Two Digital Clocks Using the I^2C Bus

Andy Palm N1KSN

Microcontroller chips are very versatile, but sometimes they need the help of other integrated circuits with whom they share data, such as additional memory storage, A-to-D converters, and real-time clocks. The I^2C bus is a "data highway" for such data communications. This bus (called the Two Wire Interface or TWI by Atmel) consists of two wires, called SDA and SCL. The former is the data line, the latter the clock line. So only two pins are required on the MCU chip.

To learn something about this bus, I decided to build a digital clock that uses two outboard chips, one a real-time clock (the 8-pin PCF8583) and the other a 7-segment LED display controller (the 24-pin SAA1064). I selected the Atmel Attiny2313 for the MCU, although something smaller would have been OK. I used a small 5V regulated supply for power.

The PCF8583 keeps the time and date, including leap year information. It also has additional RAM that can be used for general data storage. A 32.768 KHz clock crystal with a trim cap is the time base for the clock. (Why that frequency?) The clock values are stored in packed BCD (binary coded decimal) format. This type of chip is a way to add clock capabilities to a project without having to tie up the MCU with the chore of keeping time. But the MCU is still needed to set the time and date and retrieve the clock values for display.

The SAA1064 gets four digits from the I^2C bus and displays them on four 7-segment LED displays. This chip handles the display multiplexing so that the MCU doesn't have to.

To keep things as simple as possible, I used routines available on Atmel's website to "bit bang" the bus. This means that all bus protocol details are handled directly by software rather than a chip peripheral unit.

I added three push buttons. One of these is pushed to set the time and date, cycling through each part on additional presses. The other two buttons in this mode serve as UP and DOWN buttons. When the clock is in normal mode, these latter two buttons are used to display month and day, and minutes and seconds, respectively. More details are on the attached printout of program comments for asmTWIClock.asm.

The resulting clock works quite well and keeps good time. The most tedious part of the project was tweeking the trim cap. To try and avoid this chore I built a second clock that uses the 60 Hz power line to provide a clock time base instead of a crystal. I kept the SAA1064 display setup but used the Attiny2313 to keep the time and date.

This clock is powered by a 6 to 9V RMS wall wart. Besides the usual power supply components to get 5V DC for the chips and display, there is a tap off the AC input. This tap has a single diode rectifier feeding a voltage divider and 74LS14 Schmitt trigger. The 74LS14 converts the half-rectified AC into a proper 5V square wave for input to the MCU. The TIMER0 module in the MCU is configured to use the trailing edge of this 60 Hz signal as its clock. The timer repeatedly counts to 12 to get a 200 ms system tick, and so every five ticks equals one second. The clock variables are then updated each second to keep time.

Attached are the program comments from asm60HzClock.asm and full schematics for both clocks. The two clocks share the same display unit schematic.