



---

Microcontrollers: options and trends  
in today's applications

Few decisions in an engineer's working life can be described as genuinely 'emotive'. But one area where strong opinions are always expressed is in choosing a microcontroller. Most designers have their own favourite architecture; often, suggesting a change is tantamount to heresy.

From one point of view such attachments are pragmatic as well as emotional. Often an engineer will be able to solve design problems amazingly quickly by relying on an intimate knowledge of a particular microcontroller architecture and instruction set: and on an accumulated stock of routines and algorithms developed for previous projects. Choosing to switch architectures, on the other hand, incurs cost and time penalties as the design team gets up the learning curve. Such considerations have contributed to the success of the most common microcontroller and core architectures, such as the 80C51 and ARM core families.

There are two factors that can counterbalance this natural inertia. The first is the performance and scope of the feature set of the current "favourite" architecture. If a particular family cannot deliver the processing performance required by a new application, or does not have a variant with the right combination of peripherals, then there is little choice other than to look elsewhere for a solution.

The second factor is the increasing trend towards programming in high-level languages – and particularly C – which allows some portability of code and libraries and makes it more practical to consider using a different microcontroller family.

Whatever the motivation for change, the ease with which existing intellectual property can be ported to the new architecture should be considered. This means ensuring that the new architecture has the right development tools and support. The cost of such support has tumbled in recent years and today, it is taken for granted that microcontroller suppliers will provide at least entry-level tools: but it remains important to check the availability and quality of third-party tools such as C-compilers, emulators and programmers. Help-line and after-sales support may also prove invaluable: it is worth selecting a distributor which offers access to field applications engineers who know the products intimately, and are regularly trained to keep them up with latest developments.

If portability of previous applications is not an issue, the design team is effectively starting with a blank sheet of paper. In this situation, the selection process will start with a decision on what "engine size" – 4-bit, 8-bit, 16-bit or 32-bit – is required by the application. A 4-bit component, such as those in OKI's OLMS63K and 64K families and Atmel's Marc4 family, will offer the lowest-cost solution, particularly suitable for price-sensitive battery-powered consumer applications. The downside is that it can be harder to develop code for 4-bit architectures, and working with 4-bit instructions and data widths can limit arithmetic capabilities.

An 8-bit microcontroller can comfortably run most embedded applications; many are based on familiar, established architectures such as the 80C51 range. Because 8-bit technology has been around a long time, there is also a huge range of controllers available – from low-cost, low-speed 4-bit micro replacements, to devices which deliver tens of MIPS, such as Atmel's AVR series which achieves a 50ns instruction cycle time when clocked from a 20MHz crystal (Fig 1).

For applications that require more processing power, the choice is more likely to be between 16- and 32-bit devices. The latter type of component – often based on an ARM or MIPS core – is coming to dominate this sector of the market simply because vendors are able to offer 32-bit performance at competitive price points. Effectively, the extra performance of a 32-bit device comes at virtually no cost to the user (Fig 2).

The long-standing trend towards de-facto standard cores (for instance 80C51, ARM, MIPS) has led to vendors searching for factors other than pure performance on which to differentiate their products. One of the ways they have found to do this is by providing application-specific support, via a microcontroller specifically configured with the hardware (and sometimes software) required for a particular target market or application. Atmel, for instance, includes in its range devices designed for use in a variety of applications, such as smart card readers, mp3 decoders, battery charging and keyless entry systems.



Fig.1: Atmel's AVR series: a 50ns instruction cycle time from a 20MHz clock



Fig.2: ARM7TDMI® 32-bit RISC processor, are low-cost microcontrollers with Flash memory capacities ranging from 32K to 256K bytes

In fact, whether devices are marketed as application-specific or not, the mix of peripherals available on chip will always have a heavy influence on component selection. Most controllers today offer a wide range of I/O ports, timers and serial interfaces. But it is also possible to find a variety of other on-chip peripherals, including 10/100 Base-T Ethernet MACs, CAN bus interfaces, USB, RF transmitters and graphics drivers. Having these blocks integrated on-chip can not only cut system cost by eliminating external components, they can also cut development time and effort by providing a ready-integrated solution.

In addition, some devices offer blocks of customisable logic which can be used to reconfigurably implement critical functions in hardware. Atmel's FPSLIC, for example, offers an 8-bit AVR MCU together with a field-programmable gate array area of up to 40,000 gates (Fig 3).

