APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

Custom RS-485 Communication Protocol

DOCUMENT DATE:	**23th APRIL 2019**
DOCUMENT REVISION:	**1.1**

W W W . 4 D S Y S T E M S . C O M . A U

Description

This document is intended to provide fundamental information on how to create a custom protocol for data communication between devices that are connected to a half-duplex RS-485 bus.

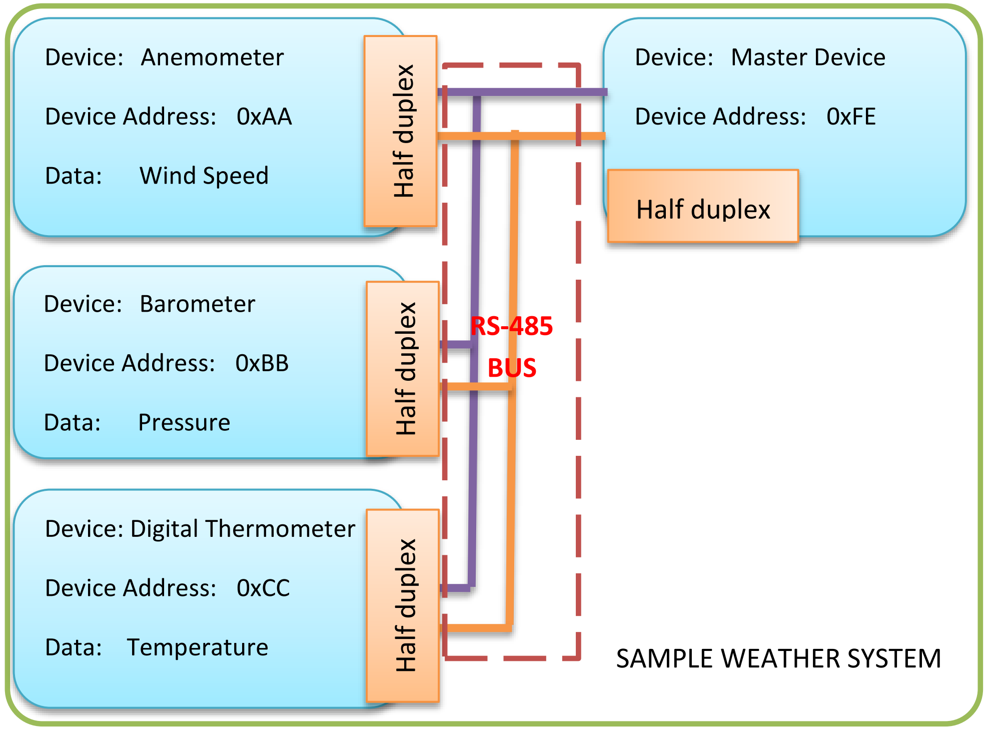
By definition a half-duplex is a system in which one or more transmitters can communicate with one or more receivers with only one transmitter being active at any one time.

For this application note, it is advised to read through the document to better understand the basics of creating a custom protocol.

Content

Application Overview

For a simplified presentation of how to create a user defined data communication that utilizes a half-duplex RS-485, the system shown below will be the reference system for this application note.

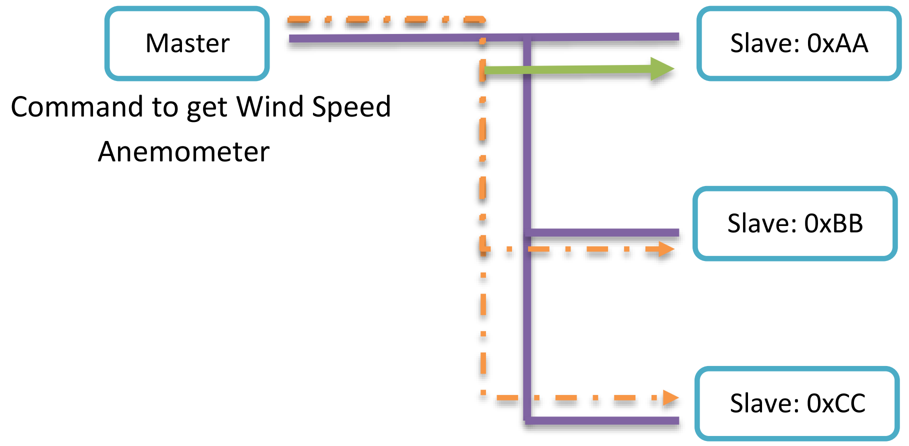


This application note does not include any particular codes that could be related to the data communication being described and explained.

