FlyWire

FireFly Development Specifications and Communication Protocol

This document describes the communication protocol between a FireFly and an external sensor device, hereafter referred to as Master and Slave. The protocol is based on the standard Open Systems Interconnection (OSI) model. Only several layers of this model are implemented, due to the plain one-to-one connection between the Master and the Slave.

Additional information on how to design a custom sensor device and implement the FlyWire communication protocol can be found in chapter 5.

The following topics will be addressed in this document:

1. Physical Layer (hardware)
2. Data Link Layer (frame)
3. Payload description
4. Command list
5. Custom sensor device
6. EEPROM Map
7. List of supported devices
8. Glossary

1. Physical Layer

The Physical Layer describes the transmission and reception of raw bit streams over a physical medium.
This layer is implemented as a wired connection between the Master and the Slave. The FireFly has a female M12 X-coded connector for communicating with an external Slave. Figure 1 shows the pin assignment of the connector on the FireFly. An external Slave may include multiple sensors, but only one external Slave can be connected to a Master at a time.

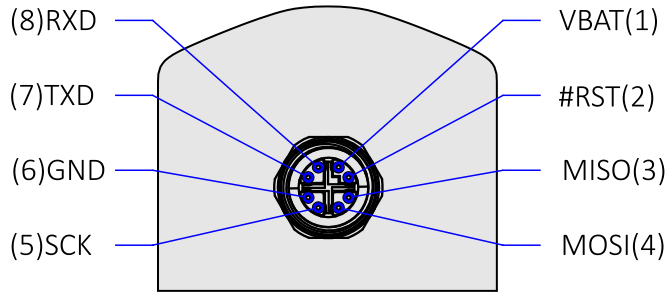

Figure 1: Front view of the FireFly with the M12 X-coded connector.

1.1 Communication

The raw data is sent by using a Universal Asynchronous Receiver Transmitter (UART). A hardware UART peripheral is implemented in the microcontroller of the FireFly. A Slave may contain a hardware UART peripheral, but a software 'bit-bang' UART can be used if timing specifications are met.

Two signals and a common ground are used for transmission and reception, i.e. RXD for the receiving and TXD for transmitting. Other signals that are commonly used for flow control, like RTS, CTS, DSR and DTR are not used. Interrupt-driven communication with proper data buffering should be adequate to omit these flow control signals.

Communication is half-duplex. The Master initiates the communication and the Slave will only respond after a successful reception of a frame (see Network layer).

The RXD and TXD signals are cross-linked between the Master and the Slave and driven at 3 V CMOS levels, as depicted in Figure 2. Idle state (marking bits) is at a 3 V level. The Slave should have enough drive strength to transmit data reliably over a 2 meter long sensor cable (AWG22, CAT6).

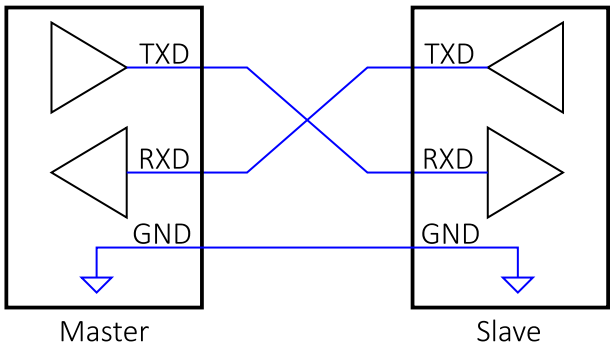

Figure 2: Connecting Master and Slave devices using UART.

The communication speed between the Master and Slave device is 19200 baud. Every character is sent with 1 start bit ('0'), 8 data bits and 1 stop bit ('1'). The data bits are shifted out with the LSB (Least Significant Bit) first. These settings are commonly referred to as 19200, 8, N, 1. Figure 3 shows a timing diagram of the UART communication.

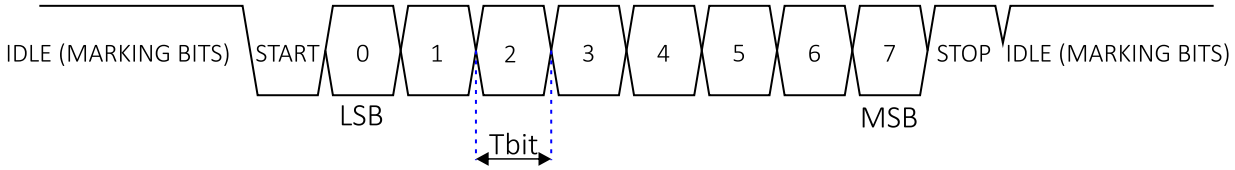

Figure 3: UART timing diagram.

