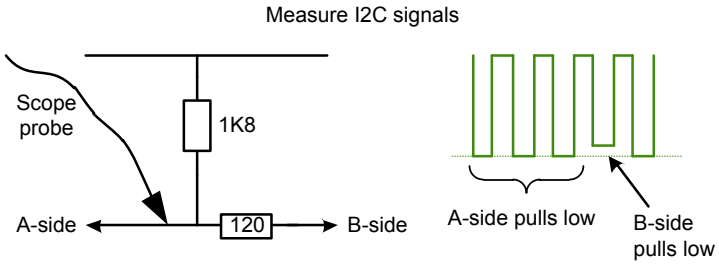
BCM2835 I2C interface bug

G.J. van Loo, 22 March 2013

There is a known bug in the 2835 I2C interface: The BCM2835 does not respond correctly if the I2C slave stretches the I2C clock.

Looking at the signals

When looking at an I2C signal (SDA or SCL) on a scope you have the problem that you cannot see who is controlling the line. If the signal is low either (or both) side can be pulling it low. To help with that I used the following measuring circuit:



The 1K8 resistor is on the raspberry-Pi board but I added a 120 ohm series resistor in each link. This causes a 200mV voltage drop. Now you can see which side is controlling the signal. It does not show you when both side are pulling the line low but you normally don't care about that situation.

Nomenclature

Write or Transmit: The Raspi sends data to the I2C slave.

Read or Receive: The Raspi receives data from the I2C slave. The Raspi still sends out the clock and the address byte. After that it switches to receive data but it only controls the clock. The data line is controlled by the I2C SLAVE.

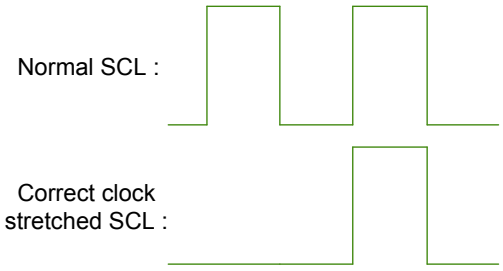
During the acknowledge phase the 'other' party takes control of the data line.

Checking the oscilloscope screen revealed why: suddenly there are I2C clock pulses which have a totally wrong duty cycle:

Description

What is happening inside the 2835? The I2C interface is running of a programmed clock. When an I2C transfer is active this clock is always running and it ***always*** has the same frequency/period. Thus for the 2835 all clock edges are equidistant.

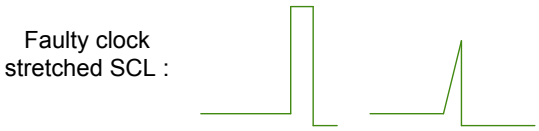
When the I2C slave stretches the clock it pulls the lock signal low. (Beware both sides can pull low. The signal goes high when neither the master not the slave pulls the signal low.)



or

(The second example of clock stretching may happen on other systems. You will not see a shifted clock like that on the BCM2835 because, as I said, on the 2835 all clock edges are at the same distance)

But we are NOT always seeing a correct clock. What is happening is that the 2835 stops pulling the clock low. From that moment on the clock polarity is controlled by the I2C slave and the pull-up resistor only. Thus as soon as the I2C slave releases the clock the signal will go high again. But the 2835 will blindly slam the clock down on the next clock edge time. Depending on when the I2C slave releases the clock you might see this:



Depending on how wide the pulse is the two systems may get out of sync: The BCM2835 sometimes sees the signal above as a correct clock and the I2C slave does not or vice versa. From that moment on everything bit transfer goes wrong until a stop condition appears.

A possible solution is to set the I2C clock frequency low enough that the clock stretching time is much smaller than the clock period. e.g. the clock is pulled low only 25% of the time. It greatly depends on the I2C slave if this works. I managed to get reliable data transfers between the 2835 and an Atmel chip using a 100KHz clock and extreme efficient (fast) interrupt handling. (Unfortunately all interrupt routines have to be extreme fast because on the Atmel device the I2C interrupt has the lowest priority.)

Ultimately you may have to write the I2C interface using 'bit banging'.

Todo

I still have to look into the I2C FSM. I have been told this only happen on the very first clock period when clock stretching starts. I have not been able to confirm that yet.