AxiCat Server

v1.3.1

User Manual

September 2016

Table of Contents

Revision History

Date	Authors	Description
2014-09-05	Peter S'heeren	Initial release.
2014-09-17	Peter S'heeren	Added 1-Wire commands.
2014-09-28	Peter S'heeren	Added 1-Wire enumeration.
2014-12-12	Peter S'heeren	Added 1-Wire probing.
2016-09-13	Peter S'heeren	Updated for revamped server program.

1 Program

Overview

AxiCat Server is an indispensable tool for working with the AxiCat. The program offers the full set of features of the AxiCat and runs as a network server. It accepts text commands over network ports and standard I/O, and as such multiple users and programs can access the AxiCat locally and remotely.

The server offers an efficient Command Client Protocol that is both human-readable and easy to format and parse programmatically. Commands and responses are composed of ASCII characters. Empty lines and comments are permitted as well. These features come in handy when you create command files.

The server is based on the AxiCat Application Layer and offers the same high performance for all available interfaces. You can easily schedule multiple transfers for GPIO, I2C, SPI, 1-Wire and UARTs concurrently. The server will efficiently handle all transfers with the AxiCat and report appropriate responses when transfers have been completed.

AxiCat Server can tunnel data between a serial interface on the AxiCat and a network port. This feature enables you to easily set up a serial monitor.

Programming languages that support network sockets can easily connect to the server and work with the AxiCat.

The server works with one AxiCat. If you want to provide access to more than one AxiCat attached to the same computer, you can run multiple instances of the server program, one instance for each AxiCat.

Command Line

Parameter	Description
-h	Display help and exit.
-console	Open a console in Windows.
-v	Enable verbose output.
-axicat PATH	Select the AxiCat with the given serial path as the interface. Example PATH in Linux: /dev/ttyUSB0 Example PATH in Windows: \\.\COM4
-p n	Enable network command clients. The value specifies the port number the server must listen to. Value n=1..65535 decimal.
-pmax n	Specify the maximum of connections with network command clients at any given time. Value n=0 for unlimited, n=1.. sets limit. The default is unlimited number of connections.
-up u n	Enable server port for serial I/O. Value u specifies the serial interface. u=0..1. Value n specifies the port number the server must listen to. n=1..65535 decimal.
-crlf	Conclude responses with CR→LF instead of LF. The parameter only applies to network command clients.
-stdio	Enable standard I/O command client.
-i FILE	Specify a file with initialization commands.

If no parameters are provided, the program displays help and exits.

The -**crlf** parameter alters the end-of-line character sequence of responses for network command clients. By default, the server concludes each response with a LF character. This renders undesirable output in terminal programs that distinguish between LF and CR. PuTTY, for example, doesn't move the cursor to the beginning of the line when it receives LF. The **-crlf** parameter solves this particular problem.

Client Types

The server can interact with various types of clients:

- Command Client
  - Initialization File
  - Network Command Client
  - Standard I/O
- Serial Client

Initialization File

If parameter **-i** is specified, the server will process the given file after it's connected to the AxiCat. Using this parameter, you can have the server send various commands to your AxiCat. Doing so enables you to bring the AxiCat to a well-defined state before any client connection can start communicating with the AxiCat.

The initialization file is processed as a separate client that issues commands from the file. All responses are discarded.

Network Command Client

If argument **-p** is specified, the server creates the specified port when it has successfully connected to the AxiCat.

The server accepts multiple incoming connections on the command port. As such multiple network command clients can concurrently work with the AxiCat.

If the AxiCat is disconnected from the system, the port will be removed along with the client connection if present. Nonetheless the server will remain active and keep trying to reconnect to the AxiCat. This behavior corresponds very well with the plug-and-play nature of the USB-based AxiCat.

This also means that the AxiCat is initialized only once, during connecting, independently from connectivity on the server port. As such clients can connect with and disconnect from the server multiple times without having to worry about the state of the AxiCat; the state will not change in-between socket connections, unless the AxiCat went through a USB replugging phase. This is especially interesting if you want to communicate with the AxiCat from a website where each page has to reconnect with the server program.

Standard I/O

Standard I/O is enabled if argument **-stdio** is specified. Standard I/O allows another running process to directly write commands to the server and read responses back on the same system. That process is called the standard I/O command client.