One bit time (Tbit) is 1 / 19200 ≈ 52 µS.
No horizontal parity is used since vertical parity is implemented in the Data Link Layer.

1.2 Power and Reset

An additional signal is available on the M12 connector. This is the reset (#RST) signal of the FireFly. The reset can be activated by holding a magnet close to the upper side of the enclosure of the FireFly, near the connector. The #RST signal is active low and when activated, the FireFly will re-initialize, blink its LEDs and it will try to join the LoRa network.

The #RST can be monitored by a Slave such that action can be taken if needed. An example would be a scale that tares itself after detecting a falling edge on the #RST line.

The #RST can also be driven by a Slave. Note that the signal should only be pulled low, as it is pulled high by a resistor inside the FireFly. If the external device uses a push-pull output instead of an open-drain output, the FireFly cannot be reset by using a magnet.

The Slave can be powered by the Master from its main battery. Since the battery voltage is between 4.2 V (fully charged) and 3.1 V (end voltage), the Slave should have it's own 3 V regulator to be able to drive its TXD signal at the appropriate 3V level. See Section 5 for further information on power consumption requirements.

2. Data Link Layer

The Data Link Layer detects errors that may occur in the physical layer. It defines a mechanism to establish and terminate a connection between two physically connected devices. It also defines the protocol for flow control between them.

2.1 Waking up the slave

In normal operation, both the FireFly (Master) and the sensor device (Slave) are in sleep mode to reduce power consumption. At regular intervals the Master will wake up and measure its internal sensors. To get the additional data from the Slave, the Master needs to wake up the Slave before starting to query.

Waking up the Slave is done by sending a Break character. A Break character consists of a start bit, followed by at least 8 data bits which are set to zero, and an (inverted) stop bit ('0'), as shown in Figure 4. This is a violation of a normal byte received by a UART, since a stop bit must be a logic high, and is therefore considered a Break condition. To avoid character errors or character fragments during a wake-up event, the wake-up data bits must all be zeros. The number of data bits can be increased as long as their value is zero ('0'). 13 data bits are recommended, similar to a LIN-bus break condition.

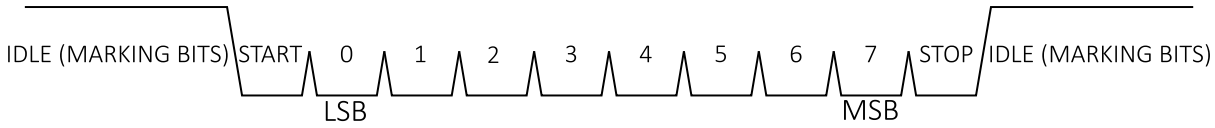

Figure 4: Minimum break character required to wake-up the Slave device.

For a software bit-bang UART the falling edge of the RXD signal can be used to wake up a microcontroller. On microcontrollers with a UART peripheral, the break feature is often implemented in hardware.

The time it takes to wake up depends on how wake-up from sleep is implemented in the Slave's microcontroller. After sending a break character a FireFly will wait for 2 ms before sending any commands to allow the Slave to wake up. Since crystal oscillators have a certain Oscillator Start-up Time (OST) some microcontrollers will wait for several Oscillator periods (Tosc) before executing code. Please consult the datasheet of the microcontroller and timing device to determine the required time-out to account for this ring-up as to guarantee proper timing.

2.2 Frame format

The frame format is identical for both the Master and the Slave. The frame consists of 4 sections, as shows in Figure 5.

1. Start byte (STF; start of frame)
2. Frame length (LOF; length of frame)
3. Payload (PYL)
4. Checksum (CHK; XOR checksum).

All bytes in a frame are in binary format. The start byte, frame length and checksum are single bytes. Only the payload can vary in size. The total length of the frame should be less than or equal to 128 bytes.

The start byte is 0x5A (hexadecimal) or 90 (decimal) and corresponds with ASCII “Z”.
The LOF is the total number of bytes in the frame, including the start, length and checksum byte.

