Castle Link Live – Communication Protocol

Castle Creations, INC.

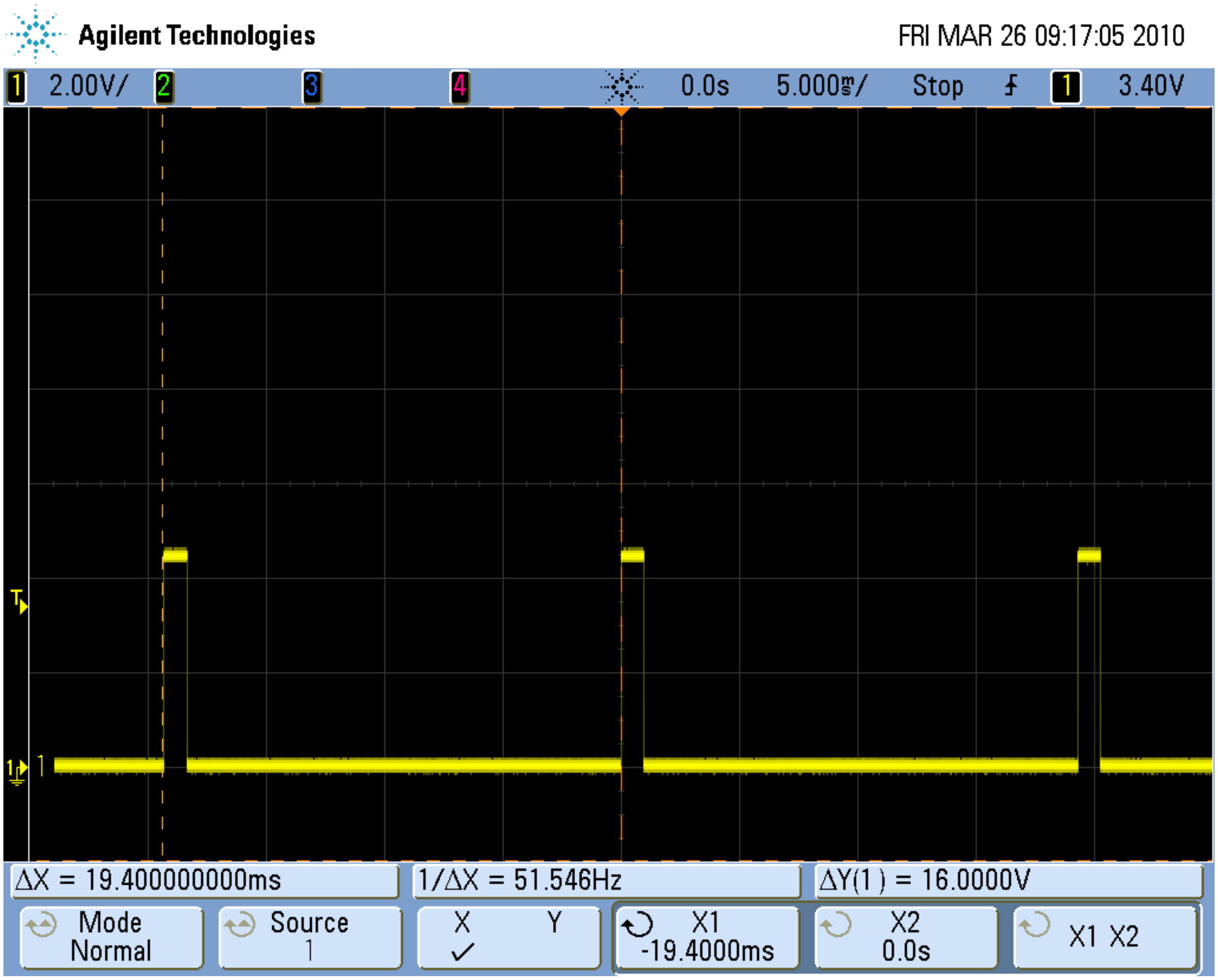
6-Feb-2012

Version 2.0

Subject to change at any time without notice or warning.

1) Standard RC PWM Signal:

Castle Link Live is a communication protocol that is very similar to the standard PWM signal used in RC control. The standard RC protocol, as pictured below (a), is simply a pulse repeating around 20ms (50Hz). The actual pulse frequency can vary greatly between manufactures but is generally close to either 50Hz or 100Hz. The throttle input (or servo position) is defined by the width of the pulses. In general 1.0ms is the minimum and 2.0ms is the maximum. All RC transmitters are different; some only make it down to 1.1ms while others can go all the way down to 0.8ms. Most speed controls deal with this via some type of calibration (automatic or otherwise) process.