Typical use-cases include manually typing in and sending commands from the console, piping files with commands to the server, and controlling the AxiCat from a scripting language.

Note that, when **-stdio** is specified, the server will exits when the AxiCat is not attached to or gets disconnected from the system.

Network Serial Client

AxiCat Server can tunnel data between a serial interface on the AxiCat and a network port. If argument **-up** is specified, the server creates the specified port for the specified serial interface when it has successfully connected to the AxiCat.

Note that you still have to send commands to configure the serial interface. A convenient way to configure the serial interface(s) is using an initialization file.

Windows Console

The Windows version of the server is built as a Win32 application, as opposed to a console application. Nonetheless, the server is capable of opening a dedicated console for displaying information. Parameters **-console** and **-h** will produce the console.

It's not advisable to combine parameter **-console** with I/O redirecting. For example:

```
> axicatserver.exe -axicat \\.\COM3 -console -stdio < commands.txt >
responses.txt
```

The console will take over standard input and output and the result won't be what you expect. Instead, run the command as follows:

```
> axicatserver.exe -axicat \\.\COM3 -stdio < commands.txt > responses.txt
```

No console will appear and the server will perform the job as expected, taking in the commands from the input file and emitting the responses to the output file.

2 Client Command Protocol

The server implements the AxiCat Application Layer module in its back-end and exposes a server socket (parameter **-p**) and standard I/O (parameter **-stdio**) in its front-end.

Clients are the processes that communicate with the server's front-end. Clients can be any kind of process, like a terminal program, a command line interpreter, a Python program.

You can find the specification of the Client Command Protocol in the documentation directory of the AxiCat source code.

3 Usage Examples

Network Command Port and Terminal Programs

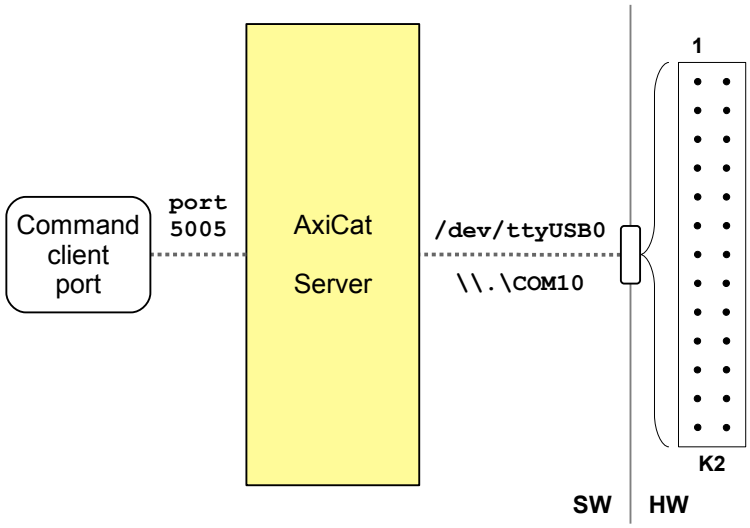
Suppose you want the server to enable access to the AxiCat via network port 5005. The following command will do the job in Linux:

```
$ ./axicatserver -axicat /dev/ttyUSB0 -p 5005 &
```

This will run the server silently in the background.

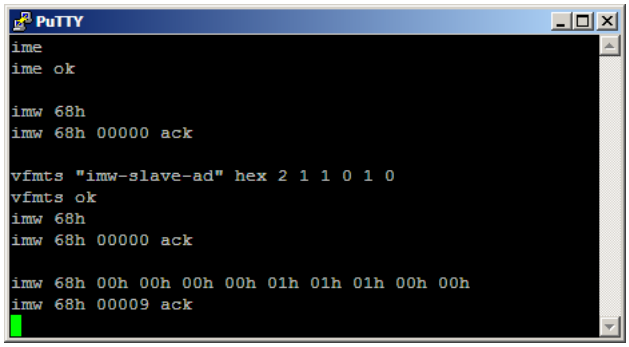
The equivalent command in Windows is:

```
> axicatserver.exe -axicat \\.\COM10 -p 5005
```



The server will start listening on port 5005 as soon as it's connected to the AxiCat. Any computer on your network that has access to the server is now able to communicate with the AxiCat.

You can manually send commands using a terminal program, like PuTTY:



In the example, the server is connected to from a remote computer. All it takes is the IP address and port number of the server. Next, a number of commands are typed in to which the server responds.

