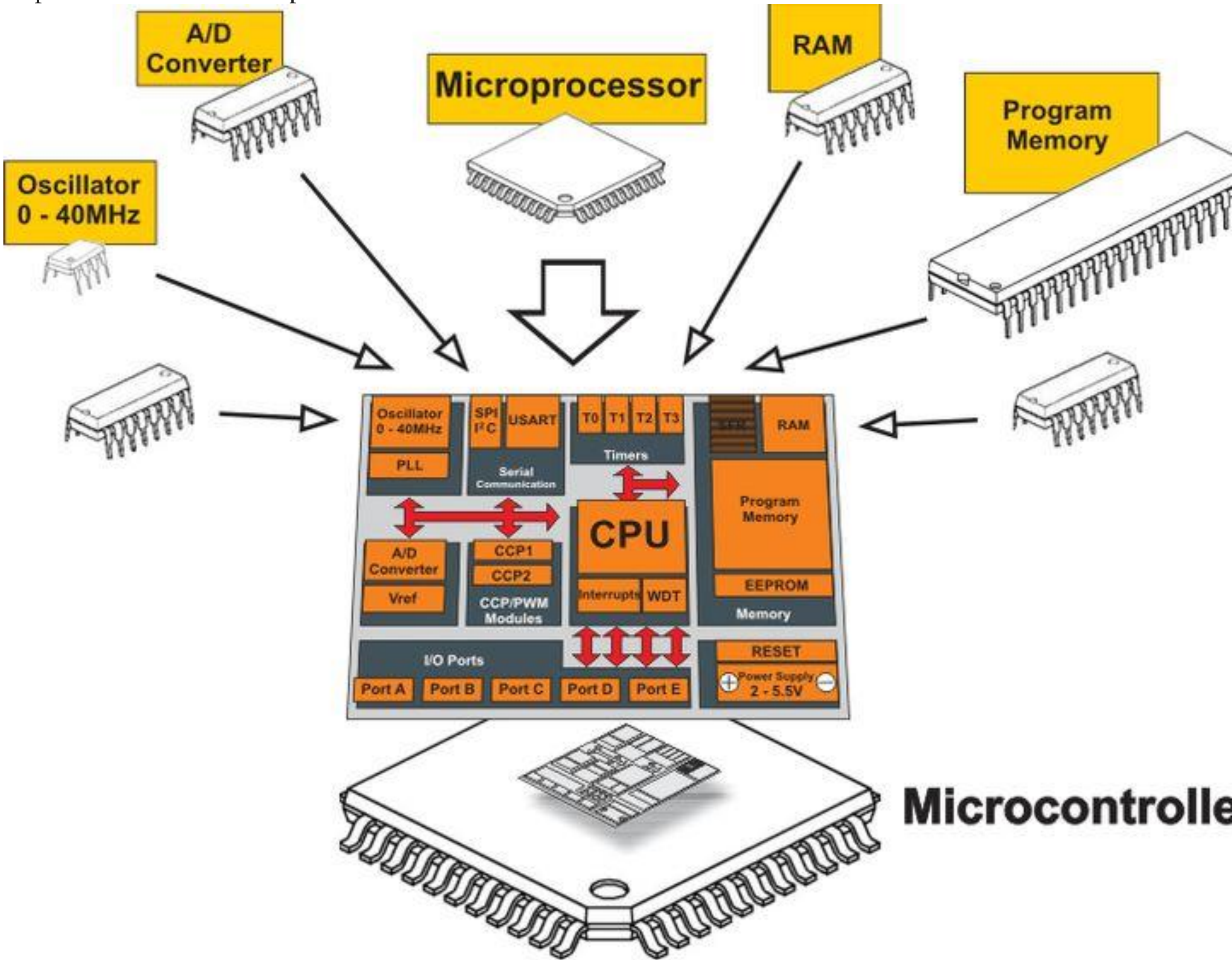
Selecting the right device on which to base your new design can be daunting. The need to make the right balance of price, performance and power consumption has many implications. First, there will be the immediate technology considerations for the design you are able to embark on. However, if microcontroller (MCU) or microprocessor (MPU), becomes the basis of a platform approach, the decision can have long-lasting consequences. Difference between microprocessor and microcontroller becomes an important debate at this point.



