Precision Analog 32-Bit Microcontroller Peripherals Add Value to Signal-Intensive Applications

Introduction

General-purpose 32-bit microcontroller (MCU) devices are ubiquitous in today's interconnected, sensor-rich embedded world. The proliferation of embedded intelligence and connectivity in virtually all aspects of our lives has led to a universe of increasingly capable 32-bit microcontroller devices with more precise onboard sensors.

Motor control, radio control, audio sample generation and waveform generation using digital-to-analog converters (DACs) are typical applications that require MCUs with higher precision analog functionality. Because MCUs are not natural companion devices for radios, they often require specialized high-precision analog components or peripherals to be "well-behaved" in the presence of radio signals. End products involving radio or motor control technology generally benefit from the use of MCUs with high-precision peripherals and system IP.

For example, a motor controller with a high-precision pulse-width modulation (PWM) generator is able to control a motor more efficiently, thus saving power and increasing the lifetime of the motor. A radio system with a microcontroller containing a finely-controllable phase-locked loop (PLL) requires fewer external resources to reduce signal interference, resulting in a better end user experience and higher value. In short, 32-bit microcontroller devices with best-in-class precision-analog peripherals can impart higher value to a wide range of applications.

PWM Engine with High-Resolution Capability and Safe-State Functionality

A typical microcontroller almost always contains a PWM signal generator. These signal generators are useful when coupled with external resistor-capacitor (RC) networks for generating audio tones or other sinusoidal waveforms. PWM signals are also used to drive motor control circuits. Thus, placement of the PWM edge can be crucial in generating a smoother sinusoidal waveform with finer frequency and phase control and controlling a motor to a finer degree of efficiency.

A typical motor-control capable PWM engine generates both center- and edge-aligned PWM signals. It also supports a differential mode capability with dead time insertion for applications requiring a "break-before-make" capability. The typical PWM signal generator should have a resolution of at least the highest operating frequency of the device.

For example, in the case of SiM3U1xx/SiM3C1xx Precision32™ MCUs from Silicon Labs, the PWM generator runs at a maximum frequency of 50 MHZ, allowing PWM edges to be generated with a maximum resolution of 20 ns. These 32-bit microcontroller devices also implement a mode in which the PWM edge can be placed on both edges of the highest operating frequency clock of the device, resulting in an edge resolution of 10 ns as shown in Figure 1. This degree of resolution is more than sufficient for most non-power-supply-related applications.

Another key capability in a motor control (or other high-speed control) system is the ability to shut off the motor in the face of some catastrophic external or internal event, such as an over-current condition. This

"kill" capability should shut off the PWM engine, place the control signals in a known good state and configure the I/O pads to a known good state to prevent damage to external circuitry.

