Designer's Corner

Designing for Efficient Production with In-System Re-programmable Flash µCs

By: OJ Svendlsi

For products where time-to-market and efficient production is important, selecting the right microcontroller architecture plays a big role. To get the shortest possible development time, the following requirements should be filled:

- Good and easy to use development tools
- Enough resources on-chip to meet requirements
- Efficient for high level languages
- Flash program memory for fast and reliable programming

The megaAVR family from Atmel has all these features and more, making them a perfect choice for advanced products requiring short time-to-market.

This white paper discusses the advantages of this family related to getting the shortest time-to-market and the most efficient production once the product is ready.

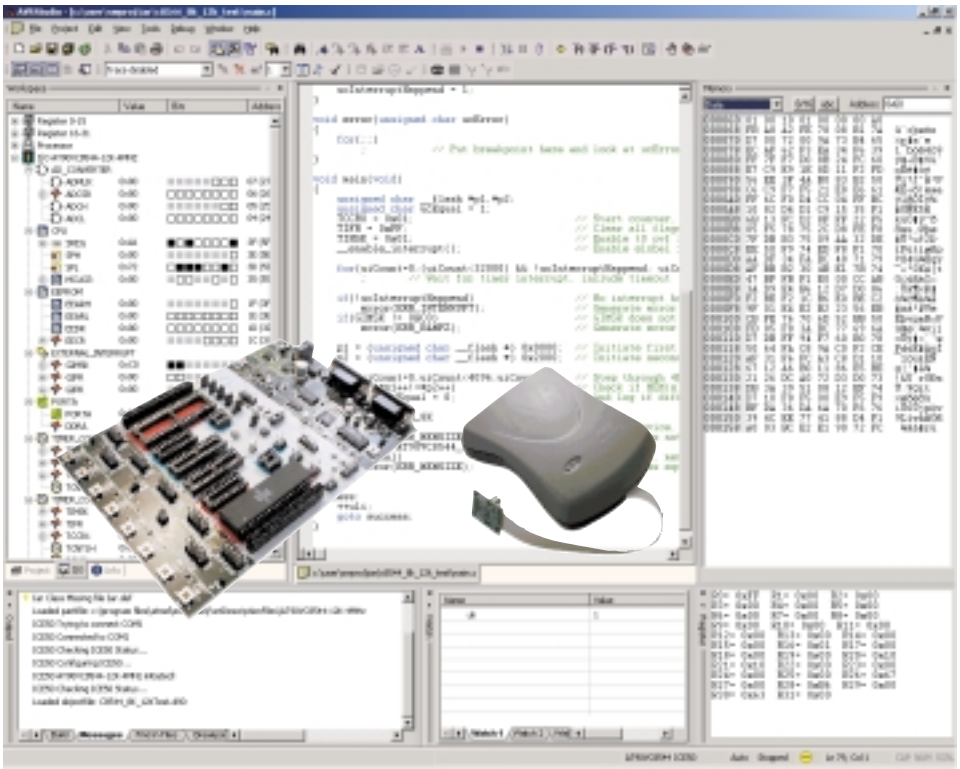
Development

During development of a new product, the microcontroller is normally the most important module, and always the component where the majority of the engineering hours are spent. Thus, making sure the microcontroller has what it takes to ease the development should be given some priority.

Selecting the Right Development Tools

When selecting a microcontroller, you are also selecting the development tools you are blessed with. These development tools range from the simplest evaluation kits to high-end In-circuit emulators and production programmers. Depending of the complexity of the end product, different debugging solutions will be optimal. A 1-2K microcontroller with a minimum of on-chip resources can often be debugged using a free architectural simulator such as AVR Studio. For complex applications on big parts with many peripherals, using an advanced In-circuit emulator with advanced features such as Trace capabilities, advanced breakpoints and Code coverage functionality might reduce the design time significantly.

Parts with on-chip debug capability offers a very low cost alternative for the debugging. The JTAGICE mk-II from Atmel is available for only $299, but still provides debugging capabilities like breakpoints, data watch, single stepping through code and full overview of processor internal resources. An additional important



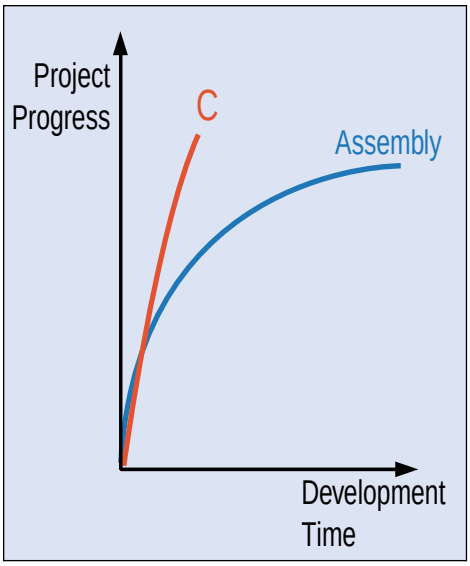
advantage of on-chip debug is that the debugging is done on actual production silicon. Because of this, many of the problems designers run into when switching from an emulator platform to the real thing is eliminated.