Microcontroller vs Microprocessor: Primary Differences

Typically an MCU uses on-chip embedded Flash memory in which to store and execute its program. Storing the program this way means the MCU having a shorter start-up period and executing code quickly. The only practical limitation to using embedded memory is that the total available memory space is finite. Most Flash MCU devices available on the market have a maximum of 2 Mbytes of Program memory. This may prove to be a limiting factor, depending on the application.

MPUs do not have memory constraints in the same way. They use external memory to provide program and data storage. The program is typically stored in non-volatile memory, such as NAND or serial Flash. At start-up, this is loaded into an external DRAM and execution commences. This means the MPU will not be up and running as quickly as an MCU but the amount of DRAM and NVM you can connect to the processor is in the range of hundreds of Mbytes and even Gbytes for NAND.

Another difference is power. By embedding its own power supply, an MCU needs just one single voltage power rail. By comparison, an MPU requires several difference voltage rails for core, DDR etc. The developer needs to cater for this with additional power ICs / converters on- board.

Difference between Microprocessor and Microcontroller: Application Perspective

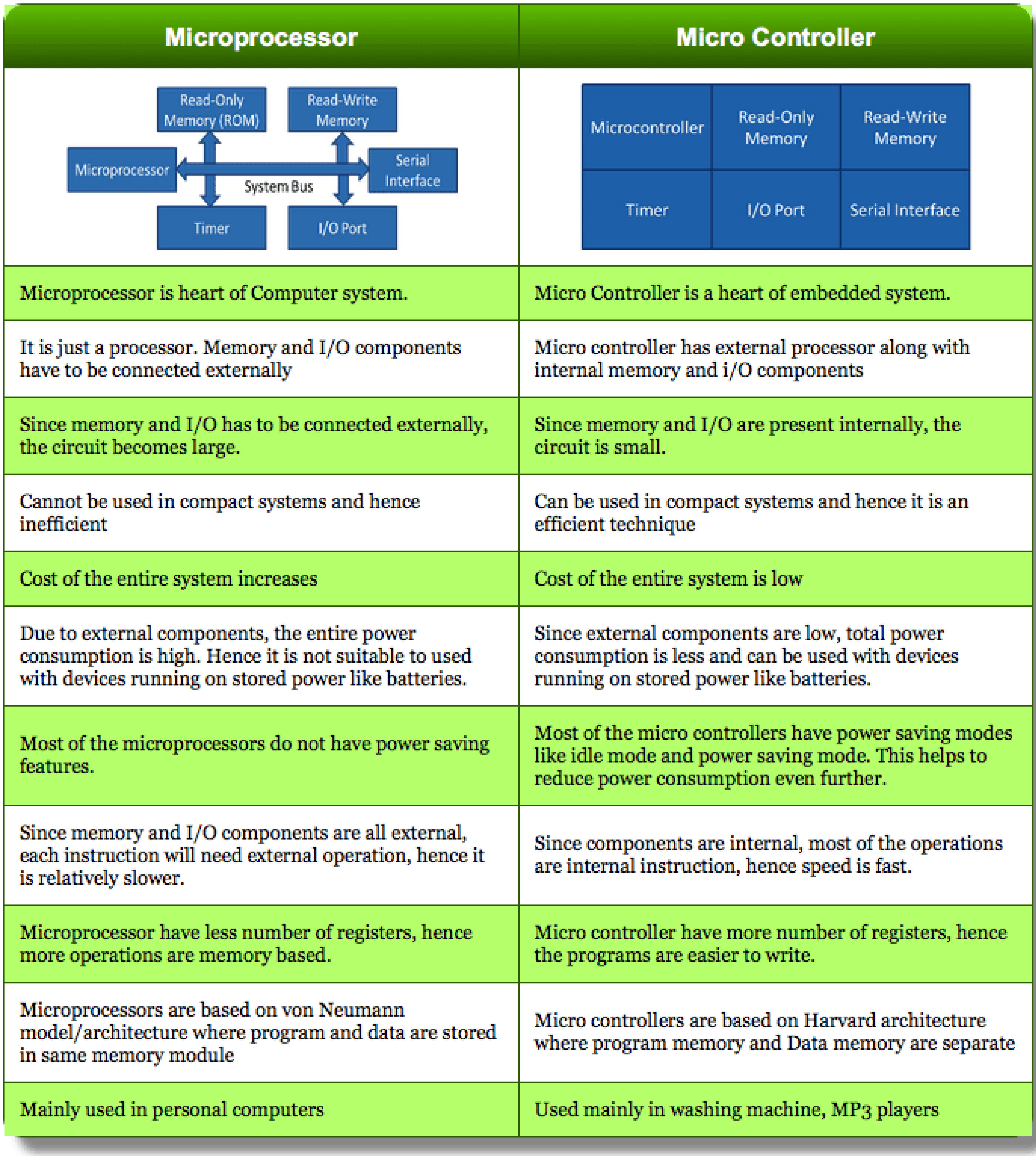
From the application perspective, some aspects of the design specification might drive device selection in particular ways. For example, is the number of peripheral interface channels required more than can be catered for by an MCU? Or, does the marketing specification stipulate a user interface capability that will not be possible with an MCU because it does not contain enough memory on-chip or has the required performance?

