

Dragonhide

A line driver / protection circuit for ATAVRDRAGON

Kasper Pedersen <dragonhide@kasperkp.dk>

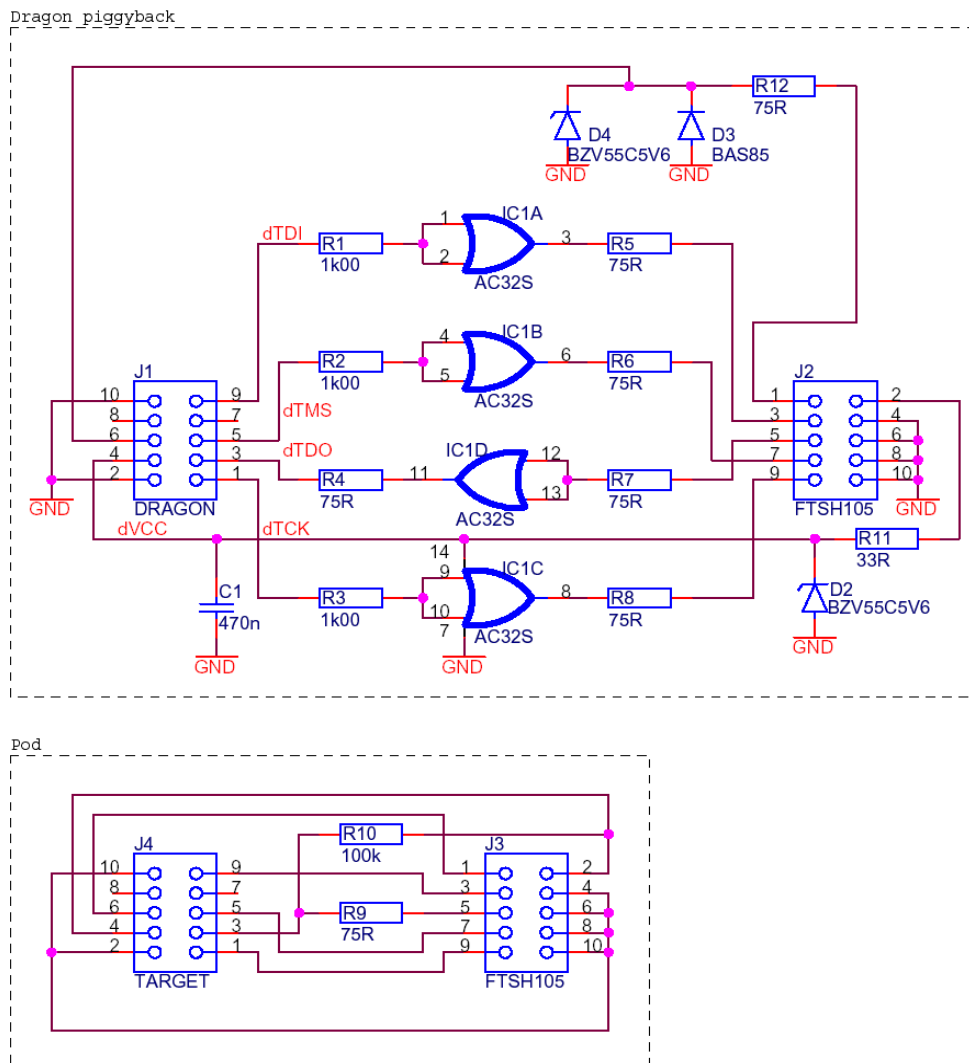
Why?

The Dragon is a *very* strongly cost optimized: In the original JTAGICE the signal path consists of a level shifter, a driver, and a neat transient protection array (from Protek) that will handle 40A pulses. The ICEmkII use MAX3392 series 15kV ESD protected level translators which, while not as much overkill, are still pretty robust.

The Dragon on the other hand *only* has transistor based level shifters, which means that not only can't it drive a cable load, it's also *_very_* sensitive to ESD or other transients.

This solution

If you don't have a cable between the Dragon and the target, protecting the Dragon with a handful of 5,6V zener diodes might be all you need. On my desk, on the other hand, i have multiple devices in the JTAG chain, it's a noisy environment, and I have a hint of signal-integrity-freak. I want it to work. Always. Even with the beefy motor attached to the setup.



There are two boards, similar to the configuration used by JTAGICE:

The first board contains a quad OR gate used as line driver and receiver, with impedance matching resistors, and a transient protection circuit for the DebugWire interface. The second board reorders the pins and provides impedance matching for the target TDO. The

two boards are connected with a ribbon cable (*this is important!*).

