# UART Thou Mad?

*An Introduction to the UART Hardware Interface*

Mickey Shkatov

Toby Kohlenberg

# Table of Contents



## Abstract

Although hardware hacking has gained visibility in the last couple years there are still interfaces and capabilities that are relatively unknown. In this paper we attempt to provide an introduction to the UART interface; what it is, why it is interesting, how to interact with it and some of the things a researcher could do with it. Our goal is to increase awareness and understanding of the impact that a UART interface has on system security.

## Introduction to UART

UART stands for Universal Asynchronous Receive Transmit. It is a generic term used to describe interfaces that enable translation of data from a serial interface to a parallel interface. UARTs have been around for a very long time – Gordon Bell is referenced as implementing the UART interfaces for PDP series computers.

In modern systems UART interfaces are frequently used to provide an interconnect between components in a device. Because of the simplicity of the interface it is tremendously easy to intercept and/or inject traffic into a UART connection. It has also gained popularity as an alternative debug/recovery interface for embedded devices where there is not likely to be a full keyboard/video/mouse attached to it but problems may still need to be debugged or undergo repairs in the field.

The UART interface provides a useful and different function from the more well-known JTAG interface. Where a JTAG interface allows among other things direct memory access and CPU register access, the UART interface is much less complex and is best thought of as an alternate serial interface for the system which may be connected to the OS console terminal.

On systems that have UART it is generally fairly easy to identify the interface. The most common configuration you are likely to see is something like this:

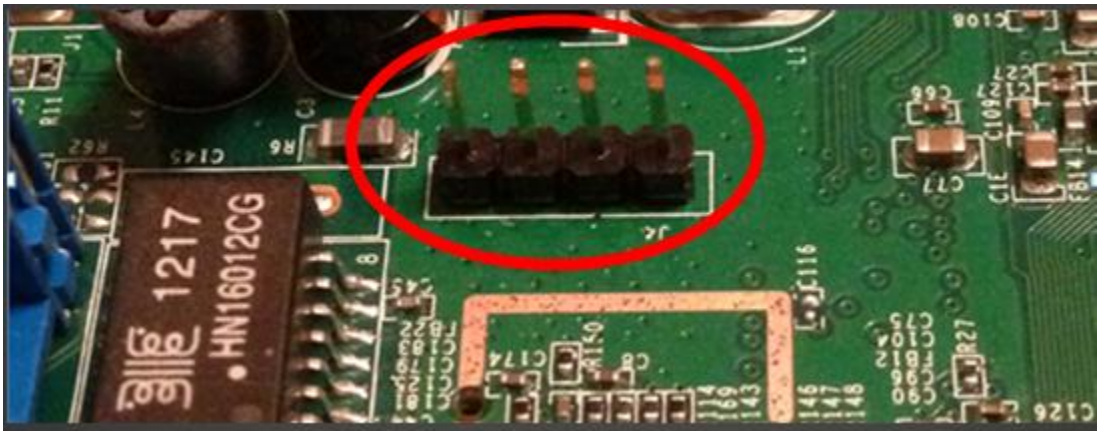


Figure 1 A common pin configuration for a UART interface

Even in cases where pins are not present it is frequently possible to identify the UART contact points by looking for descriptions on the PCB silkscreen as seen here:

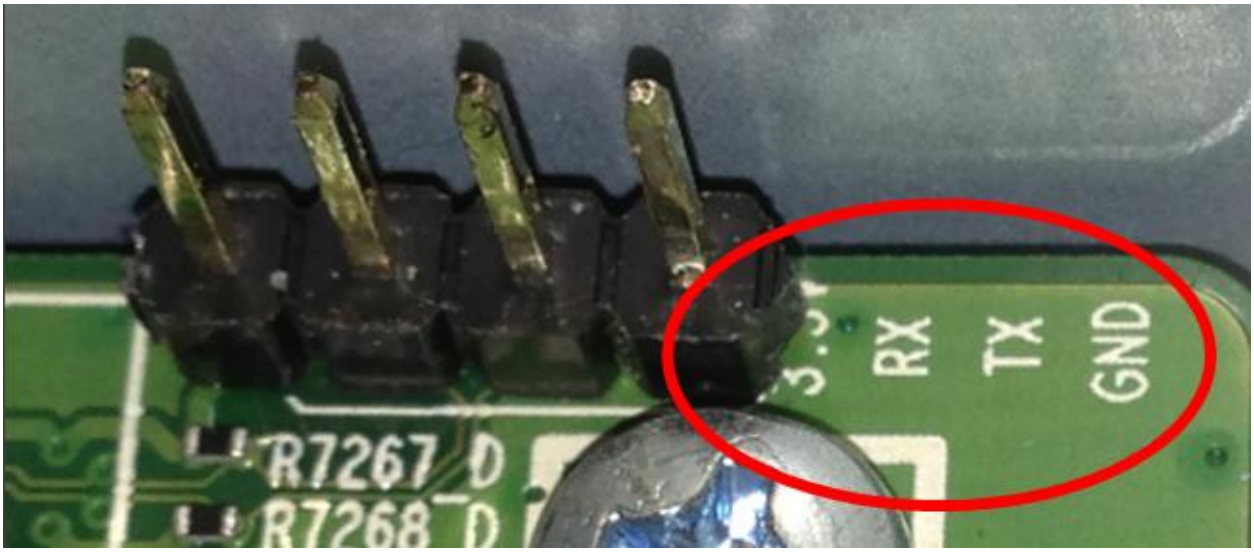


Figure 2 the four descriptions seen here are characteristic of UART

The descriptions mentioned above can also hint at another method of finding a UART interface. By leveraging a multimeter and testing sets of four pins or pads to first find Ground, you can then connect the other pins to your tool of choice (frequently a BusPirate) and boot the device. If anything is displayed via the BusPirate that isn't random garbage you can have a high confidence that you may be looking at a UART interface.

