The art of noise: how to design to minimise interference

In a perfect world, signal margins and power supply voltages are maintained for a safe, stable environment, says **Graeme Clark, Principal Engineer, Renesas Electronics**

In an ideal world, signal voltage margins and signal timing margins are always positive, power supply voltages are always within the operating voltage range, and our environment is completely benign. Unfortunately, we live in the real world.

The real world is dirty and noisy with power distribution systems that are not perfect. The supply voltage can drop below the operating voltage, resulting in system malfunction or failure. Switching transients create noise and reduce signal margins, and impedance discontinuities can distort signals, reducing signal operating margins.

There is also radiated or conducted noise from internal and external sources and electrostatic discharge and lightning surges which can disrupt or even destroy systems. In addition, thermal stress, mechanical stress and component ageing can cause systems to fail.

This article will look at some of these issues and the measures that can be applied to designs to remove or at least minimise some of them. In particular, some features implemented in microcontroller design can mitigate some of these issues.

In the past, around 25 years ago, microcontrollers were designed and implemented on 1.0µm or 0.8µm CMOS technology. Today's devices use significantly more advanced process technology, with line widths orders of magnitude smaller. For example, the latest Renesas RA family microcontrollers are implemented on the latest 40nm and 22nm process geometries.

As the transistor size in the latest devices becomes smaller and the transistor switching frequency becomes faster, noise becomes an increasing concern.

Figure 1 shows a simplified comparison between a 1.0µm device operating at 8MHz and a modern device operating at 100MHz on a 40nm process. It is clear that the switching time of the 40nm device is much faster and the signals can be faster than the typical noise signal. Noise becomes a larger concern as design moves towards smaller process geometries. Design practices include integrating features to help operate in such an environment, with carefully designed power supply circuits, optimised I/O buffers and specialist protection circuitry. It is still important to minimise any effects in designs as much as possible; once noise enters the device, it is much harder to remove.

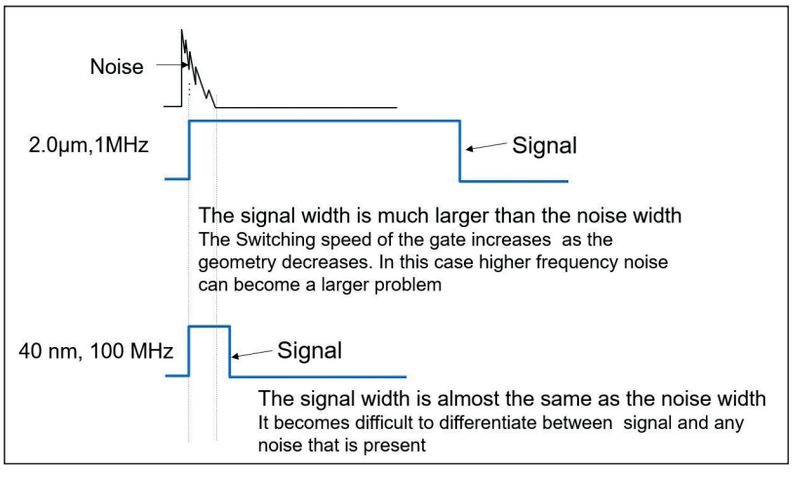

Figure 1: Simplified comparison between a 1.0µm device and one implemented in a 40nm process.

Noise sources

External noise sources are often one of the biggest threats to reliable operation. These can include switching noise from a power supply, noise caused by sparks from industrial machinery or motors, for example. Other sources include induction noise from relays, transformers, buzzers or fluorescent lamps, as well as static discharges, typically, but not exclusively, from the body of users, and of course lightning.

Internal noise can come from a wide variety of sources. Current loops on a PCB can be a significant source of radiated noise. If current flows in a closed loop formed by the microcontroller and its I/O signals, as shown in Figure 2, this current loop can act like an antenna and significant noise can be radiated, especially if the current is large.

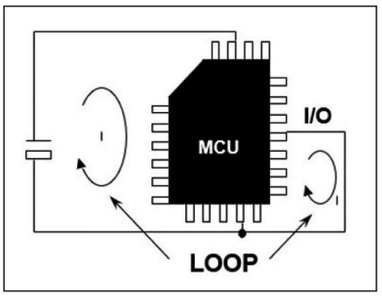

Figure 2: An example of PCB current loops.

With a badly designed ground plane, where there is a voltage difference between the ground at different points on the PCB, current can flow between these points, which can act like an antenna. Badly designed oscillator circuits can also radiate noise. This is why it is advisable to follow the recommended circuit parameters and ground plan. This is especially important as a badly designed oscillator can inject noise into many parts of the circuit. A badly designed oscillator circuit can also fail.

Another key area is the I/O system, especially with multiple high-speed devices on an external bus. A badly designed I/O system, where over- or under-shoot occurs, can cause devices to exceed the electrical specification. This can damage devices over time, causing failure, as well as increased power consumption and noise radiation.