Line termination for those who haven't ever worked with transmission lines:

The source impedance matching on the first board works as follows: Let's assume that TCK (Dragon pin 10) is low, and so wire 9 in the interconnecting cable is also low. Then TCK changes to high, and the following happens: The output of IC1C goes high, but wire 9 does not go high immediately: The cable has a distributed capacitance and a distributed inductance, and together they form a characteristic impedance of 90 or 100 ohm, depending on whether you use 0.025" or 0.05" spacing ribbon cable. IC1C itself has an output impedance on the order of 20 ohm, so the output impedance with 75 ohm in series is roughly 90 ohms. So, instantly after IC1C goes high, we see, not VCC, but VCC/2 on wire 9 (the voltage divider between the driver impedance and the cable impedance). The VCC/2 step then propagates through the cable, until it reaches the pod (there is a current flowing into the cable, and there's voltage, so there's energy travelling down the line). The pod is high impedance, so the energy can't be absorbed, and is instead reflected back through the cable. To understand the effect, the result is identical to what would happen if the pod had a 90 ohm resistor to GND that absorbed the step, and then somehow generated an additional VCC/2 step of its own: At the pod end the signal doesn't change from 0V to VCC/2, it changes to VCC. Then when the reflected step arrives back to the driver board it's absorbed in the driver impedance, the output of the driver is now VCC, and the circuit idle. For this to work the cable needs to have reasonably good transmission line characteristics; To get this we need to have every other conductor ground, we *cannot* use the pinout of the JTAG connector.

Similarly at the pod end: When the target AVR drives TDO, it should be impedance matched to the cable (this is done by R9). R7 is there in the event that you forget R9 (really not recommended, but, it'll work): If the AVR manages to drive a 5V step into the cable, then when it arrives at the receiver, we'll get the same reflection phenomenon, and the cable will attempt to pull the input to 10V (or -5V if going the other way). This will cause the protection diodes in the AC32 to conduct, giving an input impedance of ~90 ohms, and absorbing the pulse instead of making it bounce back and forth across the cable. If you DO remember R9 then driving the cable consumes less than half the ICC current, and you get neat signal integrity.

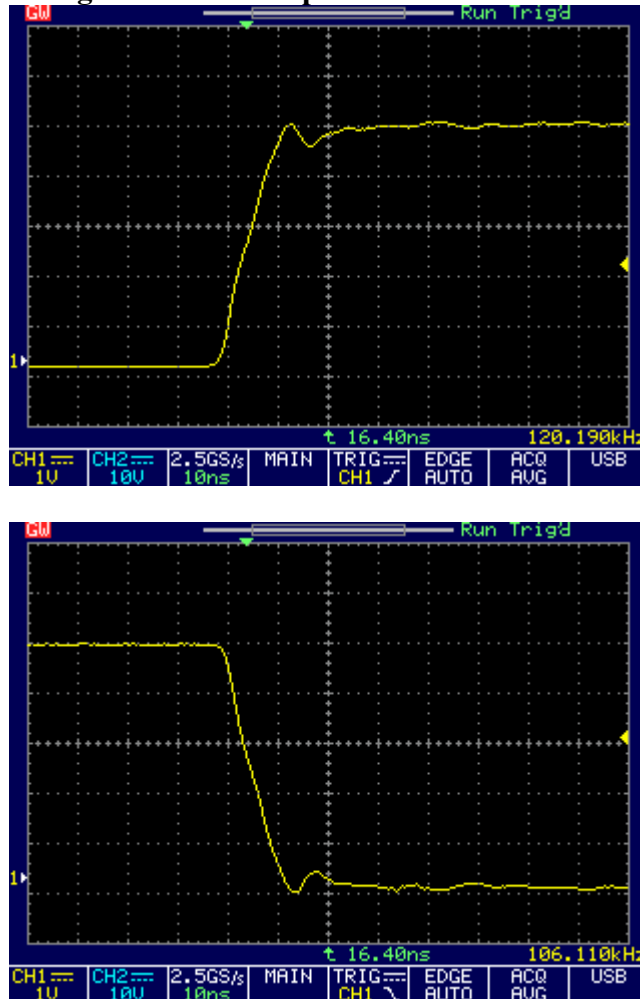
R10 is there to prevent the receiver from going into the linear region when the AVR isn't driving the TDO pin.

JTAG functionality was tested with 30cm, 1m, and 5m 0.025" pitch cable, and was found to work reliably.

Noise immunity

In this setup, if one end 'bounces up and down' because there's a noisy motor attached to it, a current will flow in the four parallel GND wires, giving a magnetic field around the ribbon cable. This in turn induces a voltage, in the same direction, on the signal wires; This cancels the noise, and the immunity is at least as good as with a JTAGICE connected to the target, and a lot better than a JTAGICE connected with flyleads.

