



RS-485 Networking: 4Discovery Range

DOCUMENT DATE: **21st September 2021**
DOCUMENT REVISION: **1.0**

Description

This Application Note demonstrates a simple multi-drop RS-485 network, with a single Master node, and up to 254 slave nodes.

This demo is set up to use either 2, 3 or 4 different 4Discovery modules connected to a single network. The demo is set to work in a few different combinations. Please refer to the Application Overview section.

No micro-SD card is required for this demo, as all widgets used in the demo are utilising basic built-in functions. This is just to convey and focus on the communications, rather than to make the demo ‘pretty’.

This demo has been written with the 4Discovery range of products in mind, however it could be also used with the MOTG-RS485 module on gen4-uLCD products too, with some tweaks.

Visit <https://www.4dsystems.com.au/products> to see the latest display module products.

[Workshop4 IDE](#) (installed according to the installation document)

Compatible [4Discovery](#) modules are required (See Application Overview), along with a 485 Programmer for the 4Discovery, in order to load the demo.

Description 2

Application Overview 3

Setup Procedure 3

Configuring the Demo 3

Demo Modes 4

Screens of Demo 5

Network Overview 7

Network speed..... 7

Nodes..... 7

Points..... 7

Addressing..... 7

Topology..... 8

Frames..... 8

Frame format 8

Octet format..... 9

Frame types..... 9

Transactions 10

System boot..... 10

Frame tracing..... 11

Adding a new node to a system 12

Proprietary Information 13

Disclaimer of Warranties & Limitation of Liability 13

Application Overview

Please take note of the requirements for each of the roles of devices for this demo.

- 4Discovery-50, 4Discovery-35 and 4Discovery-13 compatible
- **Master** must be either 4Discovery-50 or 4Discovery-35
- **Slave1** can be either 4Discovery-50 or 4Discovery-35
- **Slave2** can be either 4Discovery-50 or 4Discovery-35
- **Slave3** can be any 4Discovery-50, 4Discovery-35, or 4Discovery-13

The combination of modules connected, for the demo to work correctly are as follows:

- 2 Devices. Master and Slave1, **OR** Master and Slave2, **OR** Master and Slave3
- 3 Devices. Master, Slave1 and Slave2, **OR** Master, Slave1 and Slave3, **OR** Master, Slave2, Slave3
- 4 Devices. Master, Slave1, Slave2, and Slave3.

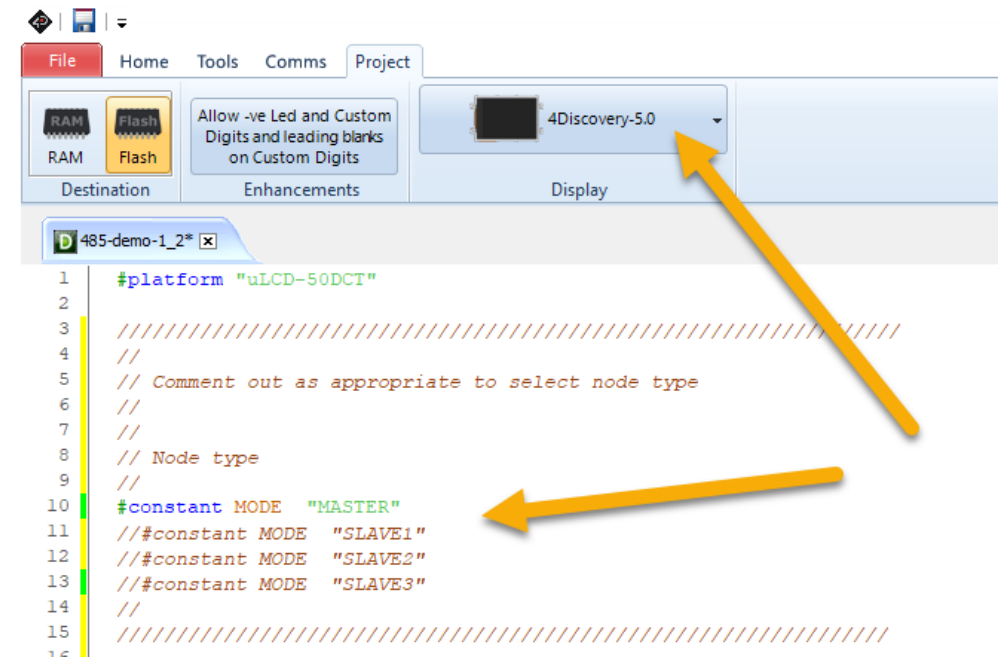
The objective of this application note is to acquaint the user on the procedure of setting up RS-485 networking, using 2 or more 4Discovery modules.