Another convenient way of connecting to the server is using **netcat** in linux:

```
$ nc 192.168.1.110 5005
ime
ime ok
imw 68h
imw 68h 00000 ack
```

Network Command Port and Programming Languages

The network command port is great for controlling the AxiCat from a programming language. Every popular language has built-in support for programming with network sockets. The format of commands and responses allows for very simple processing in your language of choice.

Run the server with network command port enabled. This command will do the job in Linux:

```
$ ./axicatserver -axicat /dev/ttyUSB0 -p 5005 &
```

The equivalent command in Windows is:

```
> axicatserver.exe -axicat \\.\COM10 -p 5005
```

The following example shows how to use the AxiCat from a Python program. Enter the following Python code in an editor:

```
import socket

# Change host if needed. For example: '192.168.1.110'
host = 'localhost'

# Change port number if already used.
port = 5005

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((host,port))

s.send('ior 5\n')

data = s.recv(1024)
print 'Received:'
print data

s.close()
```

Don't forget to change host if needed. Save this file as **client.py** for example.

You can run the program in Linux as follows:

```
$ python client.py
Received:
ior 05 1 0 out

$
```

The output shows that the Python program successfully connected to the server, sent the GPIO read command, and received the response containing the state of GPIO pin 5.

Standard I/O and Console

A typical use for standard I/O is manually typing in commands in the console. Example in Linux:

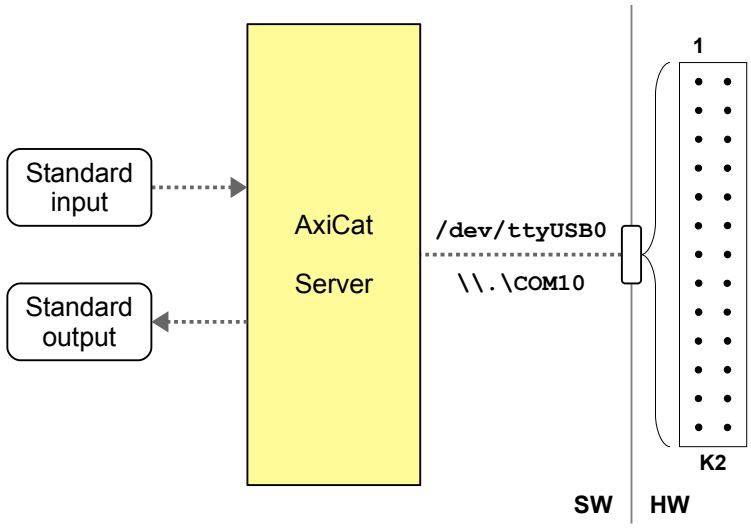
```
$ ./axicatserver -axicat /dev/ttyUSB0 -stdio
ior 0
ior 00 1 1 in
```

To accomplish the same in Windows, a console must be opened:

```
> axicatserver -axicat \\.\COM10 -stdio -console
```

An empty console appears where you can type in commands. A great feature of Windows console is line editing and history; you can use the arrow keys to navigate in all four directions.

Note: if you forget the **-console** parameter, the process will start up in the background and immediately terminate.



Standard I/O and Piping

Another application of standard I/O is piping. You can pipe an input file with commands to the server and pipe the responses to another file, like this:

```
$ ./axicatserver -axicat /dev/ttyUSB0 -stdio < commands.txt >
responses.txt
```

The equivalent command in Windows is:

```
> axicatserver -axicat \\.\COM10 -stdio < commands.txt > responses.txt
```

As mentioned in section Windows Console, don't specify the **-console** parameter in this case.

Initialization File

Let's create an initialization file that sets up $\overline{SS2}$ and enables the SPI master. Create the initialization file with an editor:

```
# Initialization file

# Set up SS2/GPIO5 as output, logic one
iod 5 out
iow 5 1

# Enable the SPI master
sme
```

Save the file as **spi-init.txt** for example.

Let's start the server with network command port enabled. This command will do the job in Linux:

```
$ ./axicatserver -axicat /dev/ttyUSB0 -p 5005 -i spi-init.txt &
```

The equivalent command in Windows is:

```
> axicatserver -axicat \\.\COM10 -p 5005 -i spi-init.txt
```

With the server up-and-running, you can now start working with the SPI master without having to worry about setting up the AxiCat.