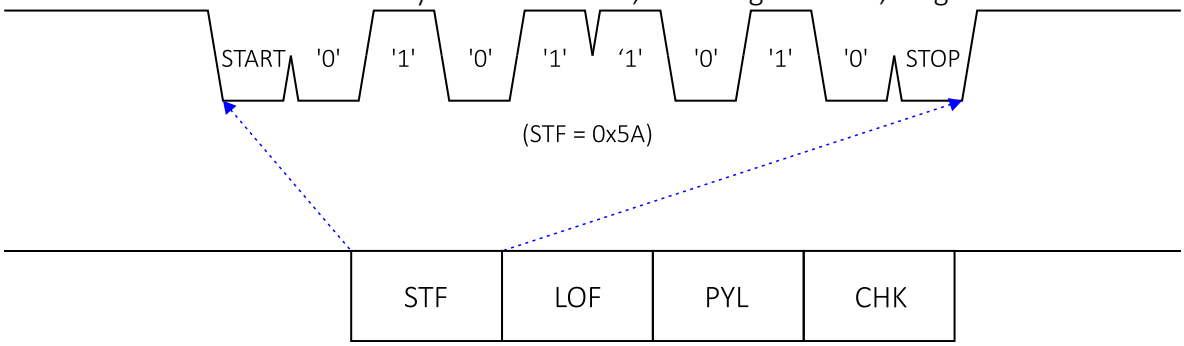

Figure 5: Frame format showing the four sections.

The CHK is a single byte that represents checksum of the frame. It is calculated by bit-wise XOR-ing (^) all bytes in the frame, excluding itself. Below is an example of a short frame, where the STF = 0x5A, the length of the frame is 0x04 and the actual payload is 0xF1.

```
STF = 0x5A =        0b01011010
LOF = 0x04 =        0b00000100
PYL = 0xF1 =        0b11110001
                    ----------
CHK = 0xAF =        0b10101111
```

2.3 Timing

Bytes within a frame should be sent head-to-tail with a maximum inter-byte delay of 10 milliseconds. This equates to roughly 20 bytes at 19200 baud including start and stop bits. This relaxation allows for other interrupts in the system to be serviced.

The communication is initiated by the Master by sending a Break character. After the connection has been established, successive commands can be sent by the Master. If the Slave receives a valid frame, the Slave should reply within 500 milliseconds. This allows for slow AD conversions of dual slope converters or successive approximation ADCs with a slow settling rate.

If the Slave receives a partial frame or a frame with an invalid checksum, the Slave must not respond but it should flush its buffers and wait for a new STF.

If the Master does not get a reply, the Master can re-transmit the frame after the inter-frame delay of 500 milliseconds, as mentioned above. No delay is needed for the Master to transmit a new frame after a valid frame has been received from the Slave.

No additional break characters should be sent after the Slave has woken up. To get the product ID and serial number the correct command should be sent (see for Section 4 for more details). Due to the simplicity of the response to this command, a reply is expected within 50 ms. This shortened delay allows a FireFly to determine more rapidly whether an external device is connected.

The communication is terminated by the Master by sending a Hibernate command. If no Hibernate command is issued the Slave should go back to its low power state if the bus is idle (marking bits) for more than 1000 ms.

3. Payload description

Within a frame, the payload is the actual data that is to be transmitted. The payload is transmitted in binary format.

The Master can send simple commands or extended commands with additional data. Simple commands only contain a command byte (CMD) in the payload and are used for requesting information like sensor data, product ID, production date or a serial number.

Extended commands are used to transfer additional data to and from the Slave. Extended commands have additional data bytes after the CMD byte. The additional data can therefore hold a pointer to a memory address, the length of the data and the data itself, in case of a Write EEPROM command.
Other examples are calibration parameters, calibration date, product ID or serial number.

The Slave will reply with a simple ACK or NAK (Acknowledge (0x06) or No-Acknowledge (0x15)), followed optionally by the data itself.

If a Slave receives a valid frame, but the command is not supported, the Slave must reply with a NAK. No reply should be sent when an invalid frame is received.