This demo can be used as a base for creating a project using the 4Discovery modules, using either Designer or ViSi environments in Workshop4. This is not designed for ViSi-Genie given the customised code requirements; however, it could be achieved using ViSi-Genie with Workshop4 PRO using Magic code. This has not been tested nor covers that topic. It is more suited for Designer or ViSi.

Setup Procedure

Configuring the Demo

The demo is a single piece of code written in the Workshop4 Designer environment and has commented lines of code to change the role the code will play.



By uncommenting the #constant MODE lines relating to Master, Slave1, Slave2, or Slave3, will change the code so workshop4 will compile the code to suit that target device role/mode. Only uncomment 1 line at a time.

Be sure to change the Display selection the normal way, in the Project Tab, and select the correct 4Discovery module in question. 4Discovery-35 (4Discovery), 4Discovery-50, or 4Discovery-13 (coming soon). This will change the #platform line at the top of the code. Don't change this manually, it will be adjusted with the Project tab's Display selection.

Once the correct Display has been selected, the correct Mode line has been uncommented (only have 1 line uncommented!), connect the display module to your PC, and load the project to the module using the 4D RS485 Programmer hardware.

Repeat for the other modules you wish to connect for this demo (refer to Application Overview section), change the Display and Mode as required, and load the remaining modules.

Connect all the modules together on a common RS485 network, using standard CAT4/5/6/7 cables. Easily achieved with the 4Discovery-50 due to it having dual RJ45 ports on the back, which makes daisy chaining easy. If using 4Discovery-35 or 4Discovery-13, custom cables may be required to be made to achieve this, and also to provide power to the bus.

Note: It is recommended to use the VIN (7-30V) power for the bus power, rather than 5V. Also be sure to NOT have the 485 Programmer on the bus, as this will interfere with the RS485 operation.

Demo Modes

Each device will have its Mode printed at the top of the display (once loaded to the module), identifying it as Master, or one of the Slaves, along with its Node number, and the version of the Demo.

The role of **Master** mode is to send data to each of the Slaves or retrieve data from the Slaves and pass them to other slaves as required.

In this demo, the **Master** has a log which can be enabled or disabled with the simple coloured box (button) at the top of the screen. This shows all the transactions happening on the bus between the Master and its Slaves.

The **Master** will request data from Slave2 and send it to Slave1 and Slave3 if present, or it will increment the data itself when Slave2 is not present.