Related to the number of I/Os and on-chip peripherals is the packaging and pin-out configuration. It is, of course, important to choose a device that has sufficient input and output pins for the application: and since microcontrollers are available in many kinds of package, from small SO8 to multi-pin BGA, there is a wide range of choice. External buses increase the pin-count, because they require extra interface pins: all-told, this is a trade-off between the need to get data efficiently in and out of the device; and component size and cost. A further consideration in terms of packaging is to ensure that the chosen range includes devices that comply with the new lead-free and ROHS directives.

Memory architecture is always a critical factor in the design of microcontroller-based systems. Whether the memory is on- or off-chip, and how much is required, may well be a key factor in defining both the overall cost of building the system, and its speed of performance. The three main types of memory which are important in the selection process are program memory (the choice will be between Flash, OTP (one-time programmable), ROM and ROMless components), data memory (either on-chip SRAM or external SDRAM) and non-volatile data storage (EEPROM or Flash).

An increasing number of devices are including all three of these memory types on-chip. Atmel's AVR family, for instance, is available with Flash program storage of 1Kbyte to 128Kbyte, on-chip SRAM for data and a few bytes of EEPROM for the storage of data such as configuration information and serial numbers. These features make the AVR family very popular for general-purpose applications in industrial and security markets.

In mainstream applications, Flash has overtaken OTP. Flash gives the design team the flexibility to make code changes late in the design cycle, and provides the ability to use in-system programming (ISP). In this manufacturing approach, a "blank" microcontroller is soldered to the PCB during production, and programmed in-system at a later date. A similar approach also allows successive reprogramming, easing field upgrades. The use of segmented Flash blocks allows a microcontroller to reprogram one segment under control of another segment, without removing the power. Both the Atmel 89C51 and MegaAVR families support this function (Fig 4).

Although most users with very high-volume applications will usually still choose ROM-based devices, due to their lower cost and code security, Flash is also making inroads in this arena via "factory programming" services. The principle is that the microcontroller ships pre-programmed devices direct to the customer's manufacturing facility, eliminating the long lead-times and expensive code changes associated with ROM-based parts.

ROM-base devices also maintain their traditional edge in battery-operated applications, because they can tolerate very low operating voltages (around 0.9V). But here, too, Flash-based microcontrollers are catching up and can now achieve operating voltages (in the case of the Atmel AVR devices) down to 1.8V.

In terms of data memory, the main factor to consider (apart from the intrinsic data requirements of the application) is the development route of the code. C Compilers can be very demanding of RAM space, both in terms of storing variables and pointers, and in terms of stack space.



Fig.3: Atmel's FPSLIC: an 8-bit AVR MCU with a field-programmable gate array up to 40,000 gates



Fig.4.: Both the Atmel 89C51 and MegaAVR families support segmented Flash blocks

The final consideration when choosing a memory configuration is the availability of different variants within the same family. For instance, it is best to choose a device from a family that can offer more memory than the design team expects to need in case the software turns out to need more memory space than anticipated. Conversely, as the project progresses it may be possible to drop to a smaller memory configuration if the code space is smaller (or if a “stripped-down” version of the application is required in future).

As portable equipment becomes more common, designers are also becoming more aware of the power consumption figures for various devices. Assessing power consumption for a microcontroller is rarely a simple business and requires the engineer to look first at the power required at the clock speed necessary to run the application. However, most micros now have a variety of idle and sleep modes and careful consideration needs to be given to how to maximise the use of these modes to reduce power consumption.

The final factor in the choice of a microcontroller is perhaps the most important: cost. Of course, like all of the other factors, assessing the cost of each solution is rarely as simple as one might hope. Component cost may need to be traded against engineer time if one solution is significantly easier to implement than another. On-chip features will trade with inventory and assembly costs of using extra external components. Somewhat more difficult to quantify are the cost implications of using a technology like Flash memory: how do you put a price on the ability to make a late code change, or to offer field upgrade facilities?

The cost equation will also change depending on the quantities of components to be ordered and assemblies to be manufactured. It therefore makes sense to analyse as closely as possible all of the elements of system cost, and if possible produce some “what if?” scenarios to assess the effects of various changes.