Serial Client

The following invocation of AxiCat Server configures UART #0 to 115200 baud 8-N-1 and binds the serial interface to port 5006. In Linux:

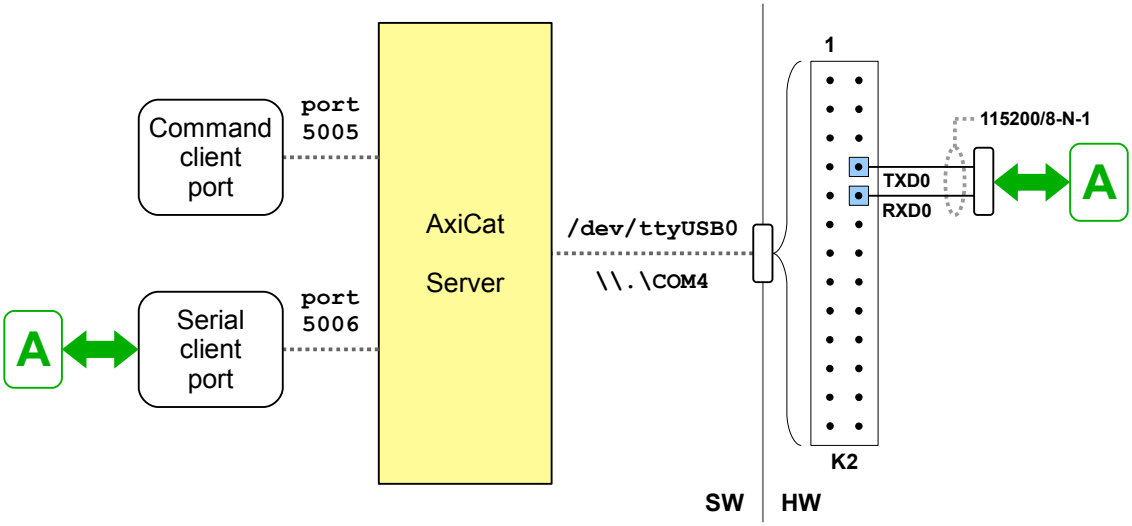
```
$ axicatserver -axicat /dev/ttyUSB0 -p 5005 -up 0 5006 -i inicmds.txt
```

In Windows:

```
> axicatserver.exe -axicat \\.\COM4 -p 5005 -up 0 5006 -i inicmds.txt
```

Initialization commands file **inicmds.txt**:

```
u0sb 115200    # Set baudrate
u0sd 8         # Set word size
u0ss 1         # Set stop bits
u0e            # Enable UART #0
```



The server tunnels the serial data between the two points marked A in the picture.

Use a terminal program to connect to the serial client port. For example:

```
$ nc 192.168.1.125 5006
```

Similarly, you can create a serial client port for UART #1 or both UARTs.

All Client Types

Enable all types of clients. In Linux:

```
$ axicatserver -axicat /dev/ttyUSB0 -p 5005 -up 0 5006 -up 1 5007 -stdio
-i inicmds.txt
```

