ThermIIC Series

Dr. Simon J. Melhuish

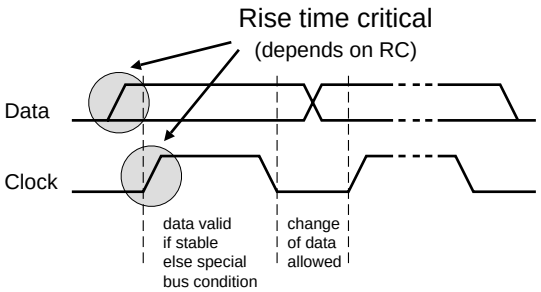
Part 4

Wiring it up

The I²C bus operates over two wires (not counting ground or any power lines). These carry a clock and the data signals. When the bus is idle both lines are held at +5 V, through pull-up resistors inside the computer. When the bus master wants to access a slave device on the bus it switches on and off transistors connected between the bus lines and ground. This way the slave devices “see” the bus lines switching between +5 V (high) and 0 V (low). If they receive their slave address they can respond by pulling down the data line themselves, sending binary bits back to the bus master. Notwwwe that the bus lines are totally bi-directional, since the master or any slave can affect the whole bus in any direction. In fact slave devices can even pull the clock line low. This is called “clock stretching” and can be used when a slave needs time to get its data ready − it forces the master to wait.

I²C devices rely on the relationship between pulses on the data line and pulses on the clock line to distinguish between various bus conditions. For example, if data goes high to low whilst the clock is high, it was a START condition. Used inside the computer this is fine. But when external devices are connected over long (more than a few metres) cables there can be problems.

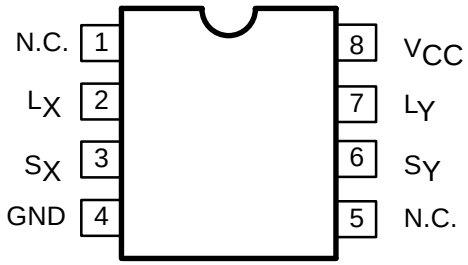
Both the cables and the I²C devices themselves have capacitance. When one device pulls a bus line low, the voltage at other devices does not go to zero instantly. This is because electrical charge is stored by all the capacitances spread around the system, and current must flow to discharge it. The same applies when the bus line is left to float back to +5 V, except this takes longer because the current flow is limited by the pull-up resistors. If it takes too long (more than about 1 µ sec) for the signal states to change you will get bus errors. If you notice the *ThermIIC* readouts getting greyed-out, this is probably the cause. Also, it can lead to errors on the real time clock value. Longer cables add more capacitance and make the problem worse.