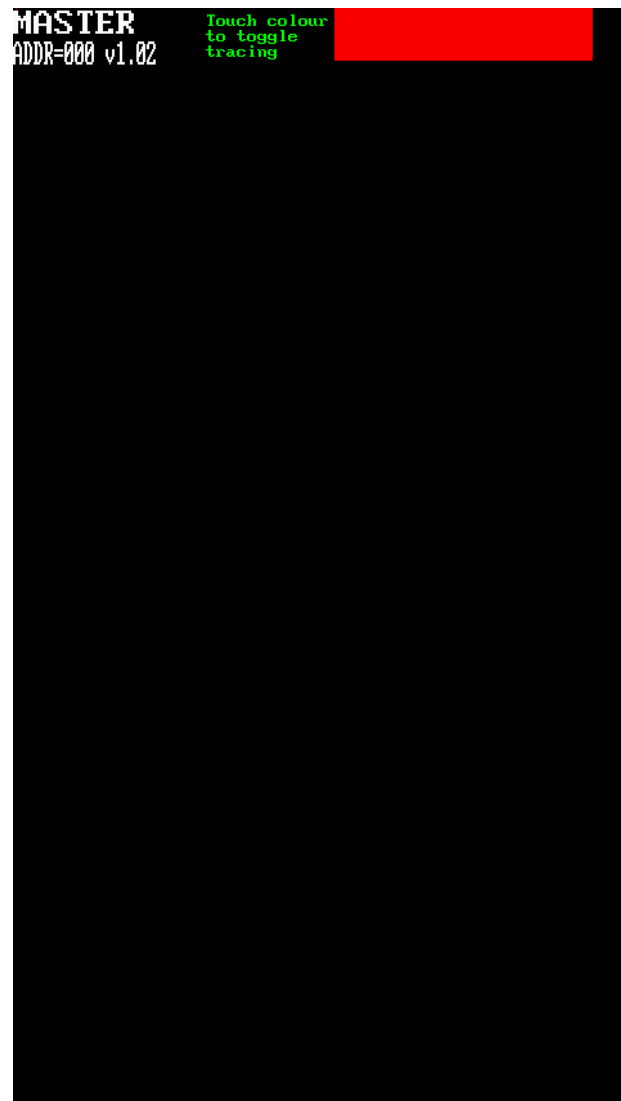
Slave1 has a 4-digit readout, which increments depending on what modules are present in the system. If Slave2 is not connected in the network, then the 4-digit readout will increment with data received from the Master (Master controls the increment). If Slave2 is present in the network, then the 4-digit readout will increment each time the button on Slave2 is pressed.

Slave2 has a button which when pressed will be picked up by the Master, and the Master will increment the 4-digit readout on Slave1 (if present) or the gauge on Slave3 (if present).

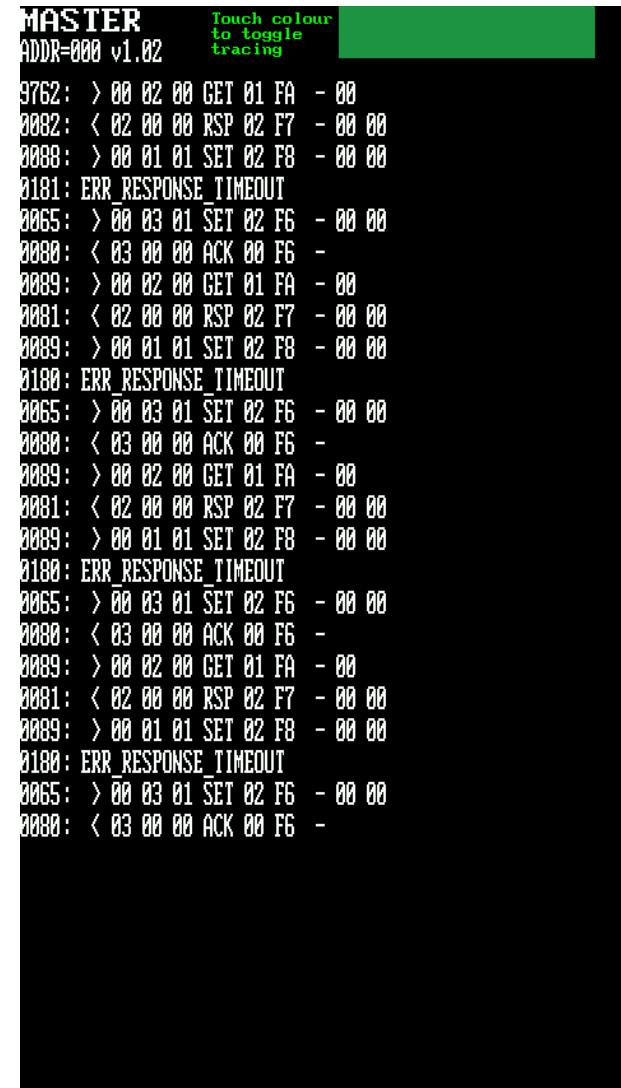
Slave3 has a full screen gauge on the screen, which is either updated by the Master if Slave2 is not present on the network, or by pressing the button on Slave2 when it is present on the network.

