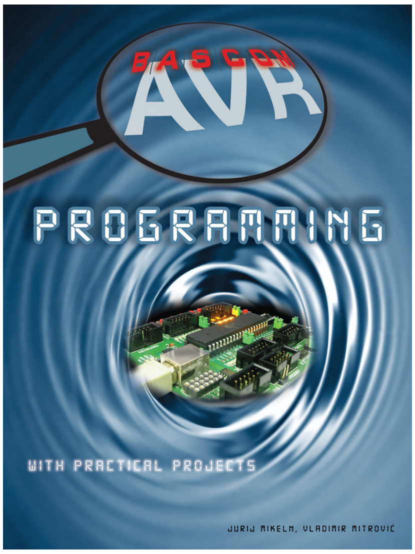
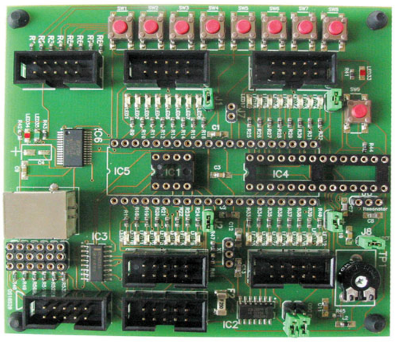
```
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga');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



We are witnessing great progress in all areas of technical science. Much of that progress we can attribute to the development of processing power in our PCs, which have enabled development of new powerful programming tools running on our PCs. In the field of electronics there are many tools that we use daily and which are indispensable to our work. The same goes for electronics components, especially microcontrollers, which have progressed tremendously in the last 15 years.

This progress can be seen in both lower component prices and better technical specifications. Today, a 32-bit ARM microcontroller, with extensive peripherals like A/D (analogue to digital) and D/A (digital to analogue) converters, Ethernet, CAN and USB interfaces built in, can be bought for few Euros. However, we should clarify a few terms: in a PC there is no microcontroller, but a powerful 64-bit (possibly multi core) processor which sends and receives signals to various peripheral units such as video, memory, hard disk drive etc. The difference between a microcontroller and a processor is basically in the peripheral units: a processor does not have embedded peripheral units.

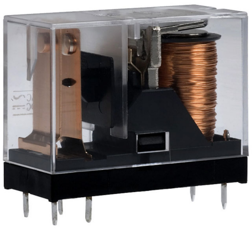
This book will not deal with ARM microcontrollers, nor with processors but will outline the basic information which will be a foundation for your AVR microcontroller programming. Typical AVR microcontrollers will drive motors, relays, receive signals from sensors and display data on LCD display, for example. Further on in this chapter we will show the basics of how to program microcontrollers. Upon completing the last chapter, you will know how to write a microcontroller program on your own. We hope that this knowledge will encourage you to develop an understanding of microcontroller programming.



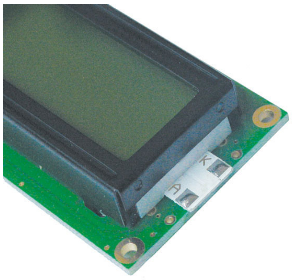
Microcontrollers, processors, controllers

Let’s try to define the differences between the terms microcontroller, processor and controllers. The term controller is normally used when referring to a PLC controller, i.e. a controller like Simatic etc. used for industrial automation. A PLC controller is therefore a device enclosed in a box equipped with I/O (input/output) connectors, and sometimes also containing buttons, LED or LCD displays etc.

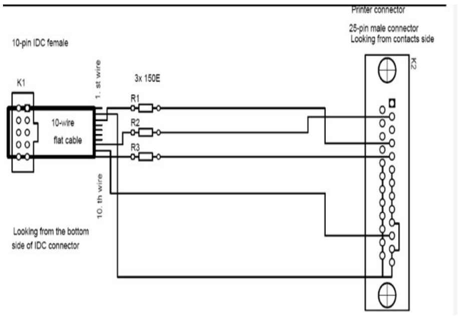
The term processor has already been described as a chip having no built-in peripheral units such as A/D or D/A converters. A processor receives data, and performs computations on them within a powerful arithmetic logic unit (ALU) core (or cores) and then sends them back to the bus. The term microcontroller refers to a chip which includes a modest processor (compared to a PC processor) and a rich set of built-in peripherals.Microcontrollers can be characterized by the number of bits in the microcontroller’s core: 4, 8, 16 and 32 bit ones are the most common. The microcontrollers that we will describe in this book are 8-bit ones.

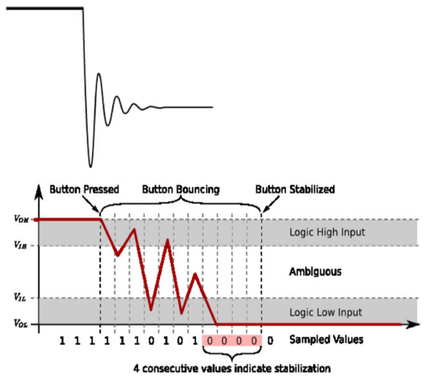


AVR microcontrollers are one of the popular microcontrollers available over the last 10 years or so. Because I want to describe a microcontroller as simply as possible, I will refer to it as a “black box”, having several I/O ports. A microcontroller with no program yet installed in it is like a “dead bug” i.e. it will perform no useful function. Besides this program, a power supply voltage is needed for a microcontroller to start performing its task. Microcontrollers are currently being used in virtually every device: be it a toy or a PLC controller. In a modern car there are numerous microcontrollers used for both engine control, safety devices and occupant conveniences such as entertainment systems and GPS.



AVR microcontrollers incorporate an ALU (arithmetical logical unit), more I/O ports, FLASH memory and other units that will be described later (see Figure 1). With a suitable programmer, we can program its FLASH memory with our own program, which will then be executed by the ALU step-by-step. I/O ports are bi-directional, which means that they can function as either an input or an output. The port’s data direction must be defined by our program. As mentioned earlier, a microcontroller needs a power supply voltage to operate. Additionally it needs an external crystal or ceramic resonator (although this function is commonly built-in to newer microcontrollers). The crystal or resonator, along with an internal oscillator circuit generates a clock for the ALU. Lacking a clock, the ALU stops executing the program. If the clock is restored, the ALU starts executing again. Depending upon the microcontroller, it may not start back up where it left off. Most AVRs contain an internal RC oscillator which generates the clock needed by the ALU, meaning that no external crystal or ceramic resonator is needed. Selection between the internal RC oscillator and external crystal/resonator is done during programming, and will be described later in this book.





The AVR program being executed is actually machine code instructions, which “tell” to ALU what to do. Since the ALU operates on binary numbers, coded with zeroes and ones, each instruction is made up of zeros and ones. When microcontrollers were first introduced, this machine code language was the only way that programmers could write their program code. Because this was very inconvenient, programmers soon switched to assembly language (using an Assember program). Even this was cumbersome, so today high-level compilers such as C, Pascal and Basic are more commonly used. Such high-level compilers translate your program into machine code. Among Basic compilers, a notable one is Bascom-AVR, which has both good and bad points. However, I believe that Bascom is an ideal compiler for both beginner and professional programmers, as it combines the advantages of the high-level Basic language along with the ability to embed assembly language in your program.

Short introduction to programming microcontrollers with Bascom-AVR

2012_AVR_UK_11

Continue reading

Shop area