The Changing Nature of Development Cycles for Portable Devices

Anders Holmberg, Product Manager, IAR Systems

Not so long ago I worked for a mobile handset company as a software developer. A team of 40 worked across several different sites and countries on a major 2 year project to design a new handset. It was not uncommon for us to work on a subset of code and never see the whole application. Responsibility for specifications, such as the power budget was shared between the hardware and software teams. We software developers rarely had immediate access to tools to see how their application was performing. Naturally, the impact of combined and un-identified power “hot spots” only appeared to get bigger as the development cycle progressed. As a consequence attention to battery lifetime only became an engineering priority several years into the project by which stage a lot of code had been written and tested. Looking back, if we had been able to analyse power consumption as our code ran, using easy to use low cost tools, we would have been able to avoid the headaches later. This relatively new approach to embedded software development, called power debugging, makes our job a lot easier.

The technology for power debugging is based on the ability to sample the power consumption and correlate each sample with the program's instruction sequence and hence with the source code. This correlation allows you to get instant insight to how the code you write affects power consumption and what works and what does not. There is no need to run to the test lab just to see how a small change in the source changes the power consumption. You can do that at your desk, instantly.

There are several different formats in which you can have the information displayed alongside your source. Figure 1 shows an example of the Timeline window in IAR Embedded Workbench. By visualizing power consumption alongside the interrupt activity on a single timeline, you can get an instant understanding of what events cause power consumption to change.

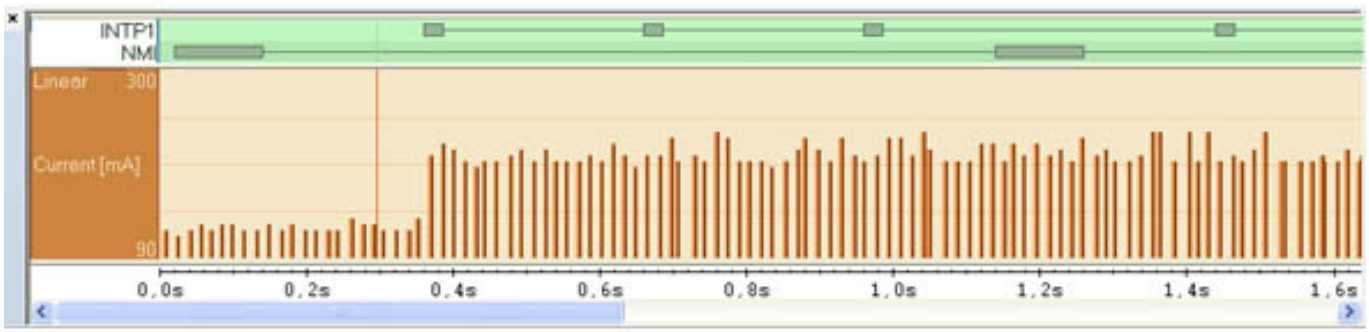

Figure 1. An example of the Timeline window in IAR Embedded Workbench.

Another useful approach is to make a power profile of your application, much like an execution time profiler but measuring instead how much power is consumed in each

function. Once you can see what the application's power profile looks like you can start optimizing the code. Perhaps using a poll loop to wait for a peripheral device status change would be better implemented with a hardware timer. Could you put the CPU into sleep mode during a DMA transfer? While waiting to receive data on a serial port what mode is the CPU in? If it is still running at full speed when idle you should be using one of the many sleep or low power modes that might be available on your MCU.

Being armed with the power consumption information you can tweak your code during the development cycle thereby making battery life an integral part of your design. I didn't have the opportunity to use power debugging back when I was writing code, but if it had been available for us then the customers we had would not have had to recharge their handsets quite so often.