Example of a simple command issued by the Master:

A simple command will have only one command byte in the payload.
STF | LOF | CMD | CHK

Slave reply to a simple command:

In case the command is supported and the data is valid, the Slave will reply with an ACK.
Additional data bytes may follow, depending on the command.
STF | LOF | ACK | (DATA) | (DATA) | … | CHK
In case of an error or unsupported command, the Slave will issue a NAK.
STF | LOF | NAK | CHK

Example of an extended command issued by the Master:

An extended command will hold one or more additional data bytes after the command byte
STF | LOF | CMD | DATA | DATA | … | CHK

Slave reply to an extended command:

In case the command is supported and the data is valid, the Slave will reply with an ACK.
Additional data bytes may follow, depending on the command.
STF | LOF | ACK | (DATA) | (DATA) | … | CHK
In case of an error or unsupported command, the Slave will issue a NAK.
STF | LOF | NAK | CHK

4. Command list

To comply with the FlyWire protocol some commands are obligatory to be implemented and some are optional.

Optional commands can be omitted for Slaves that don't have internal or external EEPROM, or require calibration data. An example is a Slave with only a relay that does not require any calibration. If a Slave receives an unsupported command, it must reply with a NAK.

Standard commands range from 0x01 to 0x6F. Optional commands range from 0x70 to 0xDF. Factory or test commands range from 0xE1 to 0xEF.

Custom commands are not specified in this document and can be used by a third party manufacturer to implement any commands that are not listed here. As an example a command could be implemented to access EEPROM above the 255 (0xFF) address limit of the standard Read EEPROM command.

Standard commands:

0x00 Hibernate
0x01 GetID
0x02 Request Sensor Values
0x03 Reboot

Optional commands:

0x70 Read EEPROM
0x71 Write EEROM
0x72 Set Values

Factory/test commands:
0xE0 Stay Awake

Custom commands:
0xF0 Custom commands

Hibernate:

0x5A | 0x04 | 0x00 | CHK
This command will put the device to its low power sleep mode. It is used to prevent unnecessary power consumption.
In normal mode the Slave will go to sleep after 1 second of no communication.

The Slave will respond with an ACK:
STF | LOF | ACK | CHK

GetID:

0x5A | 0x04 | 0x01 | CHK
This command will get both the Product ID (PID) and Serial Number (SN) from the Slave.
The product ID is used to identify the type of accessory that is plugged into the FireFly. A list of supported PIDs is maintained by Quantified. A list of currently supported devices is shown in chapter 6.

The Slave will respond with:
STF | LOF | ACK | PID | SN(high byte) | SN(low byte) | CHK
Both the SN bytes make up an unsigned 16 bit integer indicating the serial number of the device. This number can be hard coded in firmware, but storing it in EEPROM is recommended. See chapter 6 for the recommended EEPROM map.

Get Sensor Values:

0x5A | 0x04 | 0x02 | CHK
This command will request the current sensor values.

The Slave will respond with:
STF | LOF | ACK | DATA | … | CHK
The response will have one or more data bytes depending on the number of sensors in the Slave and the resolution of each of these sensors. The FireFly sends the data bytes in a payload and does not interpret them.
The format of the data bytes must be known by the dashboard to be able to interpret the payload as to show the actual values.

Reboot:

0x5A | 0x04 | 0x03 | CHK
This command will reboot the Slave, similar to a power-up. Any changes made, like the Stay Awake feature, should be reset. This command must be issued after recalibration, if the Slave is using calibration data from EEPROM.

The Slave will respond with an ACK before performing the reboot:
STF | LOF | ACK | CHK

Read EEPROM:

0x5A | 0x06 | 0x70 | ADDRESS | LENGTH | CHK
This command will read multiple bytes from EEPROM starting at location (ADDRESS). The number of bytes being read is (LENGTH) and must be limited to 64. Note that this potentially could lead to reading
bytes beyond the 255 address limit. The Slave should fill these bytes with dummy data. See chapter 5 about using the memory map.

The Slave will respond with:
STF | LOF | ACK | DATA | … | CHK
The response will have one or more data bytes depending on the number of memory locations requested.

Write EEROM:

0x5A | 0x06 | 0x71 | ADDRESS | DATA | CHK
This command will write a single byte to EEPROM at location (ADDRESS). Only a single byte can be programmed as some EEPROMs take some time to perform an erase/write cycle on a memory cell. Writing multiple bytes could conflict with the FlyWire timing.

The Slave will reply with an ACK or a NAK:
STF | LOF | ACK | CHK or
STF | LOF | NAK | CHK

Set Values:

0x5A | LOF | 0x72 | DATA | … | CHK
This command will set values in the Slave. Similar to the Get Sensor Values, the FireFly has no knowledge about the content or format of these data bytes. This command could be used for instance to set or reset a relay in a Slave.