Screens of Demo

Master with Logging disabled (Default):



Master with Logging enabled. Showing GET comms to Slave2 and SET comms to Slave3, and Timeout Error to Slave1 after a SET (disconnected):



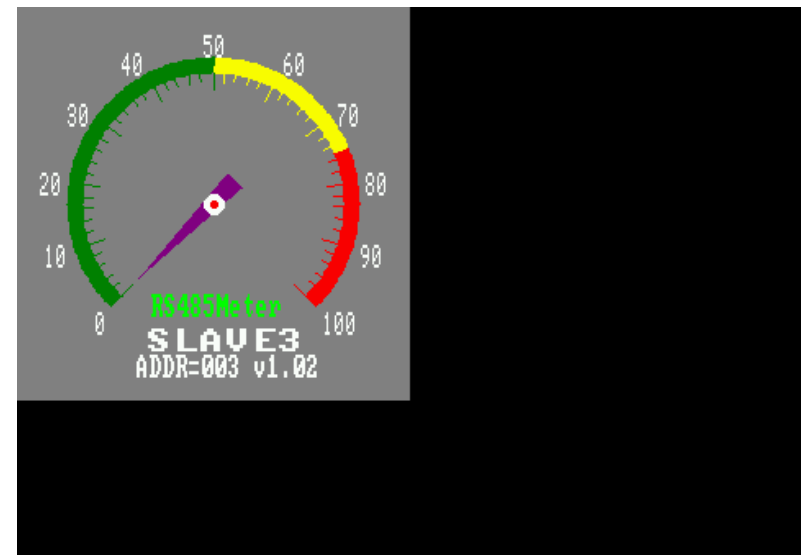
Slave1 showing some data received:



Slave2 showing its counter and button for setting Slave1 and Slave3:



Slave3 show its gauge with data received:



Depending on the size of 4Discovery being used, will determine what it looks like on those displays. The above screenshots are when using a 4Discovery-50.

Network Overview

Network speed

Depending on the hardware used, some are recommended to at a slower baud than others, as this is the recommendation based on the chipsets used. Check the revision of hardware vs the Datasheet to see the maximum. However, some versions of hardware are quite capable of working at transmission speeds up to 2187500 baud, depending on the cable run length. If using this example (or derivative thereof) in a finished installation between rooms of a building, or some distance, then it is recommended to stick to the recommended baud rates – however please test to determine stability. This demo is set to use a slow 9600 baud, to make things simple.

Depending on the actual node hardware in use, and what each node has to do to action a command, there is usually little point in running the network at high speed.

Nodes

A node is defined as any hardware device with a CPU that can implement this RS-485 protocol. At present this is only the 4Discovery-50, 4Discovery-35 and 4Discovery-13 displays. This could be extended to MOTG-RS485 with some modifications also and be made to work with the gen4-uLCD range.

Points

Points are addressable entities on a node. A point can be one of three things.
Physical IO — For example a digital output or ADC input.

Logical IO — For example the current time and date from an RTC, or the result of an on-board calculation such as watts derived from measured amps and volts.

Human interface — Anything on an LCD, LED or other style of display, or a human input device such as a touch screen or a push button.

The actual type of point doesn't matter much to the program sending data to a point, however of course not all data types are appropriate for all points. For example there's no point sending a “Hello world” string to an audio buzzer.

It's up to the receiving node to determine the suitability of the data and if necessary, respond with a NAK:ERR_BAD_DATA_TYPE frame.

Addressing

The address field of a frame is a single byte, therefore the network can in theory address up to 256 nodes and most modern RS-485 transceivers will accommodate that many loads.