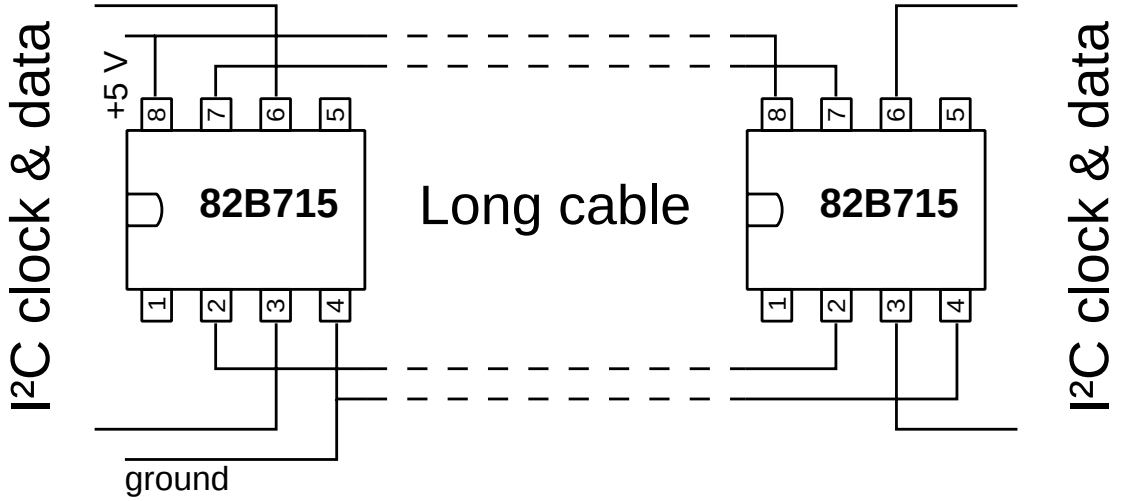
I²C bus timing

One thing that can help is to reduce the pull-up resistance by adding extra pull-up resistors from each bus line to +5 V. For example, adding another 4k7 resistor to each will approximately halve the resistance, doubling the current flow. This would reduce the time for the bus signals to change state.

Unfortunately the I²C bus specification sets a 30 mA limit on the pull-up current. I²C devices are not guaranteed to be able to pull the bus lines down if this is exceeded. It could even result in damage. Help is at hand from Philips Semiconductors in the form of the 82B715 I²C Bus Extender. These convert between a normal I²C bus and a *buffered bus*, with ten times the current flow. If a bus extender is used at ***each*** end of a long cable (it's no use trying to connect an I²C device directly to the ***buffered*** bus, because the currents will be too large for it to work), the effect of the cable's capacitance is reduced by a factor of 10. This allows the use of cables up to, typically, about 100 m in length.