ΔX = 19.400000000ms

1/ΔX = 51.546Hz

ΔY(1) = 16.0000V

Mode Normal

Source 1

X ✓ Y

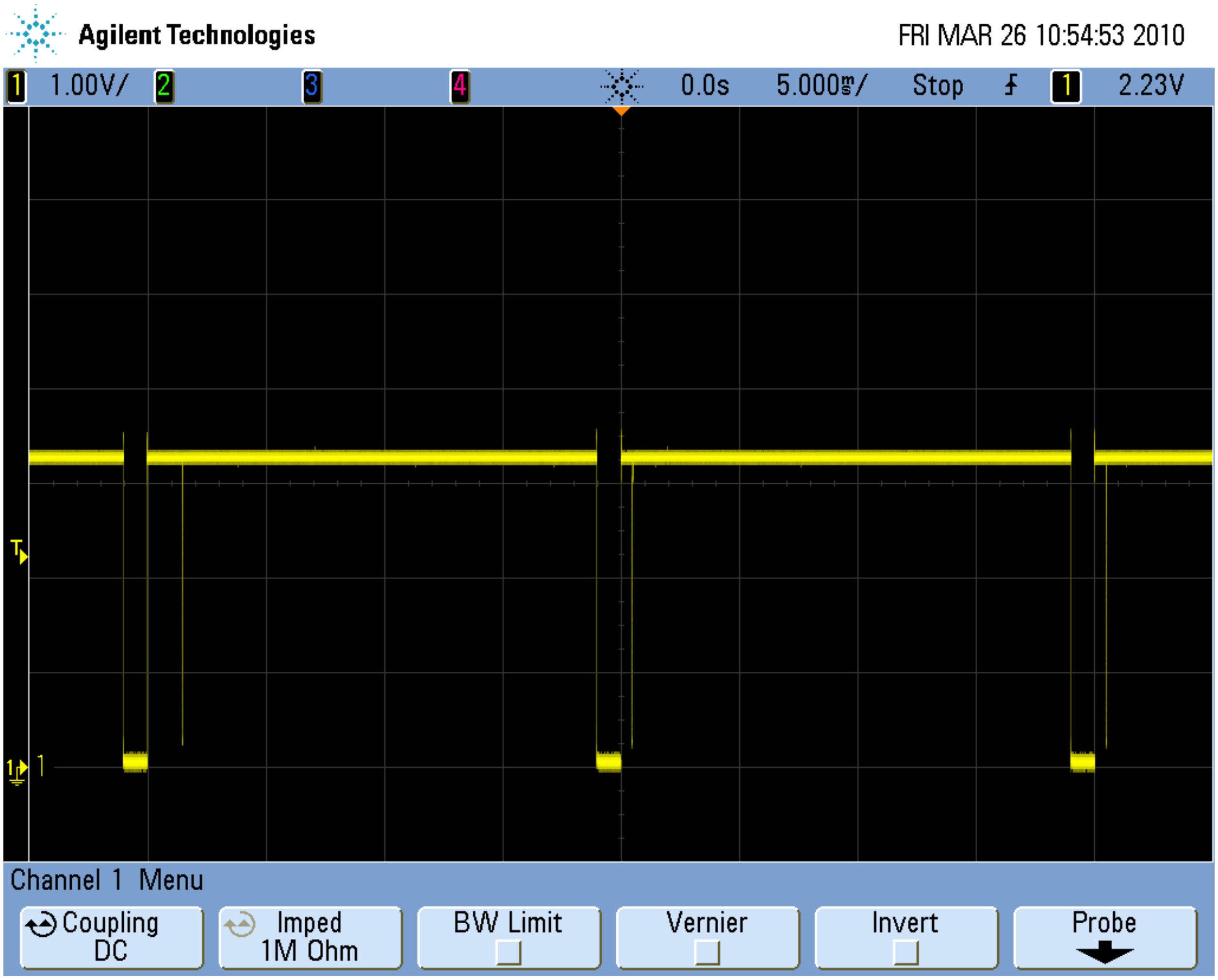
X1 -19.4000ms

X2 0.0s

(a) – Standard RC Pulse

2) Castle Link Live Overview:

Castle Link Live is an inverted form of the standard RC pulse. As you can see below (b), there is still a pulse every 20ms and that pulse still varies from 1.0ms to 2.0ms. The difference is the polarity of the pulse. Also note the negative tick just past the end of the pulse; this is how data is sent from the ESC. The time from the end of the pulse to the tick can be measured to determine the live data.



