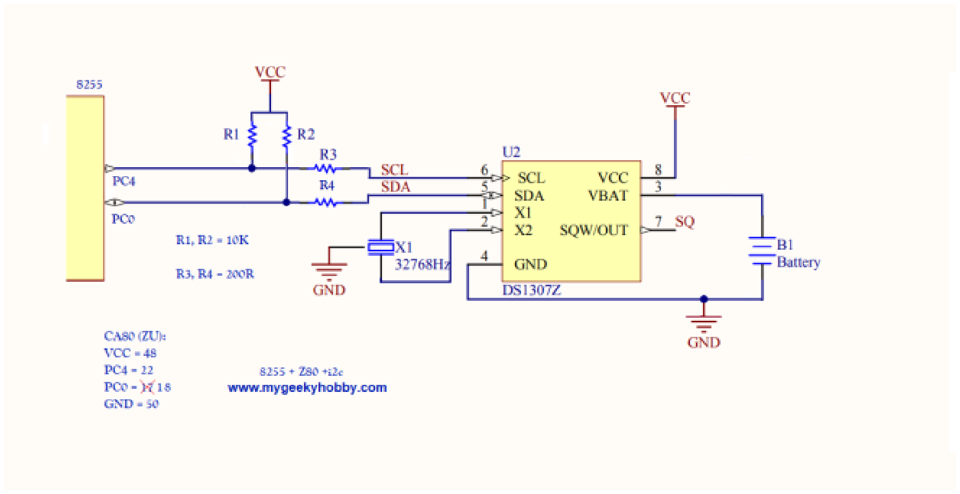
i2c bit-banging on Z80 with 8255A.

This article is mainly aimed at my Polish readers, owners of the educational computer CA80, but the i2c bit-banging for Z80+8255A may come useful on other Z80 platforms. As always source code can be found on my GitHub page.

8255 port control challenge.

There are many websites that cover i2c in detail, so I'm not going to spend any time doing that here, but in the context of this article, the key issue is the fact I'm trying to use a very basic "software-only" implementation of i2c – later used to pull date and time information from hardware real-time clock (HW RTC). I2c uses two lines to implement it's the bus: SDA (for data) and SCL (for clock). Those two lines have to be independently controlled via software, additionally, SDA is bi-directional, sometimes used as output and sometimes as input. The 8255 offers 3 ports (A, B, C) of 8 lines of general-purpose I/O connectivity. Unfortunately, the way it operates, all lines of each port can only be configured as input or output all at the same time. So if we "plug" our SDA and SCL lines to the same port (for example port B lines 0 and 1), we would not be able to implement the i2c bus this way. The only solution is to use two independent ports (for example port A, and B, or A and C). One more "trick" you can use is the fact that port C, is actually split in half and can be controlled independently (port 0-3 in one group, and 4-7 in another). This is exactly what I've done. Let's see the diagram:



As you can see, I've used PC.0 for SDA and PC.4 for SCL signals. R1 and R2 are pull-up resistors (I used 10k, but 4.7k would be ok as well) required for all implementation of i2c, R3 and R4 are used for current limiting, for situations where software might be misbehaving sending data at the same time other transmissions are in progress. In my example I'm communicating with a popular RTC from Dallas – DS1308Z (you can use a cheap module from eBay – be aware of issues with those modules where the "trickle charge" circuit may cause the re-chargeable 2032 battery to explode. This has happened to my module, so I removed all the extra components, and I used a non-rechargeable 2032 battery).

The GitHub repository contains the required code RTC_0x2000_0x2300_v1.4.asm The file contains a read and write procedures for the RTC. The code is a bit of a mess of stuff I found on other projects and my own code – but since I