The Slave will reply with an ACK or a NAK:
STF | LOF | ACK | CHK or
STF | LOF | NAK | CHK

Stay Awake:

0x5A | 0x04 | 0xE0 | CHK
This command will override the 1 second auto sleep feature of the Slave. After issuing this command the Slave should stay awake until a power cycle occurs, a Reboot or a Hibernate command. This command is useful during testing without having to send a break each time to wake-up the device.

The Slave will reply with an ACK or a NAK:
STF | LOF | ACK | CHK or
STF | LOF | NAK | CHK

5. Custom sensor device

To design a custom Slave device for a FireFly several key features must be taken into consideration. The FireFly is a battery-powered, LoRa enabled device. A low-power design with a sleep mode is crucial to ensure an operational time of several months without recharging is not compromised.

5.1 General design

A single slave device is not limited to a single sensor or actuator, and may contain multiples of each.

It is recommended that the slave has at least have 256 bytes of EEPROM for storing calibration parameters, calibration date, product ID and serial number. When no EEPROM is present, some of these values may be hard-coded and stored in program memory (flash).

5.2 Power considerations

A custom sensor can be powered by the FireFly through the M12 X-coded connector. The battery inside the FireFly is a Li-ion battery with a rated capacity of 3000 mAh. The output is fused and a recommended maximum current draw is 50 mA during operation. If more current or a higher voltage is needed, it is recommended to equip the Slave with its own power source. Current draw while in sleep mode must be in the µA range when powering from the FireFly battery. See Section 2 on how to enter and exit sleep mode. The voltage on the TXD and RXD signals should remain at 3 V levels.

The FireFly battery voltage varies between 4.2 V when fully charged and 3.1 V when nearly empty. A low drop-out regulator can be used to power the main processor and components and the optional TXD/RXD line drivers. Line drivers may be required if the processor's TXD/RXD signals have low drive strength and are equipped with 100Ω series resistors. A 74LVC1G17 or similar with 24 mA or higher drive strength is recommended.

If a higher voltage is needed, an inductive boost circuit or charge pump can be used. Take into account that these usually have a low efficiency and should be turned off when the Slave is in sleep mode. Also note that the settling time of a voltage booster may interfere with the timing requirements of the FlyWire communication protocol.

5.3 Timing

Communication is asynchronous, and therefore the total UART timing inaccuracy should be within 3.5 %. The FireFly is equipped with a resonator with an accuracy of 0.5 % over temperature, so the maximum allowable error for a Slave should be well below 3 % for reliable communication. A low drift resonator or crystal is recommended. To further minimize errors, a clock frequency should be chosen that is a multiple of the 19200 baud rate. Examples are 1.8432 MHz, 3.6864 MHz or 7.3728 MHz or above. These frequencies are available for crystals and resonators.
Internal RC oscillators may be used if the microcontoller's specifications are within these limits.

6. EEPROM Map

The use of an EEPROM is recommended to be compatible with the FlyWire testing tool. This can be an internal microcontroller EEPROM or an external one. The EEPROM can be used to store the PID and SN instead of hard-coding it in firmware. It also allows the Slave to be reconfigured or recalibrated after the firmware has been uploaded, as detailed in the Read/Write EEPROM command in Section 4.

A minimum of 256 bytes is required. A larger EEPROM can be used, but cannot be accessed with the Read/Write EEPROM commands due to the 8 bit memory pointer. Any memory above 255 (0xFF) can be used to hold critical or confidential information.

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	R	E	S	E	R	V	E	D								
0x10	D	E	V	I	C	E	N	A	M	E	0x00	0x00	0x00	0x00	0x00	0x00
0x20	PID	SN high	SN low	PDD	PMM	PYY	PYY	CDD	CMM	CYY	CYY					
0x30																
0x40																
0x50																
0x60																
0x70																
0x80	C	A	L	I	B	R	A	T	I	O	N		D	A	T	A
0x90																
0xA0																
0xB0																
0xC0																
0xD0																
0xE0																
0xF0																

0x00 – 0x0F Reserved

0x10 – 0x1F These locations hold the name of the device in ASCII. Unused locations at the end should be padded with 0x00

0x20 – 0x22 Product ID and Serial Number. The SN consists of two bytes forming an unsigned integer. High byte at 0x21, low byte at 0x22.

0x23 – 0x26 Production date in DD/MM/YY/YY format. The year is split in two bytes. E.g. 2020 is stored as 20, 20 or 0x14, 0x14 in hex format.