Coupling DC

Imped 1M Ohm

BW Limit □

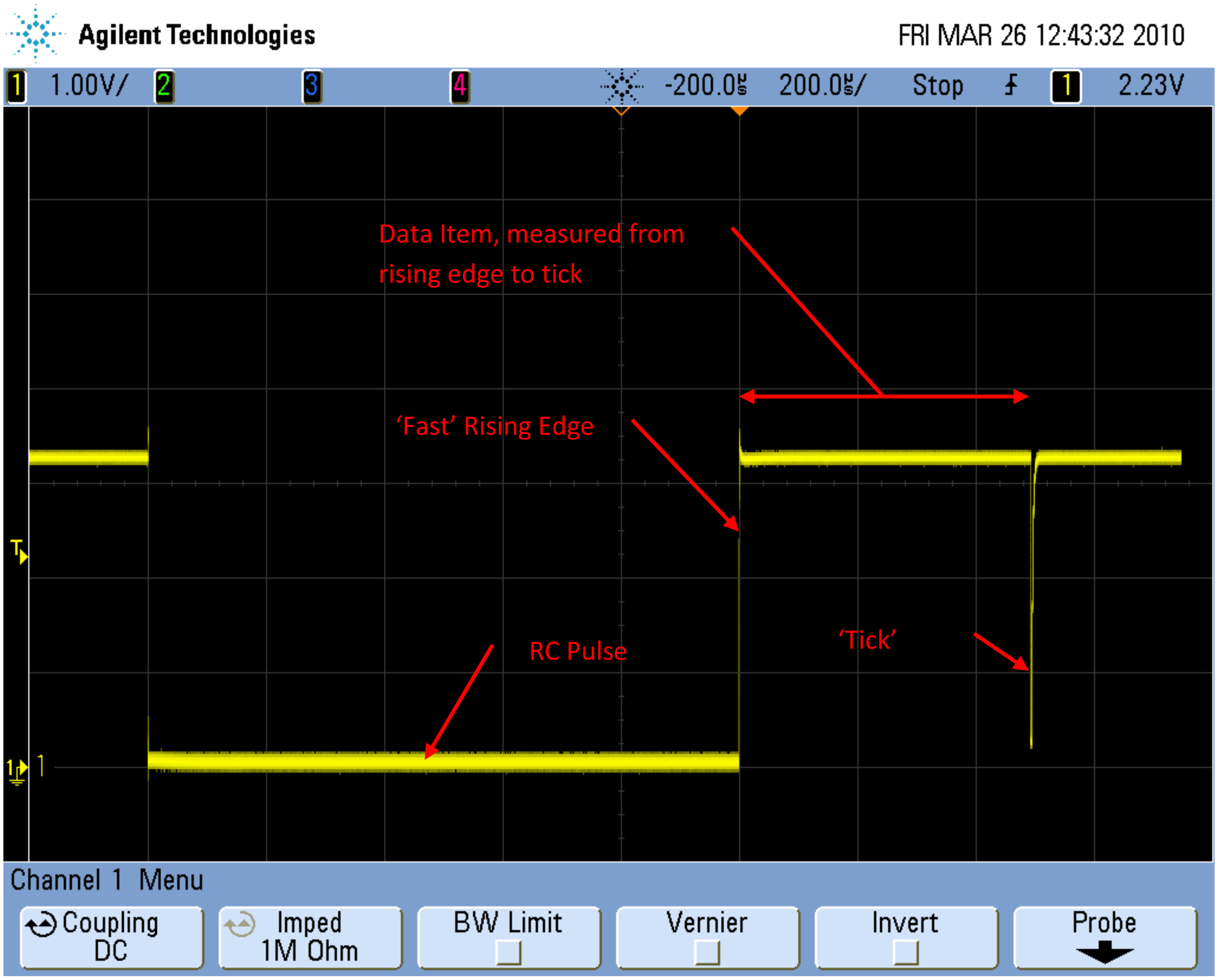
Vernier □

Invert □

(b) – Castle Link Live Communication

3) Castle Link Live Pulse Detail:

The value for the data item is determined by measuring the time between 'the rising edge at the end of the RC pulse' to 'the falling edge of the tick'. The image below (c) shows a 1ms RC pulse followed by a 0.5ms data item.