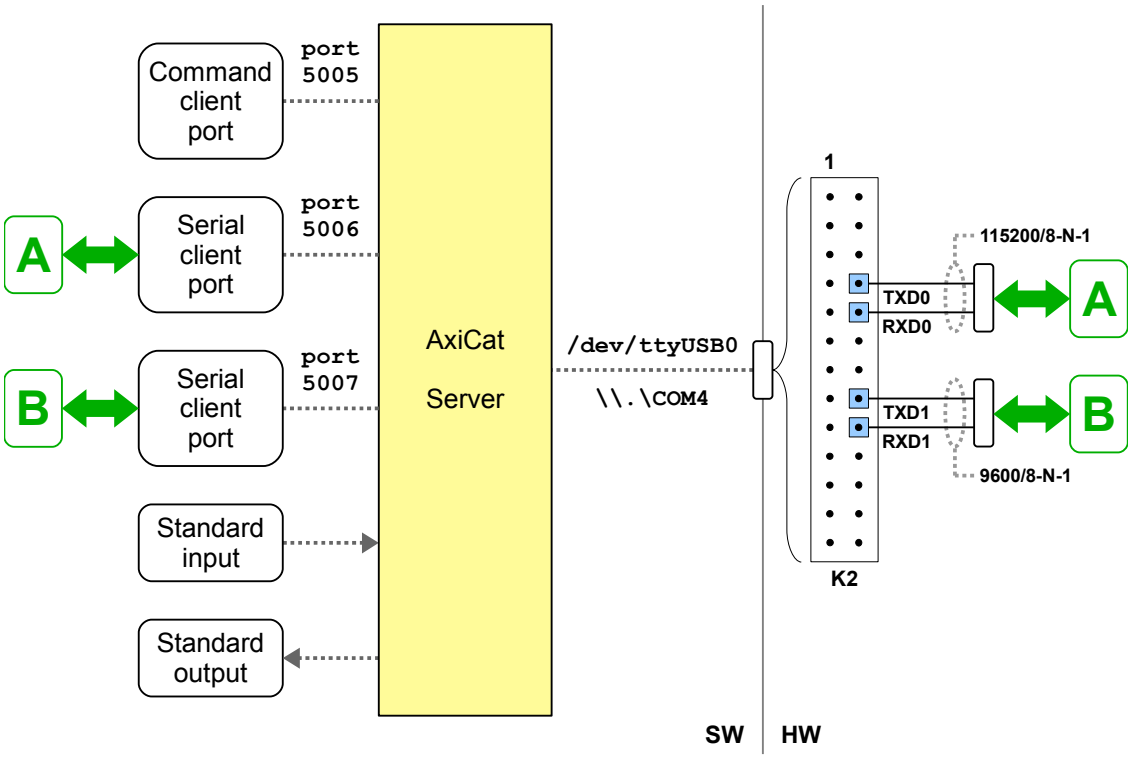
In Windows:

```
> axicatserver.exe -axicat \\.\COM4 -p 5005 -up 0 5006 -up 1 5007 -stdio
-i inicmds.txt
```

Initialization commands file **inicmds.txt**:

```
u0sb 115200    # Set baudrate
u0sd 8         # Set word size
u0ss 1         # Set stop bits
u0e            # Enable UART #0

u1sb 9600      # Set baudrate
u1sd 8         # Set word size
u1ss 1         # Set stop bits
u1e            # Enable UART #1
```



Shutting Down

When you run the program as a server process, it silently runs in the background. In order to shut down the server, you can:

- Send the **quit** command.
- Kill the process.

The first method only works when the server is connected to the AxiCat.

Killing the process always works. The procedure depends on the operating system.

Linux

Look up the process identifier of the server and send the kill signal. You must have root permissions to do that. For example:

```
# ps -A | grep axicatserver
 2111 pts/0    00:00:01 axicatserver
# kill -9 2111
```

You can use command **killall**:

```
# killall axicatserver
```

Note that **killall** will terminate all processes with the specified name.

Windows

Press CTRL-ALT-DEL to open the Task Manager. Look for a process with image name **axicatserver.exe**. Right-click and choose End Process. The process will be terminated.

4 Serial Paths

You need to specify a serial path in order to communicate with a serial port. The next sections explain in more detail how you specify serial paths in each supported operating system.

Linux

Serial ports are accessible in the device directory structure. A serial path starts with **/dev**. A serial path is case-sensitive.

The following table summarizes serial paths that are commonly found on Linux systems:

Serial Path	Serial Port
/dev/ttyS0	The computer's 1st on-board serial port
/dev/ttyS1	The computer's 2nd on-board serial port
/dev/ttyS2	The computer's 3rd on-board serial port
/dev/ttyS3	The computer's 4th on-board serial port
/dev/ttyUSB0	1st USB serial adapter
/dev/ttyUSB1	2nd USB serial adapter
/dev/serial/by-id/	This directory contains symbolic links to serial devices. Each symbolic link name identifies a specific device.
/dev/serial/by-path/	This directory contains symbolic links to serial devices. Each symbolic link name represents a hardware path to the device, like a specific USB port on your computer.

Here are some useful commands you can run to get information about present serial devices and their corresponding serial path. The following commands were run on a Linux system with one on-board UART and one connected USB serial adapter.

