# BusRaider 1.7 Construction, Testing and Initial Setup and Use

Rob Dobson 2019

# Prerequisites

In order for the BusRaider to operate a few things need to be ensured:

* The BusRaider is in a PRO backplane (such as Spencer’s <https://rc2014.co.uk/modules/backplane-pro/> or Steve’s <https://www.tindie.com/products/tindiescx/modular-backplane-boards-for-rc2014/>) as bus additional bus lines BUSRQ, BUSAK, WAIT, NMI, RST2(Page) are used
* USB keyboard and adapter to allow it to work with the micro USB connector on a Raspberry Pi Zero

# BusRaider Functionality

The BusRaider is designed to plug into the bus of the RC2014 Retro-Computer (<https://rc2014.co.uk/>). It performs two main functions:

* Emulation of the memory-mapped video and keyboard functionality of some 1970’s/80’s era computers such as the Sinclair ZX Spectrum and Tandy TRS-80
* Single-step debugging of Z80 programs

To perform these functions the BusRaider has two main tricks up its sleeve:

* The ability to request access to the target microprocessor’s bus (the target microprocessor is the Z80 of the RC2014) using the BUSRQ line. During this period the target microprocessor “let’s go” of all the bus lines and allows the BusRaider to read/write memory and IO without interference.
* The means to detect when the target microprocessor is performing a memory/IO operation or responding to an interrupt and forcing the microprocessor to wait (under the control of the BusRaider) in the middle of this operation so that the BusRaider can (a) respond to the memory/IO operation itself or (b) wait until the user single-steps through the program or (c) detect break conditions (such as breakpoints set by a debugger) and pause execution at that point

There are two processors on the BusRaider:

Raspberry Pi Zero (or Zero W) referred to elsewhere as Pi

ESP32

Each has a specific role. In the case of the Pi this is

# Constructing the BusRaider

# Inspection and Testing

# Initial Setup

# Usage

# Appendix 1: Updating the ESP32 Program

The ESP32 firmware is contained in the folder BusRaiderESP32 in the GitHub repository. To build and run this code you will need the following:

\*

Visual Studio Code (https://code.visualstudio.com/) - VSCode

\*

PlatformIO - this can be installed from VSCode by clicking the Extensions icon in the left-hand column (it looks like a square within a square) and searching for PlatformIO - more details at https://platformio.org/

\*

Espressif 32 Platform - in the Home Page of PlatformIO (click the house icon in the bottom-left toolbar to get the PlatformIO Home Page) select Platforms and search for Espressif 32 then install it

\*

USB to Serial adapter such as 3V3 FTDI Cable https://www.amazon.co.uk/dp/B071WPW292 or equivalent "Prolific DebugCable" https://www.amazon.co.uk/gp/product/B01N4X3BJB,

Building the ESP32 Firmware

At this point you should be able to open the folder for the BusRaiderESP32 sub-project (it won't work if you open the parent folder for the whole of BusRaider) in VCCode and then click the tick icon in the bottom-left toolbar. This should build the BusRaiderESP32 project.

Programming the ESP32

First connect the serial cable (FTDI or similar) to the BusRaider PCB. This must be connected to the serial port at the top-right of the BusRaider PCB. Check the connections carefully and, if you are in any doubt about whether you really have a 3V3 cable, make sure the protection diode D1 is in place and connected the correct way around.

The connections for a genuine FTDI cable (black USB connector body) are:

\*

Black or Blue = GND (J1 Pin 1)

\*

Orange = FTDI TXD (J1 Pin 4)

\*

Yellow = FTDI RXD (J1 Pin 5)

And for the "Prolific Debug Cable" referenced above (blue USB connector body):

\*

Black = GND (J1 Pin 1)

\*

Green = FTDI TXD (J1 Pin 4)

\*

White = FTDI RXD (J1 Pin 5)

In addition two jumpers need to be in place to connect the FTDI Rx/Tx signals to the ESP32. These are on J5 (mid-way down the right side of the BusRaider PCB). You should put jumpers in place vertically to connect between Pi and ERSP1 for both the RX and TX columns.

If this is the first time you have connected the serial port it would be advisable to use a terminal emulator such as TeraTerm (described above) or the Serial Monitor built into the PlatformIO extension. The serial port settings are 115200, 8, N, 1 (but the baud rate 115200 is the only thing that generally needs setting). With the board powered up you should see a regular (once every 10 seconds) message from the ESP32 on the serial port indicating its status including WiFi connection, etc. If you don't see this (or other similar activity) then check the jumpers and the FTDI cable connections.

To get the firmware onto the ESP32 on the BusRaider you will need to make sure the settings in the platformio.ini file are correct. This file is in the root of the BusRaiderESP32 folder. The most important setting is the upload\_port. If on Windows this should be the COM port (e.g. COM3) that the FTDI serial port is using (you can find this out with a terminal emulator program like TeraTerm (https://ttssh2.osdn.jp/index.html.en) or through the device manager in windows. On Linux it will probably be /dev/ttyUSB0.

Next you need to put the ESP32 into download mode. To do this hold down the button marked DEMO on the BusRaider while pressing and releasing the ESP RESET button (the one closest to the ESP32 at the top of the PCB).

Finally press the right-arrow button on the bottom-right toolbar in PlatformIO to start programming the ESP32. If all goes well you should see the stages of the programming process and a success message.

The ESP RESET button has to be pressed again when programming is finished to start the new ESP32 firmware.