Data Item, measured from rising edge to tick

'Fast' Rising Edge

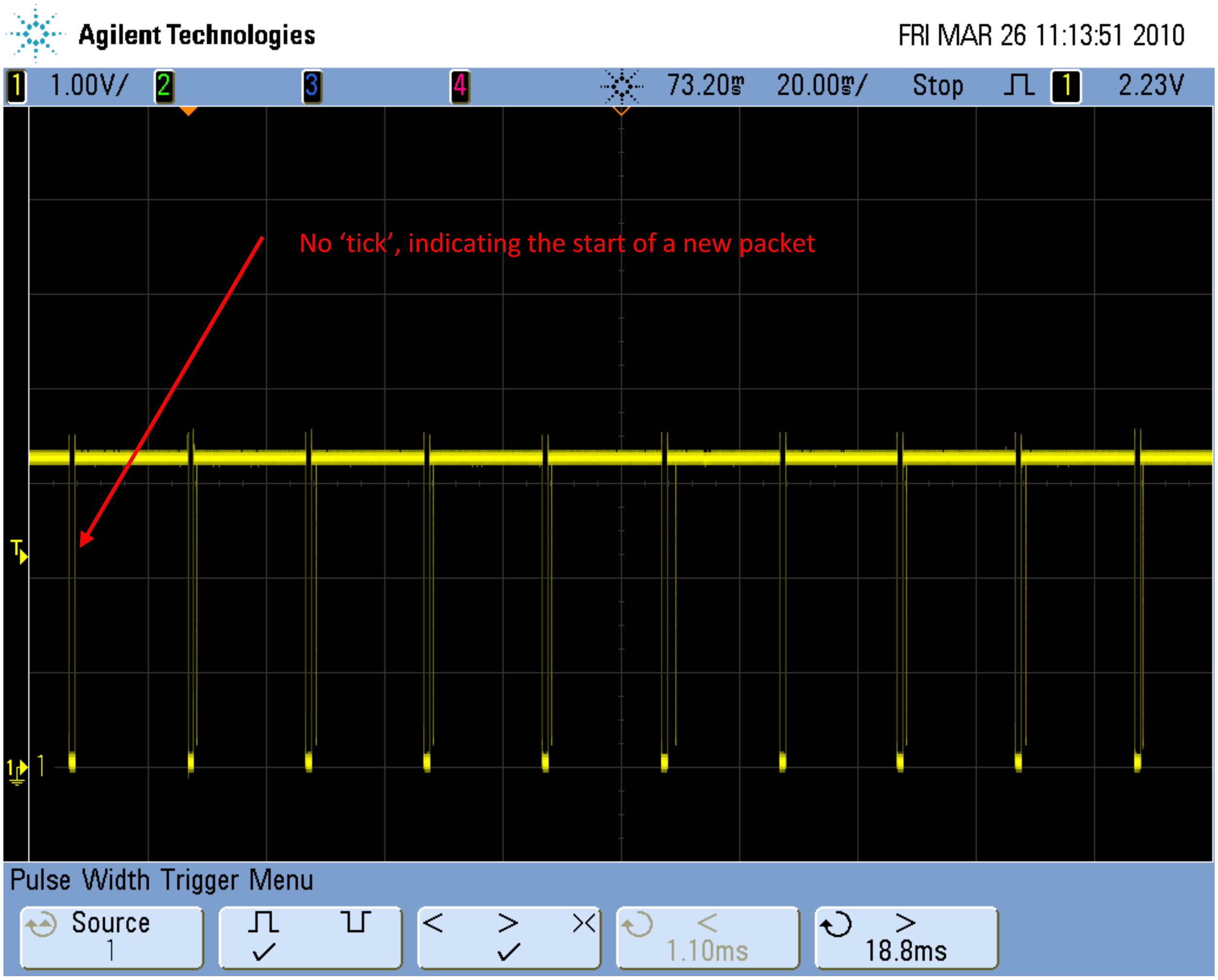
RC Pulse

'Tick'

(c) Example 1ms RC pulse followed by 0.5ms data item

4) Packet Description:

There are 12 pulses in this version of the Link Live data packet, each one containing a different data item (voltage, current, etc…). The picture below (d) shows the first 10 pulses in a packet. The very first pulse in this image does not have a tick after it; this is a RESET pulse to allow easy synchronization.