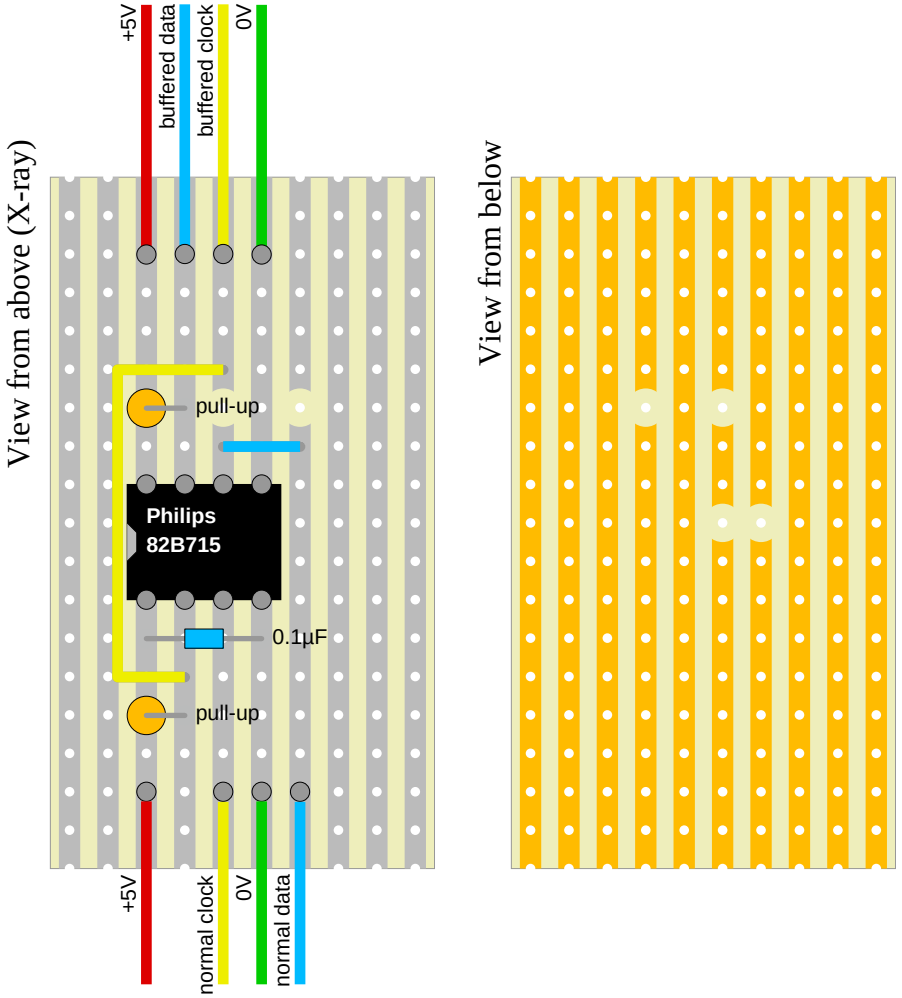


Pin	*Name*	*Function*
1	N.C.	no connexion
2	L_X	buffered bus − data or clock
3	S_X	normal I²C bus − data or clock
4	GND	Ground (0 V)
5	N.C.	no connexion
6	S_Y	normal I²C bus − clock or data
7	L_Y	buffered bus − clock or data
8	V_{CC}	+5 V



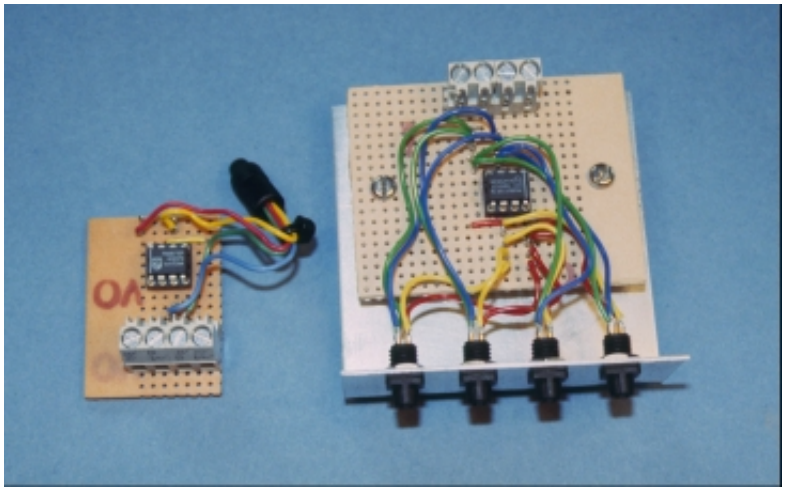
82B715 Pinout and connexion schematic

The circuit layout is pretty straightforward. Each 82B715 bus extender needs six connexions (two pins are unused). Two pins are for power and ground. Two pins go to the normal I²C bus (data and clock) and the remaining two connect to the buffered bus. Each end of the long cable must be connected to the buffered bus pins, making sure not to mix up clock and data. A decoupling capacitor (say 0.1 µF) should be connected across the power lines to each of the bus extender chips.



An 82B715 on stripboard − you'll need two of these

The circuit is almost too trivial to bother with making a PCB. The diagram shows a stripboard layout. Make sure that you break the tracks in the places shown, using a spot face cutter or similar tool. You might like to put the buffer unit in a box, especially for the remote end if it is going to be used outside. The unit that I use to monitor my telescope temperatures has also its own 5 V power supply, to power the remote bus. This way there is one less connexion to the computer, perhaps reducing the risk from nearby lightning strikes; a worthwhile consideration for an installation that is going to be left on all the time. It is necessary to share the same earth connexion, to complete the circuit, and so that “+5 V” means the same thing to both the computer and the I²C slave devices. Local earth potentials can differ significantly over the distance you might be running an extended bus.



A pair of bus extenders, one for each end of the link.

Even with the extended bus it might be necessary to experiment with pull-up resistances. The resistors can be connected the the buffered bus, the normal bus, or both. If connected to the buffered bus, resistance values must be reduced by a factor of ten, to get the same effect as if they were connected to the normal bus. For example, you could try adding 470 Ω pull-ups to the buffered bus. The technicalities are explained in the 82B715 data sheet, available as

`http://www-us.semiconductors.philips.com/acrobat/datasheets/u82b715.pdf`

That concludes the *ThermIIC* series. If you have web access please keep an eye on the *ThermIIC* page† for any updates. I'll be keeping a lookout for fresh developments on I²C, in case I can add anything useful in future. Happy soldering!

† *ThermIIC* page linked from `http://www.jb.man.ac.uk/~sjm/`