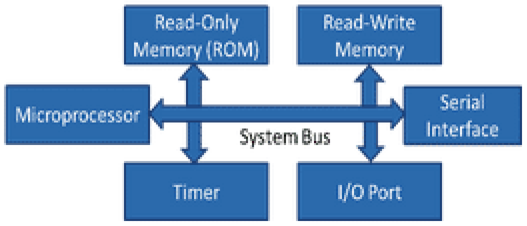
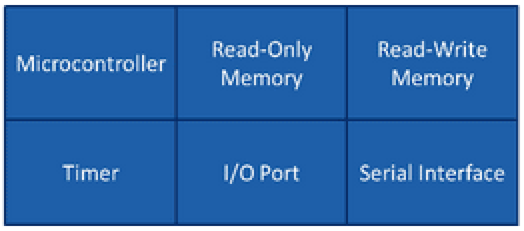
When embarking on the first design and knowing that, it is highly likely there will be many product variations. In that case, it is very possible a platform-based design approach will be preferred. This would stipulate more “headroom” in terms of processing power and interface capabilities in order to accommodate future feature upgrades.

Some measurement parameters

An attribute that is difficult to determine is the required processing performance any given design might require. Processing power, measured in terms of Dhrystone MIPS (DMIPS), helps quantify these criteria.

Explained below is table for the difference between microprocessor and microcontroller.



Microprocessor	Micro Controller
 Read-Only Memory (ROM) Read-Write Memory Microprocessor Serial Interface System Bus Timer I/O Port	 Microcontroller Read-Only Memory Read-Write Memory Timer I/O Port Serial Interface
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

Difference between Microprocessor and Microcontroller

For example, an ARM Cortex-M4-based microcontroller such as Atmel’s SAM4 MCU is rated at 150 DMIPS. Whereas an ARM Cortex-A5 application processor (MPU) such as

Atmel’s SAMA5D3 can deliver up to 850 DMIPS. One way of estimating the DMIPS required is by looking at the performance hungry parts of the application.

Running a full operating system (OS), such as Linux, Android or Windows CE, for your application would demand at least 300–400 DMIPS. For many applications, a straightforward RTOS might suffice and an allowance of 50 DMIPS would be more than adequate. Using an RTOS also has the benefit that it requires little memory space; a kernel of just a few kB being typical. Unfortunately, a full OS demands a memory management unit (MMU) in order to run; this in turn specifies the type of processor core to be used and require more processor capability.

Difference Between Microprocessor and Microcontroller: Applications

For running applications that are more number-crunching intensive enough, DMIPS allowance needs to be reserved on top of any OS and other communication and control tasks. The more numeric-based the application, the more likely an MPU is required.