No ‘tick’, indicating the start of a new packet

(d) – First 10 data items in a Castle Link Live packet

5) Data Items:

The data in the 12 pulses is as follows:

1. **RESET** – No tick at all to allow easy synchronization with the data stream.
2. **Calibration** – A 1.0ms pulse to calibrate time scales, part of a possible 2-point calibration.
3. **Voltage** – The battery voltage measured by the ESC.
4. **Ripple Voltage** – The difference between the battery voltage measured during the FET on time and FET off time. There is no Ripple Voltage at full throttle because there is no FET off time.
5. **Current** – The current measured by the ESC. Not supported on some controllers.
6. **Throttle** – The standard 1-2ms RC pulse as measured by the ESC.
7. **Output Power** – The ESC's current output power percentage.
8. **RPM** – The current electrical RPM of the motor attached to the ESC.
9. **BEC Voltage** – The current voltage of the ESC's BEC. Not supported on some controllers.
10. **BEC Current** – The current amount of current that is being provided by the ESC's BEC. Not supported on some controllers.
11. **Temperature / Calibration** – The current temperature of the ESC **if** the ESC has a linear temperature sensor. A 0.5ms calibration pulse, part of a 2-point calibration otherwise.
12. **Temperature / Calibration** – The current temperature of the ESC **if** the ESC has a NTC temperature sensor. A 0.5ms calibration pulse, part of a 2-point calibration otherwise.

6) Calibration:

Because all oscillators have some amount of error and drift over their temperature range it is very important to use the calibration values to accurately determine the data fields. Data item 2 is a 1ms pulse and the smaller of 11 and 12 will be a 0.5ms pulse. With these two points you can determine an offset and scale to correlate the two clocks

7) Temperature:

Temperature is measured in one of two ways, and embedded temp sensor or an external NTC resistor. Only one of these two data fields will have valid data (greater than 0.5ms). If the controller is NTC type, logarithmic math is required to do the conversion.

8) Conversion:

The following table (e) lists the conversion factors necessary to turn the measured and scaled pulse length into an actual value.

Data Item	Scale	Units / ms	Max Pulse Length (ms)*	Max Value
Voltage	20	Volts	5	100
Ripple Voltage	4	Volts	5	20
Current	50	Amps	5	250
Throttle	1	Milliseconds	2.5	2.5
Output Power	0.2502	Percent	4	1
RPM	20,416.7	Electrical RPM	5	100,000
BEC Voltage	4	Volts	5	20
BEC Current	4	Amps	5	20
Temperature (Linear)	30	Degrees C	5	150
Temperature (NTC)	63.8125	Units **	4	255

(e) – Conversion Factors

The live value is computed by multiplying the corrected time by the scale. For example if, after scaling, the third data pulse in the packet measured 1.5ms:

The third data pulse is Voltage, the scale value is 20.
Subtract the constant 0.5ms from the measured value:

```
  Value = 1.5ms – 0.5ms = 1.0ms
```

The result is the simple multiplication:

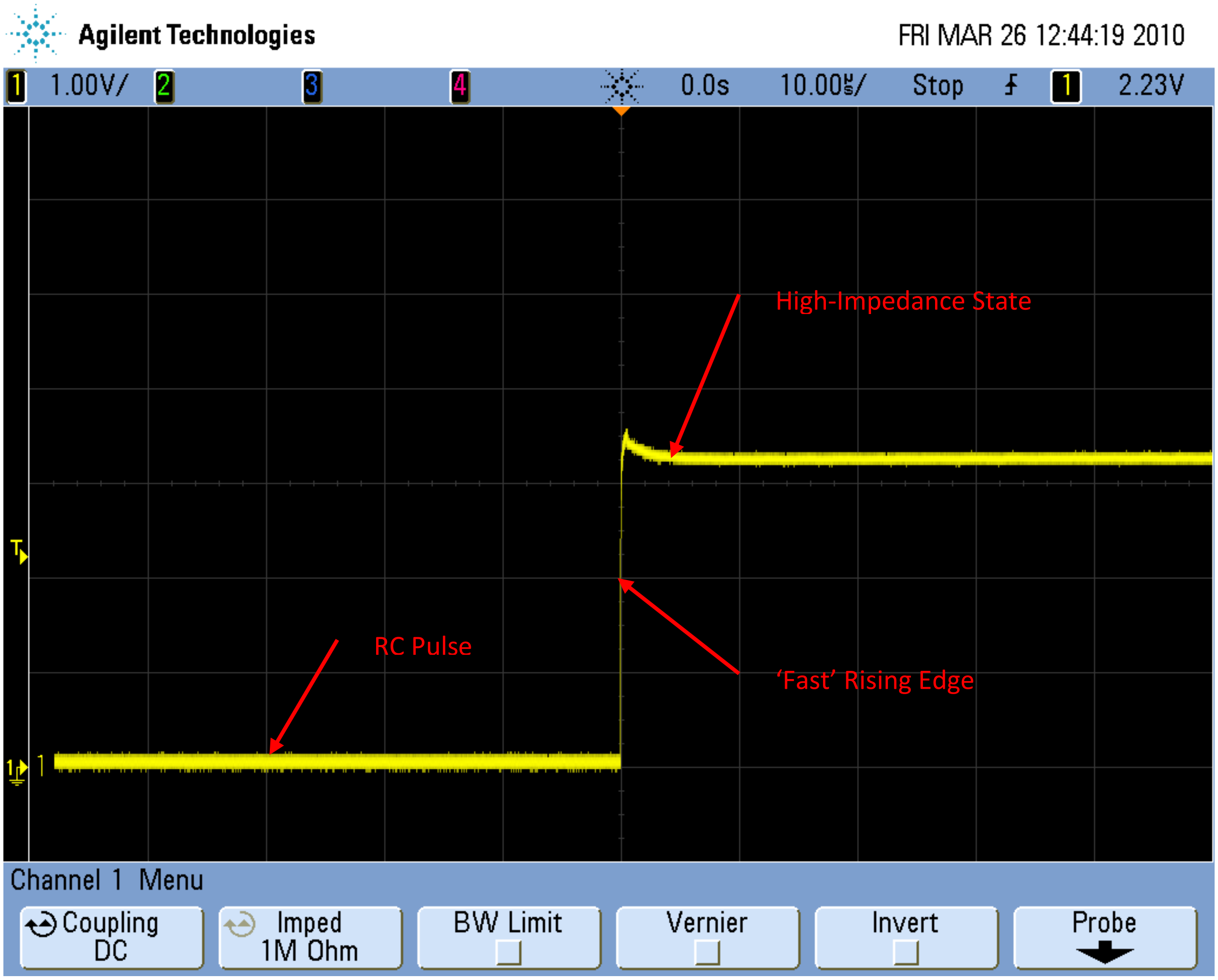
```
  Voltage = Value * Scale = 1.0ms * 20 = 20.0V
```

** Not counting the required 0.5ms delay after the receiver pulse. Every pulse will be at least 0.5ms, or in other words, 0.5ms is a value of 0.*

**** NTC Thermistor requires additional math to determine the temperature*
*Degrees C = 1 / (ln(value*R2 / (255-value) / R0) / B + 1/298) – 273*
R0 = 10000, R2 = 10200, B = 3455

9) Talking Castle Link Live:

To implement the Castle Link Live protocol care must be taken when generating the RC pulse. In any RC pulse type of application, both edges of the pulse should be driven with a totem-pole style driver circuit that is common to most microprocessors. This ensures fast edges that result in more accurate timing information. Castle Link Live has an additional requirement however, after the end of the pulse the receiver line must be placed in a high-impedance state so that the ESC can pull the line low later to make the 'tick'. The following image (f) shows the end of a pulse (rising edge) being pulled quickly to a 3.3V level and then released to a high-impedance state.