There are however two reserved addresses, these being

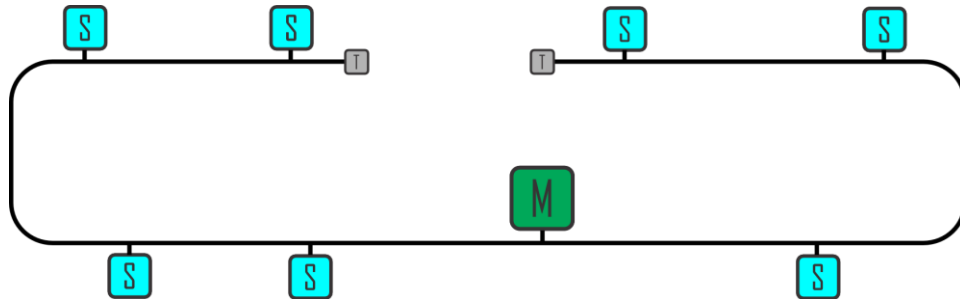
00 — Reserved for the master node.

FF — Reserved for broadcast frames.

Therefore, a maximum of 254 nodes can be addressed by this protocol.

Topology

Typically, a multi-drop RS-485 network must be terminated at both ends with 120R resistors and be a single inline wire with nodes tapping onto that wire.



Topology Example. M being Master, S being the Slaves, and T being the termination resistors at the 2 ends.

4Discovery displays allow for this termination by providing solder bridges that can be used to bring a 120R resistor into play.

Alternatively, there are many dongles available on the market to do the line termination.

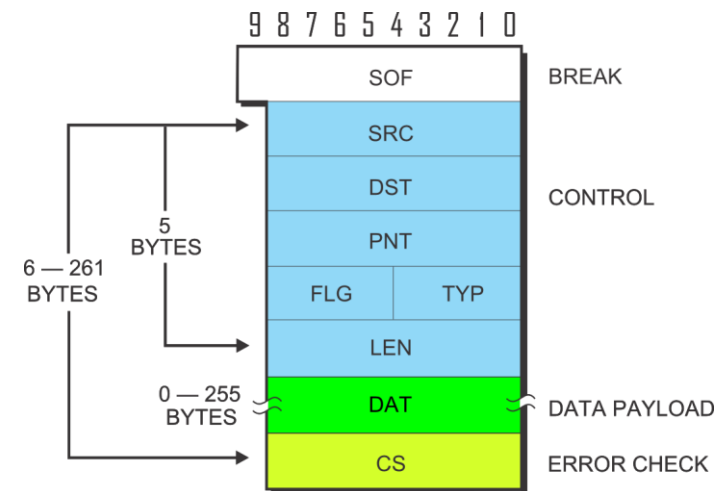
Even though ideally a network cable should be a single line, short stubs are often allowed. At slow speeds much latitude can be given as it may often be convenient to run a long stub in the field for practical reasons.

Frames

Frames are the basic data unit in this demo. Each frame holds addressing and error-detecting information as optionally some data.

Frame format

All frames have the following format.



The fields are as follows.

- SOF** — This field heralds the start of a frame (SOF), it is a string of a low logic level for at least 13-bit times, but often longer. This sequence is often referred to as a BREAK.
- SRC** — The source node, IE the one that is transmitting the frame.
- DST** — The destination node, the node to which the frame is addressed.
- PNT** — The point to which the data is intended to be written to or from which the data should be read from.

TYP — The type of the frame. The lower 4-bits of this octet are used to determine the frame type, up to 16 types are allowed.

FLG — The upper 4-bits of this octet will be used for various flags in future.

LEN — Length of the following data. A value of 0 is valid so the maximum data that can be transmitted in a frame is 255 bytes.

DATA — The data if any, up to 255 bytes.

CS — A simple add and 1's compliment checksum of all bytes in the frame not including the SOF field or this checksum.

All fields are binary so any digital value from 0x00 to 0xFF is valid in any part of any field.

Octet format

Each octet (byte) in a frame uses a standard 8N1 format, which is to say 8 data bits, no parity, and one stop bit.

Frame types

At present there are eight frame types defined for the TYP field.

NUL — 0x00
Not currently used but might be useful as a dead-man's frame to detect that a master has crashed. (0x00)

PNG — 0x01
(PiNG) Used to detect if a node exists on the network or not.

GET — 0x02
Used to get data from a point on a node.

SET — 0x03
Used to send data to a point on a node.

RSP — 0x04
(ReSPonse) The frame type used to respond to a GET frame.

