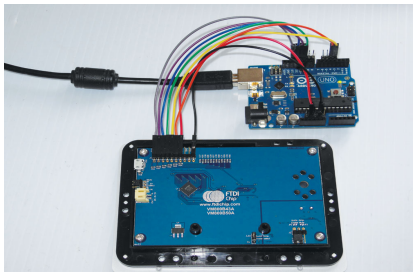



svet ELEKTRONIKE

## Basic Hook-up and Programming

In Part 1 of this series I covered the basic features and advantages of the FTDI FT800 EVE graphics controller chip. In doing so, I mentioned several other methods of obtaining color TFT touch-screen functionality, including both “dumb” TFT displays that interface to your MCU via an 8/16 bit parallel port, as well as “intelligent” TFT display modules such as those sold by 4D Systems. I feel that FTDI's EVE solution is both cost-effective and lends itself well to being interfaced with modest 8-bit MCUs, like the Atmel Mega328P that is found on many of the popular Arduino boards (i.e. the Uno). This month I'm going to show the reader how to get started using some of the EVE-powered TFT touch-screen modules that are currently available.



Before choosing which display module that you wish to purchase, you should look back at Table 1 in Part 1 of the series, to determine which of the three company's modules best serve your requirements. When I first read about the FTDI EVE controller, Mikroelektronika made the only EVE-powered TFT touchscreen modules available:  the 4.3” ConnectEVE. So, I started out by

purchasing a few of these and designing projects around them. Shortly thereafter, FTDI themselves started selling two different series of evaluation modules: the VM800B series that comes with a mounting bezel and full-size control PCB, and the VM800C series which contain a credit card-sized controller PCB and can be purchased with or without a TFT display. The VM800C modules come without a bezel, and the TFT display has no mounting tabs either.

In this article I will use one of the FTDI 4.3” VM800B modules to demonstrate the operation of the EVE controller. Later in the series, I'll describe one of the projects I designed/built using the somewhat more compact Mikroelektronika ConnectEVE modules ( the first ones I purchased).

## Module Hook-up

If you think like me, the physical hook-up of TFT display modules to your favourite MCU is probably the least of your worries. My prime consideration is how difficult it will be to write or obtain the necessary software drivers to interface such display modules with the Atmel AVR MCU family that I generally use. This consideration is further complicated by the fact that I generally write programs using Bascom/AVR and not C/C++. While I hate to admit it, it appears that the majority of drivers for peripheral devices/chips are supplied by the vendors in the form of C/C++ libraries. Part of the reason for this is that AVR-based Arduino boards are extremely popular and the Arduino is programmed in C/C++. Arduino programs or “sketches” are basically C/C++ code which has been “wrapped” by a user-friendly IDE which hides a lot of the complexity of C/C++.




In the case of the FT800 EVE controller, FTDI decided to write both their drivers and example programs in C/C++. The Arduino compiler will handle this just as well as it will handle “sketches” written in the Arduino simplified C format. If you are accustomed to writing Arduino “sketches” but are not a C/C++ expert, you will find the FTDI drivers and example programs rather hard to follow. As a long-time Bascom/AVR programmer, this C/C++ code was even more difficult for me to follow! Therefore, I decided it would be too daunting a task to convert all of this code to Basic, and instead decided to join the large world-wide Arduino “club”, and write code for my FTDI EVE-based projects as Arduino “sketches”.

To save you from having to “re-invent the wheel”, what I have done is carefully go through FTDI's example code, remove as much extra code as possible, and simplify what remained as much as possible. If a Bascom/AVR fan like myself can successfully use these FTDI EVE modules in projects, you should be able to do so as well!

So, let's see how we can hook up our EVE-based display to an AVR Mega328. The interface is just a standard SPI interface, with a couple of extra lines to handle the board Reset/Power-down and an optional interrupt. Depending upon which type of module you have, the logic levels required will be either strictly 3.3V or 3.3/5V switchable. See Table 1 in Part 1 to determine which is applicable to your board. If you have a 3.3V only module, but are using an MCU powered by 5V (i.e. an Arduino Uno), then you will have to use some level-shifting circuitry. I described a simple resistive logic level converter in Figure 1 of Part 1, which is simple and works well for this purpose.



**Download program**



**Download article**

The FTDI EVE graphics controller (2)

2014_218-31