FTDI LibMPSSE SPI

Asked 1 year, 1 month ago Active 9 months ago Viewed 1k times

1

I recently purchased the FTDI C232HM-DDHSL-0 USB cable for use with SPI and I2C devices; my goal is to read the memory from an SPI memory chip that I removed from a router. However, I am having issues getting the libMPSSE library & 2xx drivers to send signals to a connected device.**I hooked the leads of the C232 up to a Saleae logic analyzer and saw that no signals were being output!**

Useful links for this question:

- FTDI C232HM-DDHSL-0 (purchase): https://www.ftdichip.com/Products/Cables/USBMPSSE.htm
- C232HM-DDHSL-0 data sheet: https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_C232HM_MPSSE_CABLE.PDF
- Memory Chip data sheet: https://www.macronix.com/Lists/Datasheet/Attachments/7370/MX25L6406E,%203V,%2064Mb,%20v1.9.pdf
- 2xx drivers: https://www.ftdichip.com/Drivers/D2XX.htm
- MPSSE SPI library: https://www.ftdichip.com/Support/SoftwareExamples/MPSSE/LibMPSSE-SPI.htm

I am using a **Debian x86_64** machine. The MPSSE SPI download only had the i386 version of the library, so I downloaded the MPSSE SPI source (https://www.ftdichip.com/Support/SoftwareExamples/MPSSE/LibMPSSE-SPI.htm) and built the x86_64 bit version. There were no build errors or warnings.

I copied libftd2xx.so and libMPSSE.so to /usr/local/lib.

I plugged the cable in, then removed the following kernel modules:

- ftdi_sio <-- Readme says to take take out
- usbserial <-- Readme says to take out
- usb_wann <-- Needed to remove to take out usbserial
- qcserial <-- Needed to remove to take out usbserial

At this point I was hoping that my set up was done correctly.

I connected 6x cables to the 8 pin memory chip:

- VCC <--> red lead
- Ground <--> black lead
- Chip select <--> brown lead

You could try to verify if the device is functional: Disconnect the slave device (mem). If you have a scope check if you see a clock signal during transmission (be aware that the MPSSE has some built in sleep(2) in its code base - this means you need proper trigger setting to see something - especially with a clock of 30MHz). Additionally you can check the read functionality by connecting the MISO to either GND or VDD and start a read. If the response is all zero for GND and VDD, the reading should be functional. – Christian B. Oct 23 '19 at 21:57

Thanks for the response Christian. I connected the cable's MISO to ground and saw 0's in my buffer! I am not sure what VDD is, but I connected to 3.5 volt VCC and saw all F's, which I expect. I also hooked the MISO to the MOSI and saw exactly what I intended to send. – donsiuch Oct 24 '19 at 15:32

I am optimistic that my cable is fine. Thank you for the debugging suggestion! Perhaps the issue may be a state that the memory chip is, or an incompatibility: The max clock that the chip operates at is 33MHz but I am using 30MHz (cable's max) for example. I don't have a scope but I do have a saleae that I will try again – donsiuch Oct 24 '19 at 15:51

Try a lower clock frequency. Check if you selected the right CS pin, phase and polarity. I did not read the mem datasheet in detail but I noticed that the interface is not really SPI compatible as the SI pin can act as input AND output to effectively double the transmission rate. So better check under which circumstances the interface could be SPI like. – Christian B. Oct 24 '19 at 16:02

I set clock to 50KHz and the wave form looked much better. Chip select, MOSI, and clock all looked as I hoped. However, at that clock rate, MISO stayed high. When I change the clock back to 30MHz I see some non high Bytes for the first few, then high after. Is it possible this thing requires me to operate at 33MHz for a read (my cable's maximum is 30MHz)? In the data sheet it says reads are a max of 33MHz... Also I started messing around with trying to change it's mode in the case it is in a weird state. No differences in MISO yet, but will keep putzing :) – donsiuch Oct 24 '19 at 23:57

1 Answer

Active Oldest Votes

1

Set device first to MPSSE mode, and it should work. See quite minimum (Python) example below based on FTD2XX library only (tested in Windows). LibMPSSE is not required to run simple SPI. Data can be read by `s = dev.read(nbytes)` , where nbytes is byte count. Find more information in nice tutorial: Driving an SPI device using MPSSE

```
import ftd2xx
OPS = 0x03                                  # Bit mask for SPI clock and data out
OE = 0x08                                   # CS

dev = ftd2xx.open(0)                        # open first available device
if dev:
    dev.setTimeouts(1000, 1000)
    dev.setBitMode(OPS, 2)                  # bit mask, MPSSE mode
    dev.write(bytes((0x86, 59, 0)))         # set SCK to 100kHz
    dev.write(bytes((0x80, 0, OE + OPS)))   # CS low
    data = 0x55,
    n = len(data) - 1
    dev.write(bytes(((0x11, n % 256, n // 256) + data))) # write SPI data
    dev.write(bytes((0x80, OE, OE + OPS)))  # CS high
    dev.close()
```

edited Jan 29 at 5:34

answered Jan 23 at 7:17

juha
76 6