Selecting the right microcontroller for a product can be a daunting task. Not only are there a number of technical features to consider, there are also business case issues such as cost and lead-times that can cripple a project. At the start of a project there is a great temptation to jump in and start selecting a microcontroller before the details of the system has been hashed out. This is of course a bad idea. Before any thought is given to the microcontroller, the hardware and software engineers should work out the high levels of the system, block diagram and flowchart them and only then is there enough information to start making a rational decision on microcontroller selection. When that point is reached, there are 10 easy steps that can be followed to ensure that the right choice is made.

**Step 1: Make a list of required hardware interfaces**

Using the general hardware block diagram, make a list of all the external interfaces that the microcontroller will need to support. There are two general types of interfaces that need to be listed. The first are communication interfaces. These are peripherals such as USB, I2C, SPI, UART, and so on. Make a special note if the application requires USB or some form of Ethernet. These interfaces greatly affect how much program space the microcontroller will need to support. The second type of interface is digital inputs and outputs, analog to digital inputs, PWM’s, etc. These two interface types will dictate the number of pins that will be required by the microcontroller. Figure 1 shows a generic example of a block diagram with the i/o requirements listed.
Step 2: Examine the software architecture
The software architecture and requirements can greatly affect the selection of a microcontroller. How heavy or how light the processing requirements will determine whether you go with an 80 MHz DSP or an 8 MHz 8051. Just like with the hardware, make notes of any requirements that will be important. For example, do any of the algorithms require floating point mathematics? Are there any high frequency control loops or sensors? Estimate how long and how often each task will need to run. Get an order of magnitude feel for how much processing power will be needed. The amount of computing power required will be one of the biggest requirements for the architecture and frequency of the microcontroller.

Step 3: Select the architecture
Using the information from steps 1 and 2 an engineer should be able to start getting an idea of the architecture that will be needed. Can the application get by with eight bit architectures? How about 16 bits? Does it require a 32 bit ARM core? Between the application and the required software algorithms these questions will start to converge on a solution. Don’t forget to keep in mind possible future requirements and feature creep. Just because you could currently get by with an 8 bit microcontroller doesn’t mean you shouldn’t consider a 16 bit microcontroller for future features or even for ease of use. Don’t forget that microcontroller selection can be an iterative process. You may select a 16-bit part in this step but then in a later step find that a 32 bit ARM part works better. This step is simply to start getting an engineer to look in the right direction.

Learn how to develop motor control applications with NXP MagniV MCUs using model-
Memory...

Step 4: Identify Memory Needs
Flash and RAM are two very critical components of any microcontrollers. Making sure that you don’t run out of program space or variable space is undoubtedly of highest priority. It is far easier to select a part with too much of these features than not enough. Getting to the end of a design and discovering that you need 110% or that features need to be cut just isn’t going to fly. After all, you can always start with more and then later move to a more constrained part within the same chip family. Using the software architecture and the communication peripherals included in the application, an engineer can estimate how much flash and RAM will be required for the application. Don’t forget to leave room for feature creep and the next versions! It will save many headaches in the future.

Step 5: Start searching for microcontrollers
Now that there is a better idea of what the required features of the microcontroller will be the search can begin! One place that can be a good place to start is with a microcontroller supplier such as Arrow, Avnet, Future Electronics or similar. Talk with an FAE about your application and requirements and often times they can direct you to a new part that is cutting edge and meets the requirements. Just keep in mind that they might have pressure on them at that time to push a certain family of microcontrollers!

The next best place to start is with a silicon provider that you are already familiar with. For example, if you have used Microchip parts in the past and had a good experience with them, then start at their website. Most silicon providers have a search engine that allows you to enter your peripheral sets, I/O and power requirements and it will narrow down the list of parts that match the criteria. From that list the engineer can then move forward towards selecting a microcontroller.

Step 6: Examine Costs and Power Constraints
At this point the selection process has revealed a number of potential candidates. This is a great time to examine the power requirements and cost of the part. If the device will be powered from a battery and mobile, then making sure the parts are low-power is absolutely precarious. If it doesn’t meet power requirements then keep weeding the list down until you have a select few. Don’t forget to examine the piece price of the processor either. While prices have steadily been approaching $1 in volume for many parts, if it is highly specialized or a high-end processing machine then price might be critical. Don’t forget about this key element.

[Click below for more steps]
Kits, Tools, Play

Step 7: Check part availability
With the list of potential parts in hand, now is a good time to start checking on how available the part is. Some of the things to keep in mind are what the lead times for the part? Are they kept in stock at multiple distributors or is there 6 – 12 week lead time? What are your requirements for availability? You don’t want to get stuck with a large order and have to wait three months to be able
to fill it. Then there is a question of how new the part is and whether it will be around for the duration of your product life cycle. If your product will be around for 10 years then you need to find a part that the manufacturer guarantees will still be built in 10 years.

**Step 8: Select a development kit**
One of the best parts of selecting a new microcontroller is finding a development kit to play with and learn the inner working of the controller. Once an engineer has settled their heart on the part they want to use they should research what development kits are available. If a development kit isn’t available then the selected part is most likely not a good choice and they should go back a few steps and find a better part. Most development kits today cost under $100. Paying any more than that (unless it is designed to work with multiple processor modules) is just too much. Another part may be a better choice.

**Step 9: Investigate compilers and tools**
The selection of the development kit nearly solidifies the choice of microcontroller. The last consideration is to examine the compiler and tools that are available. Most microcontrollers have a number of choices for compilers, example code and debugging tools. It is important to make sure that all the necessary tools are available for the part. Without the right tools the development process could become tedious and expensive.

**Step 10: Start Experimenting**
Even with the selection a microcontroller nothing is set in stone. Usually the development kit arrives long before the first prototyped hardware. Take advantage by building up test circuits and interfacing them to the microcontroller. Choose high risk parts and get them working on the development kit. It may be that you discover the part you thought would work great has some unforeseen issue that would force a different microcontroller to be selected. In any event, early experimentation will ensure that you made the right choice and that if a change is necessary, the impact will be minimal!

Also see:
- Selecting 8-bit MCUs: A practical guide
- 5 firmware criteria in MCU selection
- 8-bit isn't dying, it's growing
- Bringing up PIC: The agony
- Using MicroPython for real-time software development
- *duino meets chipmakers: What's all this then?
- Brother, can you lend me an ARM?

Jacob Beningo is a lecturer and consultant on embedded system design. He works with companies to develop quality and robust products and overcome their embedded design challenges. Feel free to contact him at jacob@beningo.com or at his website www.beningo.com.

If you liked this and would like to see a weekly collection of related products and features delivered directly to your inbox, click here to sign up for the EDN on Systems Design newsletter.