Scope capture of TCK through 1m of 0.025" pitch ribbon cable



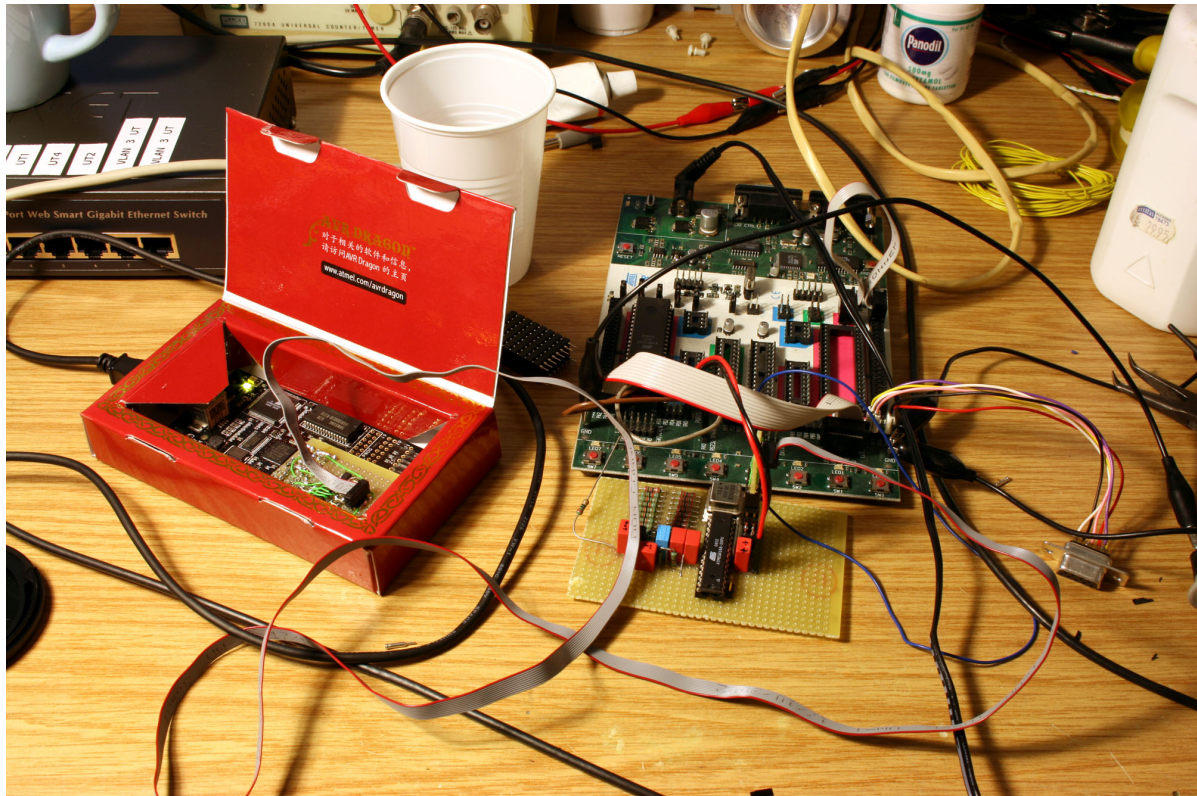
This is taken with a standard 250MHz x10 probe on STK500 with flylead ground on the probe, and 1m ribbon cable; When measured with a much faster (x20 resistive) probe the risetime is seen as ~12ns, and the little wiggle is gone. Not bad at all.

What I can't get the scope that can dump pictures to do is to show the crosstalk between TCK and its neighbour wire in the cable; The scope that _can_ capture this is my old analog, and it shows roughly 100mV worst case crosstalk, independent of whether a 1m or 5m ribbon cable is used.

Notes

There are three reasons for using AC32: It has beefier driver transistors than the HC series, and can thus take more punishment without being damaged (I discharged a SMPS power line filter into mine, and it survived), and they will operate well at 1.5V, so it'll work at any VCC voltage where the AVR operates. And on the input side there are two sets of protection diodes in parallel.

When debugging with DebugWire, make a spacer-connector where only GND, VTG, and RST are connected. Otherwise, if you use the SPI interface the AVR and the Dragon will fight and draw a lot of current, the resulting level on the VTG presented to the Dragon will be uncomfortably low, and DebugWire might not work reliably.



Wiring table for Dragon (this part is identical to JTAG-ICE MkII)

Pin	JTAG	ISP	DebugWire
1	TCK	SCK	(note 2)
2	GND	GND	GND
3	TDO	MISO	(note 2)
4	Target VCC	Target VCC	Target VCC
5	TMS		
6	/RESET (note 1)	/RESET	/RESET
7			
8			
9	TDI	MOSI	(note 2)
10	GND	GND	GND

note 1: Connecting RESET is only necessary if the code in the target sets the JTD bit; In that case it's needed in order to reprogram the target.

note 2: If the program being debugged uses the SPI pins, follow the advice AVRStudio gives when you enable debugwire, and make a go-between plug that disconnects these signals.

While the Dragon has a dedicated ISP header, the signals found on that header are in parallel with the JTAG header.

Revision history:

#1: posting of raw copy on avrfreaks

#2: rotated pin ordering on the connectors to match JTAG-ICE (the connectors were rotated 180 degrees)

#3: Nard Awater pointed out that the dragon has a separate reset wire for HV programming, and so the /RESET should be clamped to <7V.