October 5, 2018

v1.0

Using a CP2102 with BSL-Scripter

In another Github repository, I provided information on using inexpensive USB-to-UART adapters such as Silabs' CP2102 with older Texas Instruments MSP430 processors that use BSLDEMO for UART BSL flashing. This repo addresses using the same adapters when flashing the newer MPS430 parts that require BSL-Scripter.exe, including the F5xx, F6xx and FRxx parts.

All relevant MPS430 parts use a special pattern transmitted to their /Reset and Test pins to invoke BSL via hardware. BSLDEMO generates that pattern, but BSL-Scripter does not. Instead, it depends on the hardware programming interface to do that. Supported interfaces include the Rocket and the MSP-FET. However, those interfaces are large and expensive, whereas something like a CP2102 is small and cheap, and could even be embedded on a project PC board so the only hardware a user would need to update firmware would be a USB cable. But of course the CP2102 cannot generate the invoke pattern on its own.

Included here is INVOKE.exe, a C program for the Windows console, which generates the invocation pattern via the DTR (to /Reset) and RTS (to Test) outputs of a CP2102N, the Silabs VCP driver for which must also be installed. It exits leaving DTR high and RTS low, as required for BSL flashing. After running INVOKE to initiate BSL in the target device, it should then be possible to run BSL-Scripter to do the actual flashing. Unfortunately, the first thing Scripter does is bring DTR (and /Reset) low, which terminates the BSL session and puts the target MSP430 device into reset.

The source code for Scripter is available from TI, but it is complicated and depends on third party code sources. But in theory it should be possible to modify Scripter so it leaves DTR alone, or, ideally, adds the option to generate the invoke pattern itself. Perhaps TI will eventually do that modification. In the meantime, what's presented here are hardware solutions which disconnect the DTR output from /Reset after INVOKE generates the pattern, so /Reset stays high via its external 47K pullup resistor even when Scripter pulls DTR low. This allows flashing to proceed.

Solution #1 - Manual Disconnect

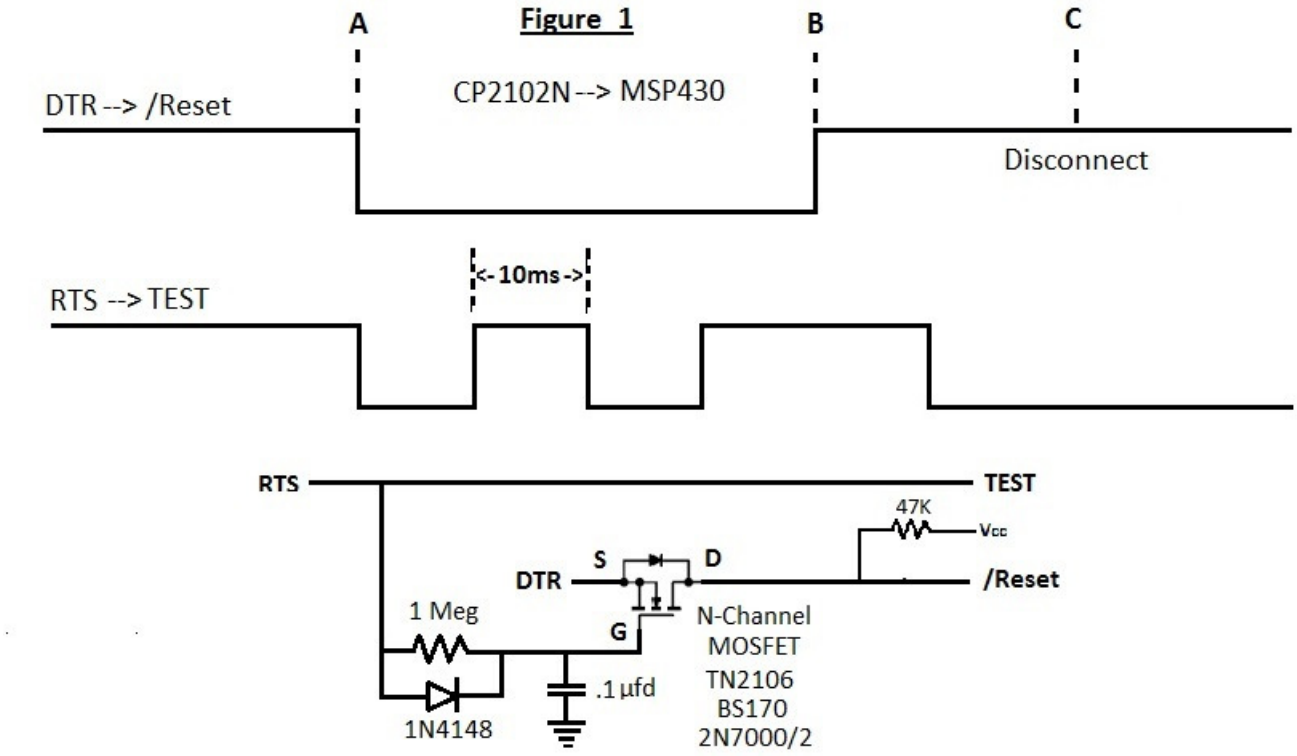
At the simplest level, you can just manually disconnect DTR from /Reset after running INVOKE. A batch file for this method would look something like this:

```
INVOKE.exe COM4
Pause
BSL-Scripter.exe script.txt
```