Starter kits are useful when evaluating a microcontroller that might fit in the application. Many starter kits can also be used as target hardware during initial code development before the target hardware is ready.

Programming in a High Level Language

By programming the microcontroller in a high-level language (HLL), like for instance 'C', it is possible to reduce the development time significantly compared to writing in assembly. Generally one can say that an experienced designer can write the same amount of lines of code per day in C and assembly. However, the code-lines written in C will do much more than the same number of lines written in assembly.

Typically, a program written in a high level language will also be much more structured than a similar program written in assembly. Because of this, it is generally easier to debug a program written in a high-level language.



Most 8-bit microcontroller architectures come with a C-compiler. However, there is a big difference in how efficient the architectures are for high-level languages, and how the C-code should be structured to be efficient with one particular microcontroller architecture. Generally one can say that accumulator-based architectures like the 8051 architecture from Intel works best with global variables, while register-based architectures like the AVR from Atmel works best with local variables.

The benefits of using local variables are that the code becomes more structured and the portability and maintenance of the code is simplified when compared to code written with very many global variables. It is also much easier to reuse code when it is written with extensive use of local variables. When all the parameters going in and out of a subroutine is defined in the function call, it is very easy to port that subroutine into a new project.

The biggest drawbacks of writing code in a HLL are that the code normally becomes bigger and slower than a similar program written in assembly. However, as the number of code-lines increases, the gap in size between the HLL code and the assembly code starts to shrink. For a typical AVR user, the crossover point where the HLL and assembly code is the same size is around 4K. However, the HLL code will almost never be faster than the assembly code. If execution speed of a certain part of the program is critical, the solution is often to write the code for the critical parts in assembly, and write the skeleton and less critical subroutines in HLL.

Integration

The level of integration in a microcontroller can also affect the development time, in addition to power consumption and board space. Integrated functions such as Brown-out protection, Watchdog timer and Power-on reset circuitry gives the most reliable operation of the microcontroller, while functions like integrated EEPROM, internal RC oscillator and strong push pull port drivers eliminates external components and reduces cost and complexity of the design. With fewer external devices on the PCB, the possibility of running into noise related problems is also reduced. Especially the internal RC simplifies the design from a noise tolerance point of view.

In-System Programmable Flash Program memory

Having In-System reprogrammable flash memory simplifies the development of a microcontroller application significantly compared to ROM/OTP solutions. The devices can be soldered into the application, and then be reprogrammed when a problem with the code is found. Using the AVR microcontrollers, the reprogramming can be done either through the SPI interface, or if using a JTAG on-chip debug or programming tool, through the integrated JTAG interface. If using a boot loader with the self-programming memory, other communication channels can also be used to reprogram the devices. The non-volatile memories of the AVR devices can be used to store history when debugging a program. The contents of the memories can then be read out after the program is finished.

Production

The advantages of using modern flash microcontrollers do not stop when the development is completed. Also when in full volume production, the benefits are many as outlined here.

Procurement

When using Flash microcontrollers, it is much easier to handle the procurement of the products. It is often easier to handle surprise orders since the microcontroller used is a standard line item. The chance that other companies are using the same microcontroller is usually big. This makes it possible to stock the microcontroller both at the distribution level and at the device manufacturer. If the same microcontroller is used for a number of applications, inventory can be moved from one product over to another to maximize the revenue.
Handling multiple versions

With a Flash microcontroller the same device can be used to handle multiple software versions. In many products, the only difference between a high-end and a low-end version of a product is the firmware. In these cases, a flash microcontroller is ideal. The PCB can be assembled and completely tested in advance. Once the order comes in for a specific product, the code for this product can be programmed in, and the product shipped to the customer.

Many larger manufacturers are selling the same product to different OEM customers. In many cases, the differences between the products shipped to these OEMs lies in the software. Again, an In-System reprogrammable flash microcontrollers can be programmed with the correct code at the production line. Alternatively, the OEM account can program the parts of the code that is individually changeable by using the boot-capability of the megaAVR devices.

Calibration

Many analog sensors have a relatively large offset error that needs to be calibrated out to achieve good measurements. Devices with integrated non-volatile data memory and high performance Analog to Digital converters are ideal for this. With the AVR microcontrollers, special calibration software can be used to run the calibration and store the calibration values in the internal EEPROM. After this, the main code can be programmed in, and use the values stored in the EEPROM.