Noise on pins

Noise can affect any pin on a microcontroller. However, microcontrollers'

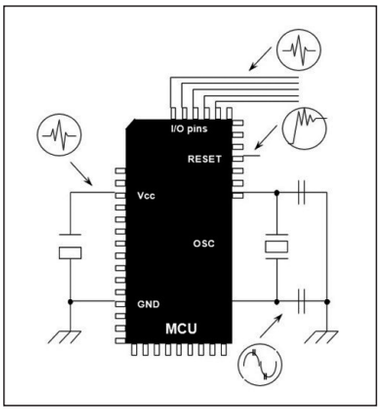

Figure 3: Microcontroller system pins.

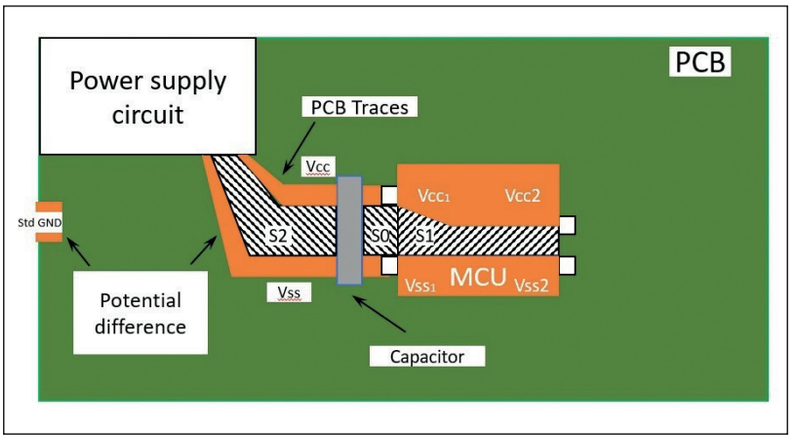

Figure 4: The relationship between microcontroller and PCB.

system pins are particularly sensitive to noise, as they typically control the fundamental operation of the device, and a failure induced by noise here could cause the device to malfunction. Special care should be taken to make sure the possibility of noise interfering with the normal operation of the system pins is minimised.

System pins on a typical microcontroller can include the reset pin, the power supply pins, the oscillator pins, and the mode or special function pins. To minimise the chance of noise disturbing these pins, special care should be taken to make sure the power supply pins have solid voltage levels with the required filtering and that the ground plan does not have any current loops. Designers should also make sure the oscillator is placed as close to the chip as possible, that the PCB layout follows the recommendations of the supplier and that the reset pin is protected from fast transient signals.

Rules to minimise noise effects:

- Guard memory/clock traces from other signals;
- Consider filtering and/or buffering external connections;
- Always use high-frequency Vcc/Vss bypass capacitors close to every device;
- Always run Vcc/Vss in parallel and as close together as possible to minimise current loops;
- Use parallel signal/return traces on PCB, especially for fast signals or traces carrying substantial current;
- Consider the use of a multi-layer layer board with dedicated, unbroken Vss/Vcc planes;
- Do not use a higher frequency than required – the correct frequency minimises noise and power consumption as well.

Power traces on the PCB

The area between the power supply lines should be kept as small as possible to minimise the potential antenna. Designers should also minimise the current flow to this antenna to keep radiated noise to a minimum.

The system power supply is often one of the largest sources of internal noise. An effective power distribution system can be designed using bypass capacitors and EMI filters. Keeping the antenna pattern area between the power traces on the PCB as small as possible means that the surrounding current loop area (S0, S1, S2) is reduced. The most effective way to do this is to use parallel tracks for Vcc and Vss lines.

A bypass capacitor connected across the power supply lines to every IC will significantly reduce noise. These should be as close as possible to each device. Typically, the most effective capacitor values are 0.01µF to around 0.1µF. In a particularly noisy system, it may be worth combining different values of a capacitor to try to improve the noise performance.

Due to differences in the high-frequency characteristics of various types of bypass capacitors, designers should choose the most suitable, lowest impedance capacitor according to the noise frequency range. For most microcontrollers, ceramic and tantalum type capacitors are usually suitable. An electrolytic capacitor can be used for filtering at the PCB power supply input.

The number of tracks between devices should also be minimised and the length of each should be as short as possible. Tracks between the microcontroller and other devices act like an antenna which causes noise. For example, the serial bus, such as I2C or SPI, could be used to talk to external devices rather than a full parallel bus. This minimises noise as well as power consumption and PCB space. These are typically high-frequency connections, so users should be sure to keep the traces short.

Traces that carry high current in a design need special care. This is why large current traces should not be placed near the oscillator or other system pins, such as the mode or reset pin. These could easily be interfered with by noise.