At Pause, you will be asked to press any key to continue, but before doing that, just manually disconnect DTR from /Reset. This has been successfully tested on an FR2311 under Windows 7/64.

Solution #2 - Electronic Disconnect

It's possible to use the same batch file, but without the Pause, by adding a small circuit consisting of four parts. The circuit will automatically disconnect DTR from /Reset after INVOKE has generated the pattern and invoked BSL. The circuit could be embedded in a project along with the CP2102, so the final user wouldn't have to deal with manual disconnect timing. This too has been successfully tested on an FR2311 under Windows 7/64. The invocation waveform and the schematic of the circuit are shown below in Figure 1.



The circuit connects DTR to /Reset through an N-channel MOSFET, the Gate of which is kept high for a time by the RTS line. As RTS goes high, it charges the Gate capacitor through the diode and resistor, which turns on the MOSFET and connects the two end points. When RTS goes low, the capacitor slowly discharges through the resistor, but the MOSFET stays on for a time. The capacitor is fully recharged when RTS goes high again, but eventually, when RTS goes low permanently, the Gate voltage drops low enough and the MOSFET turns off. Thereafter, changes on DTR have no effect on /Reset, which is kept high by its pullup resistor. Scripter takes DTR low, but it also keeps RTS low, so the MOSFET remains off while Scripter runs. INVOKE sleeps for 200 ms before terminating to make sure the capacitor is fully discharged when Scripter runs.

In the Pictures folder are scope captures. Pic 1 shows the overall invocation waveform on DTR and RTS transmitted by INVOKE. It provides a first-positive-pulse duration on RTS of 10 ms. Pic 2 shows the Gate voltage resulting from the pattern. Pics 3 and 4 show when the Gate voltage drops to the point that the MOSFET actually turns off (point C in Figure 1). The bottom trace show the voltage on the /Reset pin while DTR is tied low throughout. Eventually the MOSFET turns off and the pullup resistor takes over. Pic 3 shows the turnoff point for a BS170, and Pic 4 shows it for a TN2106, which has a lower Gate-Source threshold voltage and thus takes longer to turn off.

As a practical matter, it appears the BS170 or something in the 2N700x family should work fine. The MOSFET only needs to stay on while DTR is low. At all other times the MOSFET is off even if the Gate is still high. That's because when the Source (DTR) goes high, the GS differential is zero at most, and the MOSFET turns off. So as shown above in Figure 1, the MOSFET needs to stay on during the A-B period, but not thereafter. The longest period when the the Gate needs to stay above the threshold voltage while RTS goes low is 10 ms. Yet as shown in Pic 3, even the BS170 stays on for a full 40 ms after RTS goes low for good. So any MOSFET with a typical GS threshold voltage of 2.5V or lower would probably work.

Both the source code and the executable for INVOKE are provided here, as well as the batch, script, password and firmware files used for testing with the FR2311. COM4 is shown as the Silabs driver port, but that may be different on your system. Be sure to add the PARITY option on the MODE line in your script.

Testing was done with the CP2102, but this should also work with any USB-to-UART adapter such as the FT232 and CH340 parts, assuming their drivers are installed.

BSL-Scripter.exe should be downloaded from Texas Instruments:

http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSPBSL_Scripter/latest/index_FDS.html

This project follows an original idea of Aaditya Chaudhary to generate the invoke sequence under Linux:

https://github.com/saint-shark/TI-BSL-invoke-sequence-generator/blob/master/BSL_invoke.c