The Data Communication Model

Command Transmission from the Master

The figure show below depicts the simplified flow of communication. It shows how a command from the master is sent to the slave devices and that only one device accepts the command message.



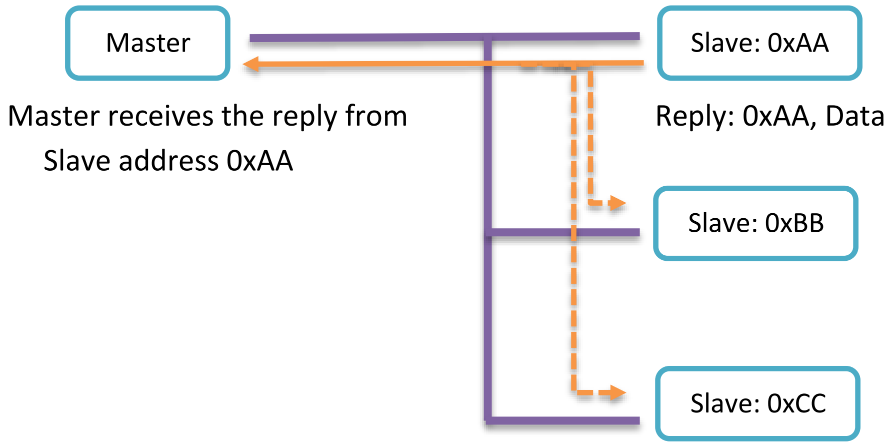
When the master device transmits data, all the slave devices on the bus receives the same data at any point. The command sent on the figure above will only be meaningful to the slave device specified.

The green arrow shows that device address 0xAA, which is the anemometer, has accepted the command that requires it to give the Wind Speed. The reply will be presented in the following section of this application note.

**It is therefore important that the data being sent to the bus must target a particular slave DEVICE ADDRESS and the COMMAND to be executed.

Reply Transmission from the Slave Device

Following the successful transmission of the command from the master, the slave will now proceed to transmitting its reply message. The slave address will transmit its reply through the RS-485 bus. The solid orange coloured arrow depicts the success of transmission.



Notice from the figure the broken line arrows. The broken line arrows represent the transmission of the reply message to all the devices in the bus. This means that only the master or other devices that can make sense of the information are able to use this information.

Furthermore more, focusing on the reply data, we can see that the slave device 0xAA sent out information that contains its address and the data being requested. The addition of the slave address helps assure the correctness of the origin of the data received by the master.

** The reply message of the slave device contained the SLAVE ADDRESS and the DATA requested.

Message formatting

Formatting the Command messages

To further discuss the command messages sent by master, the following section will now present a more specific manner of creating the command message or command packet.

First is to consider the DATA variables that are needed, these are – temperature, Wind Speed, and Pressure. Consider the following assignments.

Commands:

Read	= ’R’

Data Variables:

Temperature	= ’T’
Wind Speed	= ‘S’
Pressure	= ‘P’

At this point the messages is composed of the following;

Slave Address	Command	Data Requested

Error during transmission are sometimes inevitable which may be caused by program bugs or even hardware problems. In this regard, another information can be added to the command message. A CHECKSUM can be added at the end of the message. The checksum can be used by the slave device to counter check the command message received. Checksum is the result of continually using XOR through the values in the command message.

Checksum = SlaveAddress ^ Command ^ DataRequested

As a result, we have the following values command message.

Slave Address	Command	Data Requested	Checksum

On the other hand another problem arises, this is the question of how to detect the start of the message. If consecutive messages are sent over to the RS-485 bus then detection of the start point will be a problem. This can be resolved with the use of a **QUALIFIER**. The qualifier is a character that denotes the start of a message.

QUALIFIER	Slave Address	Command	Data Requested	Checksum

Where:

Checksum = **QUALIFIER ^** SlaveAddress ^ Command ^ DataRequested

At this point we have a reliable command message. Adding the length of the command message is also a very helpful means of detecting the end of the command message.