Special care needs to be taken with any external oscillator circuits, especially if the design uses the low power 32kHz crystal for low power operation. It is advised to follow the oscillator circuit layout in the hardware manual and follow the circuit recommendations of the oscillator supplier as well as to take advantage of an oscillator specification service if offered (especially for a 32kHz oscillator design). Other signal lines should not cross the oscillator traces to avoid crosstalk.

It is important to keep signal and power supply tracks as far as possible from the oscillator and to not feed the ground between the pins of the microcontroller. A stable oscillator circuit with a large operating margin is much less likely to cause problems. Using an internal oscillator instead also solves a lot of these issues.

Layout advice

Other good layout practices for microcontroller systems include:

- The use of wide and short traces for Vcc/Vss wherever possible;
- Reducing the impedance of the power supply circuit to reduce inductive noise problems;
- Use Vss/Vcc planes where possible. (At higher frequencies, typically >4MHz, the return current follows as closely to the signal path as possible. Therefore, users should plan signal returns paths carefully, especially for signals with high currents.)
- The ground plane should not be broken, as this increases signal path impedance;
- Use current limiting resistors on I/O pins.

A typical technique to minimise the issues caused by external noise include separating the CPU bus (with memory)

from the peripheral bus to can minimise disturbance of the CPU operation. Other techniques are the use of a distributed clock system and a module stop function on every peripheral. The RA family, for example, also includes on-chip noise filtering on some of the more sensitive inputs, such as reset or oscillator inputs as well as optimising the I/O buffers and the power supply design.

The use of an advanced process technology enables the integration of various components that otherwise would have to be located externally. This helps to improve reliability, removing the need for external circuitry such as oscillators and power management devices. The integration of on-chip oscillators, POR/LVD (brownout systems) and watchdog timers on-chip can greatly help in reducing the impact of noise.

Addressing the CPU

Designers should also consider how fast they need to run the CPU. The maximum speed may be dictated by the baud rate needed for the UART or by the time in which a Fast Fourier Transform (FFT) filter function must be completed. However much of the time, and for many designs, the device does not have to run at its full speed but only as fast as needed to achieve everything required. The higher the clock speed, the faster the signals and the more problems may arise. The slower it runs, the less current is used and switching in the power supply, and the less noise is generated.

For example, RA microcontrollers have the capability to throttle their speed. Users can change the clock dividers dynamically if required to speed up and speed down. This can add complexity to an application as users must manage the peripheral clocks, but it is worth considering if the clock speed can be reduced.

It is also advisable to check the power supply's dimensions are correct for the design. Rather than reusing the same power supply circuit, ensure it is capable of supplying the maximum amount of current required. If the supply drops occasionally, this can cause noise issues as well as generate other system problems.

Many peripheral input pins also have programmable noise filters. These are commonly found on communication peripherals such as UART, SPI, and I2C interfaces as well as many timer inputs. The correct choice of filter clock selection is important to get the best performance from these filters. Often, this is only achieved through trial and error.

Watchdog timers

Communication interfaces are connected to the outside world and often bring high-speed signals into the system. The use of external protection circuitry is often advisable depending on the nature of the external environment.

Watchdog timers are an excellent method of recovering an application when noise causes a system's crash. However, it's important to consider how to use these correctly.

For example, there are two watchdog timers in the RA microcontroller design. The first is a standard watchdog timer which can be clocked from several of the standard clock sources on the microcontroller. This watchdog is typically used to check the application's code execution and to detect any malfunction or unexpected operation. It is typically switched off during low-power applications. This watchdog can generate a reset or an interrupt.

The second watchdog timer is the Intelligent Watchdog Timer (iWDT). This uses a dedicated low speed, on-chip oscillator as its clock source, allowing a long timeout and independent operation from the normal CPU clocks. Even if the CPU clock fails, the iWDT will still cause a reset or an interrupt after it times out. Each of the watchdogs can operate with a window function, where a tickle that occurs too early, as well as the watchdog timing out, can cause a reset. Each watchdog can be selected to auto-start after a reset by setting the relevant bits in the option function select register, a flash-based register that holds initialisation data for the device.

A common mistake is to place the watchdog tickle inside an interrupt service routine, because only one instance is normally required. This can be dangerous, however, as while the main application may crash, the interrupt system may continue to operate, negating the watchdog.

Upon completion of the application, designers should analyse all the possible execution paths. This can be difficult for complex applications using an OS, but modern code analysis tools integrated into modern development tools can help.

Multiple instances of the watchdog tickle can be placed at relevant locations throughout the application. This will make the application much more reliable and have the added advantage that it will detect if the application takes an unexpected execution path. This is normally an indication of a serious issue such as a system crash or a pointer overflow.

In any system, the best way to avoid noise problems is to consider noise reduction from the beginning of the design. It is important to understand the environment that the system will operate in, what noise sources will be present, and what steps can be taken to mitigate these.

It's always worth spending some time to consider these issues at the start of the design, as it is much easier to design noise mitigations from the start than to add them at the end of the design - or even worse, modify a design when noise problems are discovered in the field.