

Using DMX1TR Modbus/TCP firmware option

This application note explains Modbus/TCP parameters for the KissBox DMX1TR interface in order to integrate them into a Modbus/TCP installation.

The Modbus/TCP firmware option turns the KissBox DMX1TR into a Modbus/DMX gateway, allowing any Modbus master (like Programmable Logic Controllers or any control software with Modbus/TCP support) to get access to DMX fixtures. The DMX1TR firmware provides full control over the 512 DMX output channels from Modbus, and provides images of incoming DMX values into Modbus input registers.

Firmware upgrade in DMX1TR

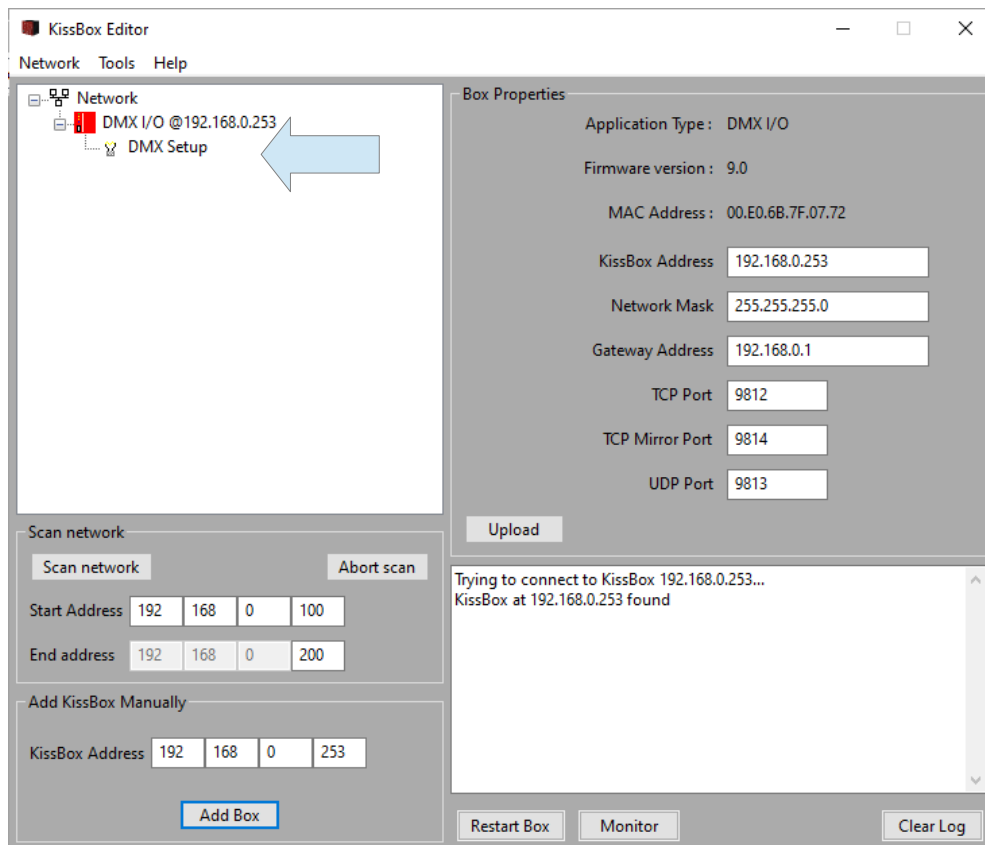
Modbus/TCP support is included in DMX1TR firmware version 9.0 and higher.