QUALIFIER	Length	Slave Address	Command	Data Requested	Checksum

Where:

Checksum =**QUALIFIER** **^**Length ^SlaveAddress ^Command ^DataRequested

To ensure correctness of data in the custom protocol , user can make use of CRCs or other means of providing counter check to the receiving slave device. Although these counter-checking means are useful it should be used with care. If the means of counter-checking the data is complicated then the system might be slowed down at some point during operation.

Formatting the Reply messages

On the other hand, the other devices on the RS-485 should also be able to reply to the command messages. The slave devices should also send their reply with the checksum so the master can counter check the message.

It should also have a qualifier that is known to the master device. The addition of the total length can also helps the master detect the end of the reply message. Considering all of the above we will arrive to a reply message that which can be formatted same as the figure below.

QUALIFIER	Length	Slave Address of the replying device	Data Requested	Checksum

Where:

Checksum =**QUALIFIER** **^**Length ^SlaveAddress ^DataRequested

It can be noticed that the reply message is almost the same as the command message. This is simplest manner of creating a user defined protocol but it allows the user to easily debug a problem related to the message for both the reply message and the command message. It also speeds up development time since users can add a fair large amount of commands and data in the messages.

Data communication sample

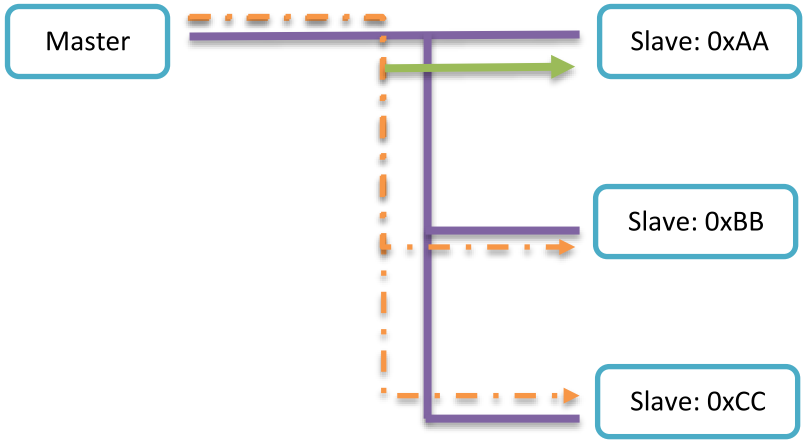
Let us consider example below to show how the message formats is used in the system. The Master request a certain slave device to send the wind speed value.

Master Message:

The masters sends a command to the Anemometer (0xAA) to READ the wind speed.

QUALIFIER	Length	Slave Address	Command	Data Requested	Checksum
'@' 0x40	6	0xAA	'R' 0x52	'S' 0x53	0xED

Checksum = 0x40 ^ 0x06 ^ 0xAA ^ 0x52 ^ 0x53



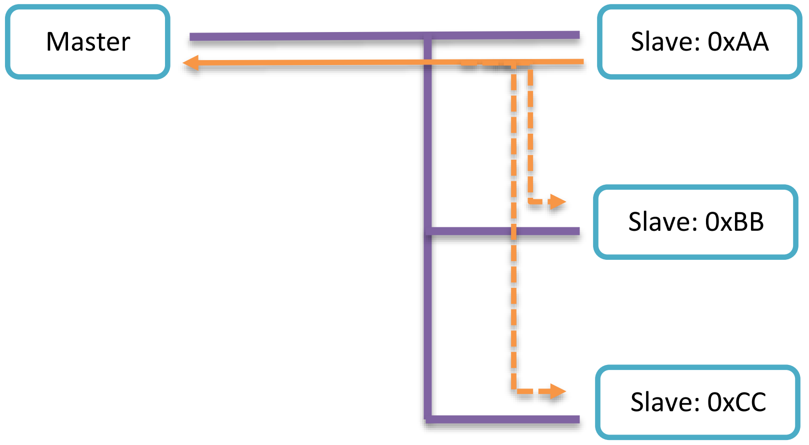
Slave Device Reply Message:

In this part let us consider the reply of slave device that detects the wind speed, the value being 120 Km/H.

Reply Message:

QUALIFIER	Length	Slave Address of the replying device	Data Requested	Checksum
'@' 0x40	4	Anemometer 0xAA	120 Km/H 0x78	0x96

Checksum = 0x40 ^ 0x04 ^ 0xAA ^ 0x52 ^ 0x78



** To counter check message validity master and slave device can simply re-compute the CHECKSUM and check if it is the same as the received checksum value.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability