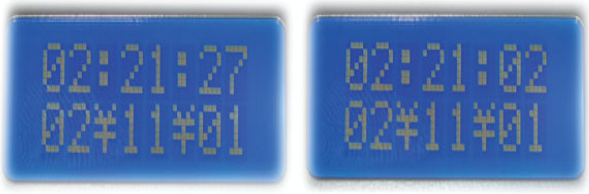
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create',
'UA-41457682-1', 'svet-el.si'); ga('send', 'pageview');



Some AVR microcontrollers have Timer2 which can operate in asynchronous mode. These AVRs are the ATmega8, ATmega168 and ATmega16 etc. With some help from simple Bascom code, one can make an accurate clock which will display date and time.



Bascom-AVR offers many “goodies”, one of which is certainly the RTC clock. I’ve met many Bascom programmers and seen many programs, where a real time clock was used. None of those programmers used the RTC clock that is described in this chapter. Programmers seem to love generating 1 second pulse with a timer interrupt service routine which increments the values of variables for second/hour/day etc. The sample code described here, is very practical, uses very little of the microcontroller’s resources and to top it off, generates an accurate clock. This program will perform well on both the MegaPin and MiniPin II development boards. It should perform well in your device too, providing that you connect it up accordingly. Possibly you might use an ATmega168P, which, according to its datasheet, promises very low energy consumption during sleep mode. That would make this microcontroller ideal for a low-power clock.

Software

The author of the Bascom software, Mr. Mark Alberts, has already written a basic sample program Megaclock.bas that can be found in the Program Files/MCS/Bascom-AVR/Samples folder. The main part of that short program is this statement:

CON-FIG CLOCK = soft /USER [, GO-SUB = SEC-TIC]

This statement is thoroughly described in the Bascom-AVR Help. For our use, the statement will look as follows:

Con-fig Clock = Soft

With this statement, we have set all of the necessary registers in the AVR to allow Timer2 to work in asynchronous mode. Place an ATmega32 (or any of the above-mentioned AVR microcontrollers) on the MegaPin (or MiniPin II) board and set the fuse bit to Internal RC – this is important! Place a 32,768 kHz crystal into the appropriate socket (J18 at MegaPin, J7 at MiniPin II), disconnect jumper (J5 at MegaPin; J3 at MiniPin II) on PortC. Then load the program into the microcontroller and our RTC clock will work.

The Megaclock.bas sample shows to us how to set the date/time using the following statements:

Da-te$ = “11/11/00” Ti-me$ = “02:02:00”



This statement is rather impractical, as it will set the date/time contained in the two strings, not the current date/time. Therefore we need a manual way to set/change the date and time. We can do this by (individually) changing the following internal variables:

_day , _month, _year , _sec, _ho-ur, _min

We can now display the date and time on an LCD:

```
Lcd Da-te$ ; “  “ ; Ti-me$
```

Actually, apart from the clock configuration statement described earlier, the whole program is very short:

```
Do    Ho-meUp-per          Lcd Ti-me$    Lowerline    Lcd Da-te$          Powersave   Lo-op
```

What you don’t see above, is that, in the background, all of the necessary variables are being updated, by an Assembly language procedure (contained in an interrupt service routine), generated automatically by the Bascom compiler.

Can the generation of a precise clock/calendar be much simpler than that? I don’t think so! Of course one must add routines for initially setting the date/time and possibly routines for an alarm. You can even make yourself a unique clock with multiple alarms, at different times on different days.

Simple RTC - Real Time Clock

2012_AVR_UK_191



Continue reading

Shop area