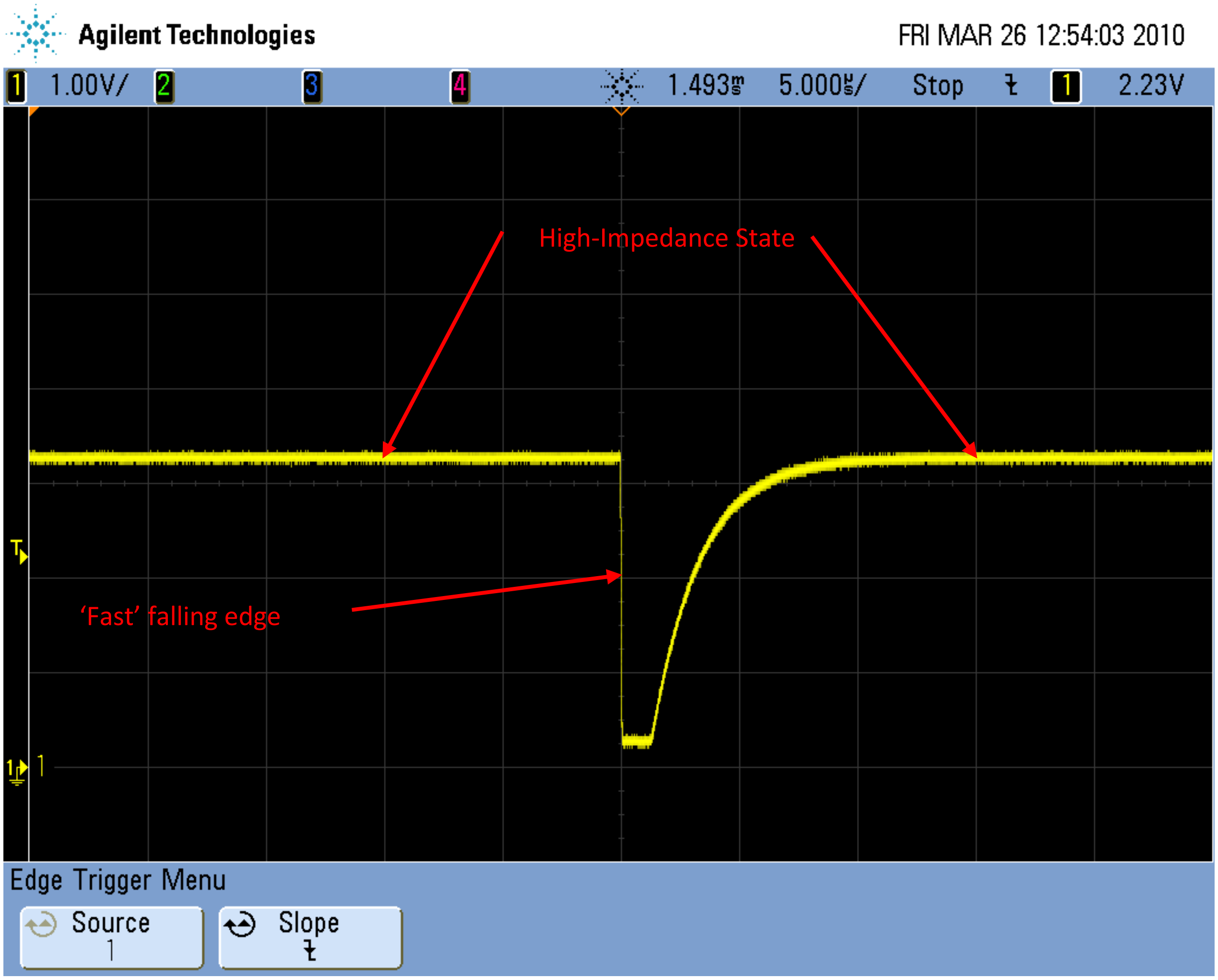
High-Impedance State

RC Pulse

'Fast' Rising Edge

(f) – The end of a Castle Link Live RC pulse, pulled hard to 3.3V then released to a high-impedance state

At the end of the data item, the controller will pull the line to ground for about 1.0us and then release the line allowing it to float back high. This 'fast' edge can be used for timing and as an external interrupt for most microcontrollers.



Edge Trigger Menu

Source 1

Slope

(g) – An up close view of the 'tick'

10) Hardware Considerations:

Castle Link Live requires the signal line be left in a high-impedance state by both the ESC and the controlling device at some times. During these times the line needs to float up to at least 2.5V to ensure that the ESC sees a logic (1). This will require a pull-up resistor on the controlling device, or on the connection between the controlling device and the ESC. The ESC has an internal pull-down resistor with a value of 6.65k, so, the required pull-up resistor will be somewhere in the 1.0k to 10k range, depending on the voltage being pulled-up too. Please reference the chart below (i).

Voltage Pulled Up To	Required Pull Up Resistor
3.3V	2k or smaller
5.0V	6.65k or smaller
6.5V	10k or smaller

(i) – Required Pull Up Resistor Based on Pull Up Voltage

13) Revision Log:

2.0 – 6-Feb-2012:
Initial Public Release