter | $8040_H$ | Parameter not valid for this execute register |
| Menu in Use | $8020_H$ | Cannot modify register values while SETUP menus are active |
| Viewer Mode required | $8010_H$ | Advanced operation chosen which requires the sensor to be in viewer mode. |
| Checksum required | $8008_H$ | A checksum is required for the chosen command. |

**Table 1: Network error codes**

## 3.4. Overall Communication Framing

In the general case when more than one sensor is connected on a RING network it is necessary to frame the message using special framing characters **<DC2>** and **<DC4>.**

**<DC2>** and **<DC4>** are the characters ASCII $12_H$ and ASCII $14_H$ respectively, here called ‘Echo-On’ and ‘Echo-Off’.

Upon receiving the **<DC2>** character the sensor begins echoing all received characters at the hardware level.

The **<DC4>** character halts the communications echo of incoming characters and provides an opportunity for the sensor to insert its response to the command. The sensor transmits any response it has and then appends a new **<DC4>** character.

The following example shows a complete POLL RESPONSE transaction for a network of two sensors. Note that the sensors adopt the message termination and checksum requirement of the POLL command. In this case a full checksum is required but if the simple message termination of a CRLF was used instead the checksums would not be generated either by the Master or the sensors.

Sent from Master:

**<DC2>**

**SOH** <Read Weight> ***CRC*** **EOT**

**<DC4>**

Received at master:

**<DC2>**

**SOH** <Read Weight> ***CRC*** **EOT**

**SOH** <Sensor 1 Weight> ***CRC*** **EOT**

**SOH** <Sensor 2 Weight> ***CRC*** **EOT**

**<DC4>**

Note that the Master receives its original poll command back along with the responses from all addressed sensors within the **<DC2> <DC4>** framing.

Examples

| | Description |
|---|---|
| **Read Gross Weight ( Read Final)**<br>**COMMAND:**<br>« 21110026₈ »<br>**RESPONSE:**<br>« 21110026₈<br>81110026:00000064₈ » | **COMMAND:**<br>Read Gross Weight (Register 0026):<br>ADDR = $21_H$ : Reply required only from sensor 1<br>CMD = $11_H$ : Read Final<br>REG = $0026_H$ : Gross Weight<br>**RESPONSE:**<br>Response is from instrument #1 which currently has a Gross weight of $64_H$ = 100 kg. |
| **Read Gross Weight (Read Literal)**<br>**COMMAND:**<br>« 21050026₈ »<br>**RESPONSE:**<br>« 21050026₈<br>81050026: 100 kg G₈ » | **COMMAND:**<br>Read Gross Weight (Register $0026_H$):<br>ADDR = $21_H$ : Reply required only from sensor 1<br>CMD = $05_H$ : Read Literal<br>REG = $0026_H$ : Gross Weight<br>**RESPONSE:**<br>Same response from instrument #1 but in literal format. |
| **Read Gross Weight (Read Final)**<br>**COMMAND:**<br>«20050026₈ »<br>**RESPONSE:**<br>« 21050026₈<br>81050026: 100 kg G₈<br>82050026: 125 kg G₈ » | **COMMAND:**<br>Read Gross Weight (Register $0026_H$):<br>ADDR = $20_H$ : Reply required from all sensors<br>CMD = $05_H$ : Read Literal<br>REG = $0026_H$ : Gross Weight<br>**RESPONSE:**<br>Same response from instrument #1 but now instrument #2 answers with a weight of 125 kg as well. |

where

« is **<DC2>**

8 is message termination **CRLF or ';'**

» is **<DC4>**