0x27 – 0x2A Calibration date. Format is similar to the production date.

0x2F Location 0x2F is used for debugging. If its value is non-zero, the on-board LED will be lit as long as the Solar Chimney is active, and switch off in hibernation. When set to zero, the LED will always be off to preserve power.

0x80 – 0xFF This area is reserved for calibration data and is device dependent.

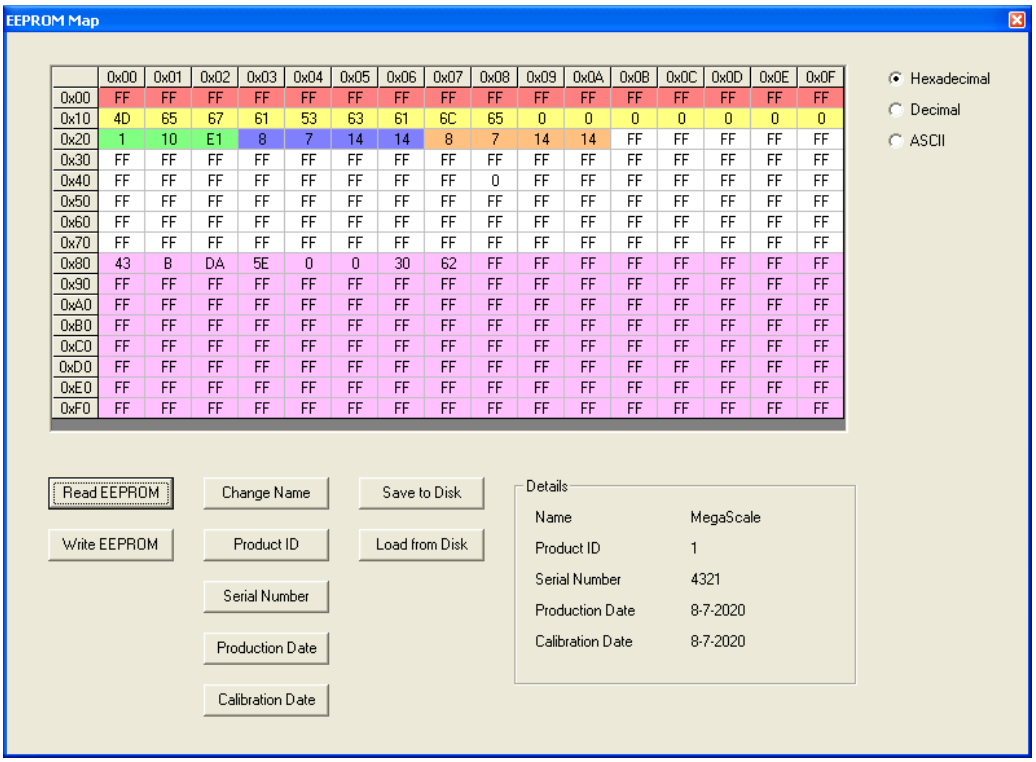

Figure 6: Example snapshot of an EEPROM memory map

7. List of supported devices

The list below shows the currently supported devices. This list will be updated when new devices become available. Custom devices can be added by requesting a PID from Quantified.

PID	Name	Company	Description
0x00	*Reserved	Quantified	Test Slave
0x01	Mega Scale	Quantified	Plant weighing scale
0x02	Chimney	Quantified	RH and temperature sensor
0x03	LWS	Quantified	Leaf wetness sensor
0x04	Pluviometer L	Quantified	Rain sensor
0x05	Probe TRH	Quantified	Probe with temperature and RH sensor
0x06	QSS	Quantified	Spectral sensor
0x07	Pluviometer S	Quantified	Rain sensor
0x80	Your device	Your company	Your description

8. Glossary

ASCII	American Standard Code for Information Interchange
ACK	Acknowledge. 0x06.
Baud	Communication speed in bits per second.
CHK	Checksum. XOR of all bytes in a frame, excluding the CHK byte itself.
EEPROM	Electrically Erasable Programmable Read Only Memory.
NAK	No acknowledge. 0x15.
STF	Start of frame. First byte of the frame sent by both the Master and the Slave.
LOF	Length of frame. Total amount of bytes in a frame.
OSI	Open Systems Interconnection.
PYL	Payload. The actual data being sent.
UART	Universal Asynchronous Receiver Transmitter.
TXD	Transmit data signal.
RXD	Receive data signal.