This firmware is compatible with CPU V4 and CPU V5 (please refer to label located under the KissBox to identify the CPU model)

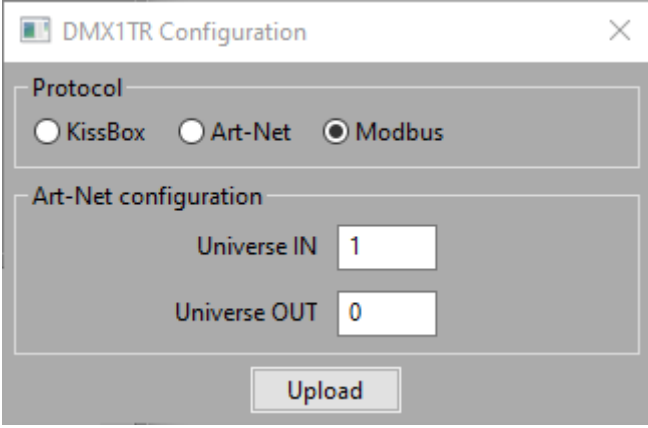
The DMX1TR firmware must be uploaded in the KissBox using KissBox Editor V15.0 minimum (previous version of the KissBox Editor can upload the firmware but they don't provide the option box to activate the Modbus/TCP mode. Please refer to KissBox Editor User's Manual for details about the procedure to load a new firmware if needed).

Firmware configuration

When the DMX1TR firmware is uploaded into the KissBox, connect to the KissBox using the KB Editor and double-click on DMX Setup item in the properties tree to open the DMX1TR configuration dialog.



Select the "Modbus" radio button and click on Upload. Once the DMX1TR is restarted, it will operate in Modbus/TCP mode.



The image shows a software dialog box titled "DMX1TR Configuration". It has a close button (X) in the top right corner. The dialog is divided into two main sections. The first section, labeled "Protocol", contains three radio buttons: "KissBox", "Art-Net", and "Modbus". The "Modbus" radio button is selected. The second section, labeled "Art-Net configuration", contains two input fields: "Universe IN" with the value "1" and "Universe OUT" with the value "0". At the bottom of the dialog is an "Upload" button.

Modbus/TCP objects vs. DMX data

Supported Modbus functions :

Read Holding Registers (function 0x03) : reads DMX IN channels 1..512 in registers 0 to 511

Read Input Registers (function 0x04) : same as Read Holding Registers

Write Single Register (function 0x06) : writes one DMX OUT channel from 1 to 512 (register addresses from 0 to 511)

Write Multiple Registers (function 0x10) : writes multiple DMX OUT channels

Note that Modbus protocol does not allow more than 123 registers to be written or read in the same command. To read or write the 512 DMX channels, it is necessary to split the DMX data into multiple Modbus commands of less than 123 channels.

Modbus/TCP network parameters

The DMX1TR supports both Modbus/TCP and Modbus/UDP.

UDP port : 502

TCP port : 502

Up to two TCP sessions can be opened in parallel in the same DMX1TR (so two Modbus masters can communicate with the same DMX1TR at the same time, allowing backup in case of failure of one of the masters for example)

NumberMaxOfClientTransaction : 1 (only one command is allowed in a TCP message)

NumberMaxOfServerTransaction : 2

Unit Identifier : 1

Typical response time : less than 1 millisecond

Automatic TCP disconnection by timeout

Modbus/TCP relies on TCP protocol sessions. Before any Modbus command can be sent to the KissBox, a TCP session must be opened by the Modbus/TCP client.

A client is expected to use "graceful" session termination once it does not need to communicate anymore with the KissBox DMX1TR. In most cases, this is done automatically by the Operating System of the client when the Modbus software closes the communication

socket.

However, it is possible that a client terminates a session "ungracefully". This happens for example when the Modbus application crashes or if the computer running the Modbus/TCP client is forced to power off. In that case, the session would remain opened forever in the KissBox DMX1TR, making the communication channel unavailable for any other Modbus master until the KissBox is reset.

To avoid this situation, the KissBox firmware implements a TCP communication timeout. Once a Modbus/TCP session is opened by a client, the KissBox must receive at least one Modbus/TCP command every minute. If no command is received for more than 60 seconds, the communication socket in the KissBox is closed automatically and becomes available again.

This disconnection system is totally transparent for most applications, since PLCs and other real-time systems poll constantly multiple times per second Modbus devices, ensuring the KissBox timeout to be reset.

In other cases, you have to make sure that your application performs at least one communication exchange every 60 seconds. An easy way is to perform a simple read operation on any slot of the KissBox at least a few times per minute. The read operation ensures that no output will be affected by the client if the application does not know what to send.