The user interface (UI) can be a serious consideration irrespective of the aim of the application. As consumers, we have become familiar and comfortable with using colourful and intuitive graphical UIs. Industrial applications are increasingly using this method of operator interaction. The operating environment, however, can limit the usage on this one. For the UI there are a number of factors.

Why are the differences necessary?

Firstly, is the processing overhead required? An overhead of 80–100 DMIPS might suffice for a UI library such as Qt, since it is widely used on top of Linux. The second factor is to do with the complexity of the UI. Higher processing power and memory is needed for more animations, effects, multimedia content and more changes applied to the image to be displayed. And these requirements scale up with the resolution, that is why for applications designed to be UI centric an MPU is more likely to suit.

On the other hand, a simpler UI with pseudo-static images on a lower resolution screen can be addressed by an MCU. Another argument in favour of the MPU is that the generally come equipped with an embedded TFT LCD controller. Very few MCUs have this capability. The TFT LCD controller and some other external driver components have to be added externally. So, while possible to achieve with an MCU, the developer needs to look at the overall BOM.

Sampling a microcontroller

Some Flash MCUs are now coming onto the market with TFT LCD controllers embedded. There must however still be enough embedded SRAM memory available to drive the display. For example, the QVGA 320 x 240 16-colour format requires 150 kB of SRAM to feed and refresh the display.

This is a fairly high amount of SRAM to dedicate. Some extra memory might be required, which would further add to the BOM and bridge the gap with the MPU solution. More complex and advanced graphical UIs, especially using screens larger than 4.3” inches, would stipulate an MPU. If MPUs are seen to dominate when it comes to run a UI on a colour TFT screen then MCUs are the kings for segment or dot matrix LCD control and other screens with serial interfaces.

Difference Between Microprocessor and Microcontroller: Connectivity Standpoint

From the connectivity standpoint, most MCU and MPU devices are available, with all the common popular peripheral interfaces. High-speed communication peripherals such as HS USB 2.0, multiple 10/100 Ethernet ports or Gigabit Ethernet port are generally only found on MPU. They are better capable to handle and process large amounts of data. Whether there are enough suitable channels and bandwidth to handle the data traffic is a key question.

Depending on the communication protocols used, the impact on code space using third-party stacks should be checked. Applications demanding high-speed connectivity especially in combination with using OS-based stacks will require an MPU-based design.

Another key aspect driving the difference between microprocessor and microcontroller selection is the need for a real-time/deterministic behaviour of the application. Because of the processor core used in an MCU, as well as the embedded flash and considering the software used that is either an RTOS or bare metal C, the MCU will definitely take the lead on this aspect and will address perfectly the most time critical and deterministic applications.

Difference between Microprocessor and Microcontroller: Power Consumption

A final point to consider is power consumption. While MPUs do have low power modes there are not as many or as low as the ones you would find on a typical MCU. With the external hardware supporting an MPU has an added factor, putting an MPU into a low power mode might also be slightly more complex.

Also, the actual consumption of an MCU is magnitudes lower than an MPU. In low power mode for example, with SRAM and register retention, you can consider a factor 10 to 100. This is directly related to the amount of RAM and power required by an operating system to resume operation instantaneously. The decisions involved in selecting either an MCU or MPU-based approach are many and involve performance, capability and the BOM budget.

Selecting one?

Broadly speaking, MCUs tend to be used in cost optimised solutions which require tight control of BOM and power saving. Functionally rich and high performance applications employ on a scale, larger number of MPUs. Ultra low power applications such as remote controls, consumer electronics and smart meters where the design emphasis puts longevity of battery life and none or little UI interaction find larger use of MCUs.

They are also used where a highly deterministic behaviour is needed. MPUs are ideal for OS-based industrial and consumer applications. These might be computed intensive and require multiple high-speed connectivities or a rich UI.

Selecting a vendor offering highly compatible MCU and MPU products where you can easily migrate up and down and maximize software reuse provides the best return on investment over time.