## Who uses UART

As we mentioned previously the most common use for UART is as an alternate means of console access for devices that are unlikely to have a keyboard and video display available to them. The most frequent examples of this are embedded devices leveraging System-on-Chip platforms. While embedded devices used to be relatively rare they are now a fact of daily life. This is a result of the increasing availability (due to reduced cost and increased performance) and the increasing demand for processing capacity in commonly used devices.

Most people when given the description of an embedded system will immediately think of their personal wireless network router or possibly their phone. These are both good and obvious however there are many more that are worth exploring. Some interesting examples include:

- Smart home power controllers
- HD TV streamers
- Web cams
- Smart TVs
- Smart appliances
- Blu-ray players
- Home wireless phones
- Network-Attached Storage platforms

## Essential Tools

In order to explore and use UART interfaces there are some essential tools which must be purchased and understood. Most are not expensive and all are widely available. The tools are as follows:

### The FCC ID database

The FCC ID database is an online searchable database containing descriptions and images of all FCC-approved devices. This can be tremendously useful in identifying devices that have UART interfaces and might be worth further exploration. It has other valuable uses for hardware hackers and as such is worth getting to know well.

### Multimeter

A multimeter is essentially a device that combines the ability to test or measure multiple electrical functions into a single unit. Which measurements are combined depend on the device you purchase but the key ones we are interested in are the ability to measure resistance and voltage.

### BusPirate

The BusPirate has been described as “the Swiss army knife of hardware hacking” simply because it gives the researcher access to a wide range of capabilities and makes it easy to add more features in a custom fashion. An extensive list of the features can be found on the Wikipedia page (http://en.wikipedia.org/wiki/Bus_Pirate) or by reviewing documentation pages provided by vendors selling them.

### USB-UART cable

This is a very simple cable that offers a UART interface on one end and a USB interface on the other. In some cases this is all that is necessary to connect your research system to your research target.

### Soldering Iron, Magnifying Glass, Bright Light

It is not uncommon for the UART pads to have no pins attached and these basic tools are essential for all hardware hacking or exploration.

## UART and Security

What makes UART interfaces so interesting from a security perspective is what they are used for within systems today. The main use that we have found is to act as a reliable method of connecting to a terminal or console session for the embedded operating system. This doesn't necessarily have to have a negative impact for security unless it is done wrong. Unfortunately we have found multiple cases of it being done wrong.

The most critical mistake that we've seen is providing a console that is running as root without requiring any additional authentication. The consequences of doing something like that should be self-evident to everyone. Rather than discuss why doing this is a bad thing we will enumerate some of the possibilities that are more unique to embedded systems.

Having root shell access enabled us to:

- Modify the running system
- Explore the file system and find services or binaries (including debug-enabled binaries that have no business being on production platforms) that might be good targets for fuzzing. This may enable non-physical-access compromise of the embedded systems.
- Flash the firmware on the platform to change the behavior in any way we liked.
- Explore the file system and find interesting data that the vendor left behind. In one case although the firmware was encrypted and signed, the encryption and signing keys were both left in the device's file system[1].

One significant advantage of modifying the running system is that on embedded systems the running image may be completely in memory. This means that when the device is power cycled there will be no forensic traces left behind to indicate what was done to the device.

The obvious and appropriate control to put in place here would be to require authentication when accessing the embedded system's operating system however that would still leave the next vulnerability we are going to discuss exposed.

Even when an unauthenticated console session is not available having the system's console output to the UART results in all debug and system log information being made available to a researcher. This is

[1] http://www.laplinker.com/2013/05/cve-2013-3518-belkin-wemo-information.html

tremendously useful for reverse-engineering and learning about the behavior of a system. For instance, when a USB device is plugged into a Linux system, the path being searched by an update tool may be displayed on the console. This makes it trivial to create said path on a USB drive and provide any kind of update files you choose to the device.

The correct choice here is to simply not use the UART interface as the target for system console. A second-best option may be to disable logging to the console[2].

## Conclusion

As we stated at the beginning, the UART interface is not well-known but has significant power and potential value to a security researcher or attacker. This power makes it more interesting as a research target as does the low amount of knowledge and low cost of tools that are required to explore it. There are simple and fundamental things that anyone developing a platform that includes UART should do to protect the interface and to manage the attack surface that is presented by having a UART interface.

[2] http://www.tldp.org/HOWTO/Remote-Serial-Console-HOWTO/upload-logging.html