SYS — 0x05
(SYStem) Used to send system commands to nodes. Currently the only system command is to change the bitrate, but in a future multi-master version this might be used to hand over control to another master.

ACK — 0x06
(ACKnowledge) Used to acknowledge a recently received frame. If the frame being replied to is a GET frame use an RSP frame type instead as the ACK frame does not allow for any data.

NAK — 0x0F
(Negative AcKnowledge) Used to inform the source node that there was a problem with the recently received frame. NAK is similar to ACK but it does allow a single data byte which is used to convey the error type.

Transactions

A transaction is the process of the master sending a frame and receiving a response in return.

With the exception of a broadcast frame, all frames must be acknowledged one way or another. Typically, this is with an ACK, NAK or RSP frame.

As mentioned, a broadcast frame does not comply with this because it may be handled by multiple nodes and they cannot all acknowledge at the same time.

System boot

The system boots up using a default bit rate of 9600bps. After initialising itself and allowing one second for all other nodes to initialise the master will send a SYS:SYSCOM_BITRATE packet with a new bitrate as defined by the BITRATE constant at the top of the project file. In the Demo this BITRATE value is set to 9600, the same as the default bitrate, so that nodes can easily be added on the fly for testing, also see later.

The master then waits another second then settles into a loop asking for data from one node and sending data to another. If tracing is enabled (rectangle at top of the screen is green, touch to toggle) frames will be displayed as they are sent or received.

Note: Printing to the screen takes some time so tracing will affect the net's overall throughput.

Note: This sequence only applies to a master with an address of 0x00 (the grand master).

Frame tracing

Frames can be traced on the master screen in a format similar to a data analyser. Tracing is turned on/off by touching the coloured rectangle at the top of the screen. The default is off (red).

Note: This tracing requires printing formatted text to the screen, this is a relatively slow process that (depending on the bit rate used) can vastly increase the inter-frame time.

As such tracing is useful to see traffic flow and generally debug logic problems, but it will affect any network timing.

The format of this frame trace dump is as follows. Note that this is almost the same as the actual frame format with the exception that the data is printed after the checksum, so all fields align on the display.

Four such frames will look something like this (not counting the first line which just labels the fields).

```

mS   IO S  D  P  T   L  CS   DATA
0083: > 00 02 00 GET 01 FA - 00
0079: < 02 00 00 RSP 02 E9 - 00 0E
0082: > 00 01 01 SET 02 BA - 00 0E
0079: < 01 00 00 ACK 00 FB -

```

mS — Time in milliseconds since the last frame was received or transmitted.

IO — The direction of the frame relative to the master.

> means frame was outgoing (Tx)

< means frame was incoming (Rx)

S — The source node, IE the one that is transmitting the frame.

D — The destination node to which the frame is addressed.

P — The point to which the data is intended to be written to, or from which the data is read from.

T — The frame type.

L — Length of the following data.

CS — A simple add and 1's compliment checksum of all bytes in the frame not including the SOF break or this checksum.

DATA — The data if any. Only the first 10 bytes of the data will be displayed.

In this example above we see the master doing a GET of point #0 on slave #2. Slave #2 replies with an RSP (response) frame holding the value of point #0 (the 7-seg display with a value of 14 or 0x0E) on that node.

The master then echoes that data in a SET packet to point #0 (another 7-segment display) on slave #1 and finally slave #1 sends an ACK (acknowledge) frame back to the master.

Adding a new node to a system

If a network is up and running at any bitrate other than the default rate (9600bps) a new node will not be able to communicate when it is added because it will be expecting comms to be at that default rate.

Therefore after adding the new node you need to either cycle the power to all nodes to invoke the boot sequence, or provide a mechanism on the master to change the bitrate under operator control. In which case you would change it to 9600, add the node, then change it back to the operating speed.

In this way, no transmissions will be lost during the process.

The best solution is really to make the default rate your decided rate to simplify the adding and removing, or even rebooting, of nodes. In this case the code relating to SYSCOM_BITRATE can be removed from both the **Master** and the **Slaves**.

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.