Filter information from the kernel message buffer:

```
$ dmesg | grep 'tty'
[    0.000000] console [tty0] enabled
[    0.516785] serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
[    0.517463] 00:0b: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
[    0.559097] tty tty55: hash matches
[   66.076326] usb 2-1: FTDI USB Serial Device converter now attached to ttyUSB0
```

Use the **setserial** command to produce a list of serial paths:

```
$ setserial -g /dev/ttyS* /dev/ttyUSB*
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
/dev/ttyS1, UART: unknown, Port: 0x02f8, IRQ: 3
/dev/ttyS2, UART: unknown, Port: 0x03e8, IRQ: 4
/dev/ttyS3, UART: unknown, Port: 0x02e8, IRQ: 3
/dev/ttyUSB0, UART: unknown, Port: 0x0000, IRQ: 0, Flags: low_latency
```

List serial devices in the device directory:

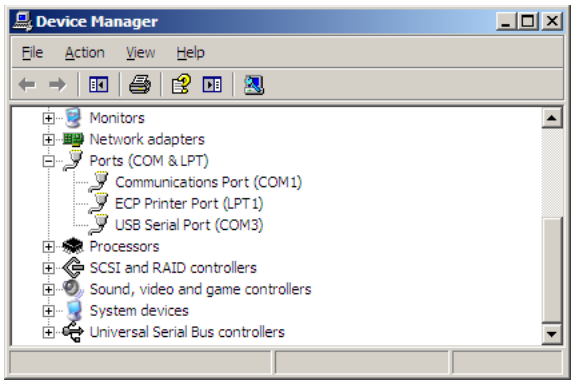
```
$ ls -l /dev/ttyS* /dev/ttyUSB*
crw-rw---- 1 root dialout   4, 64 2012-02-22 14:19 /dev/ttyS0
crw-rw---- 1 root dialout   4, 65 2012-02-22 14:19 /dev/ttyS1
crw-rw---- 1 root dialout   4, 66 2012-02-22 14:19 /dev/ttyS2
crw-rw---- 1 root dialout   4, 67 2012-02-22 14:19 /dev/ttyS3
crw-rw---- 1 root dialout 188,  0 2012-02-22 14:20 /dev/ttyUSB0
```

Windows

Serial ports are accessible through the Win32 device namespace, which is part of the NT namespace. As such a serial path starts with **\\.** followed by the device name of the serial port. A serial path is case-insensitive.

A serial port is typically named **COM<x>** where **<x>** is a number between 1 and 256. Other naming schemes may apply and aliases may exist, depending on the serial driver that controls the serial port.

You can obtain a list of available serial ports in the device manager. Open the device manager and view devices by type. The section named *Ports (COM & LPT)* contains all available serial and parallel ports.



The device name of a serial port is shown between parentheses. In the picture to the right, you can see two serial ports. COM1 is the PC's on-board serial port, COM3 represents a USB serial adapter.

The serial paths to the serial ports in the picture are:

Serial Path	Serial Port
\\.\COM1	*Communications Port (COM1)*
\\.\COM3	*USB Serial Port (COM3)*

Serial paths like **COM1** (that's without the **\\.** prefix) will work because COM1 to COM9 are reserved names in the NT namespace. COM10 to COM256 aren't reserved names and you'll have to specify the **\\.** prefix with these device names. By comparison, serial path **\\.\COM100** will work but serial path **COM100** won't work.

5 Software Revision History

Version	Description
1.0.0	▪ Initial release of AxiCat Server. ▪ Based on AxiCat AL v1.0.0.
1.1.0	▪ Added 1-Wire commands. ▪ Based on AxiCat AL v1.1.0.
1.2.0	▪ Added 1-Wire enumeration command. ▪ Based on AxiCat AL v1.2.0.
1.3.0	▪ Added 1-Wire probing command. ▪ Based on AxiCat AL v1.3.0.

6 Software License

The license is stated in the source code.

7 Legal Information

Disclaimer

Axiris products are not designed, authorized or warranted to be suitable for use in space, nautical, space, military, medical, life-critical or safety-critical devices or equipment.

Axiris products are not designed, authorized or warranted to be suitable for use in applications where failure or malfunction of an Axiris product can result in personal injury, death, property damage or environmental damage.

Axiris accepts no liability for inclusion or use of Axiris products in such applications and such inclusion or use is at the customer's own risk. Should the customer use Axiris products for such application, the customer shall indemnify and hold Axiris harmless against all claims and damages.

Trademarks

All product names, brands, and trademarks mentioned in this document are the property of their respective owners.

8 Contact Information

Official website: http://www.axiris.eu/