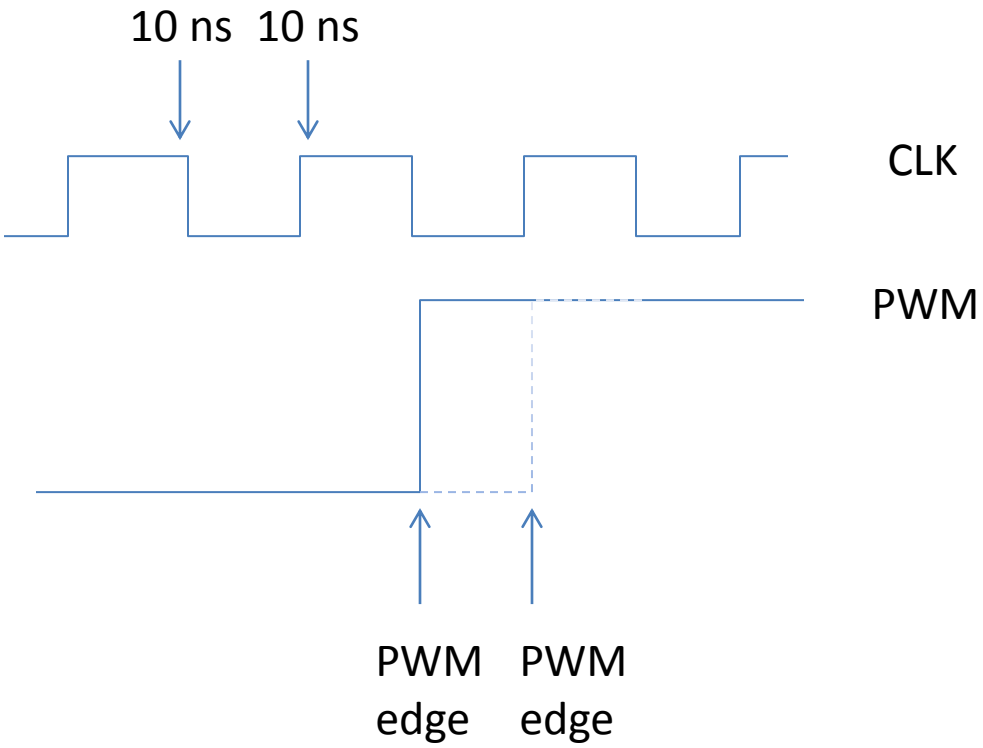

Figure 1. 10 ns Edge Resolution for PWM Generator

The SiM3U1xx/SiM3C1xx MCUs contain six high-drive pads capable of driving up to 150 mA each or 400 mA total. If all the pads are driven simultaneously and left uncontrolled, external circuitry can be damaged. The high-drive pads can be used in conjunction with the PWM engine to directly drive small motors. In the event that a "kill" signal is received, the high-drive pads have a "safe-state" function that causes the pads to revert to one of three preprogrammed states: tri-state, pull high or pull low. These safe-state configuration registers are only reset upon system power-up and otherwise remain undisturbed once written by software and locked. Any other reset does not affect them.

Fine PLL Adjustment Capability

In a typical MCU/radio integrated device, noise mitigation is a key consideration in being able to use the radio effectively. In these integrated applications, it is not uncommon to find the microcontroller shut off altogether when the radio is in receive mode, depending on the use case in transmit mode, to avoid noise pollution from the microcontroller. If the radio is a frequently used device, the impact on MCU performance can be quite severe.

One way to mitigate this effect is to change the MCU's frequency of operation to ensure any noise spurs generated by its clock appear outside of the radio band of interest. This implies having a fine tuning granularity in the microcontroller's PLL such that its frequency can be modified to place these noise spurs outside the radio's band of interest.

The PLL used in Precision32 MCUs provides the capability to move its frequency of operation in approximately 200 kHz steps between 23 and 80 MHz. This fine resolution simplifies integration of the SiM3U1xx/SiM3C1xx MCUs with any radio device without suffering a performance loss due to massively degraded frequency of operation or event cycling.

Direct Memory Access (DMA) to Everything

Direct memory access (DMA) is a mechanism commonly employed to move data between memory and peripherals. This technique relieves the CPU of this trivial task and frees up more MIPS for other useful work.

A typical DMA implementation on a microcontroller has a fixed number of sources and targets on the device, thus restricting the usefulness of the DMA engine. Given the nature of a general-purpose MCU, it is difficult to predict which peripherals need DMA. The general rule is to apply DMA to the high-bandwidth peripherals and ignore all others. However, in a real-time system, it can be beneficial to provide DMA access to all addresses on the machine regardless of bandwidth.

The SiM3U1xx/SiM3C1xx MCUs, for example, implement just such a system. While only a certain number of peripherals have explicit DMA support (i.e., they can be configured as DMA masters with FIFOs and interrupt logic to support detailed bandwidth management), all peripherals can be accessed by the DMA. Those peripherals without explicit DMA support must be bandwidth-managed in software and will not have a mechanism to signal back to the CPU in case of a buffer error condition. For example, the port pulse generator can be controlled by just such a system with some software overhead, thus enabling an arbitrary waveform generator that can be controlled more precisely.

This DMA engine can also be used to control the MCU's PLL using DMA-chaining based on radio traffic. For example, if the radio notifies the microcontroller that it was about to transmit a packet, a software DMA trigger can be used to issue a DMA request that will load the PLL with relevant parameters that shift its frequency, thus reducing its radio interference footprint.

Higher Precision Current DAC

A DAC generates an analog voltage when presented with a digital code. A typical DAC will have a certain bit-precision that it can achieve given the design. Thus, a 10-bit DAC will convert a 10-bit digital code into a voltage (or current) in a given range.

DACs integrated on MCUs can be used for many purposes, such as bias current generators, arbitrary waveform generators or, in the case of a current DAC (also known as an IDAC), as a relatively noise-immune communication medium.

Given the waveform generation capability of a DAC, a typical implementation allows the DAC to have DMA accessibility and function as a DMA master with a concomitant FIFO. This arrangement allows waveform generation using the DMA from a defined waveform in memory, for example.

The SiM3U1xx/SiM3C1xx MCUs take this DAC concept one step further. A FIFO structure implements a looping capability that extends the precision of the IDAC from 10 bits to 12 bits at one-fourth the data rate by interpolating between adjacent 10 bit values without DMA intervention. For example, if the IDAC outputs three identical values of "x" and a fourth value of "x+1", the IDAC output as measured at the package pin will be a 12-bit precision value as shown in Figure 2. This capability can be used to generate a 12-bit precise bias current.

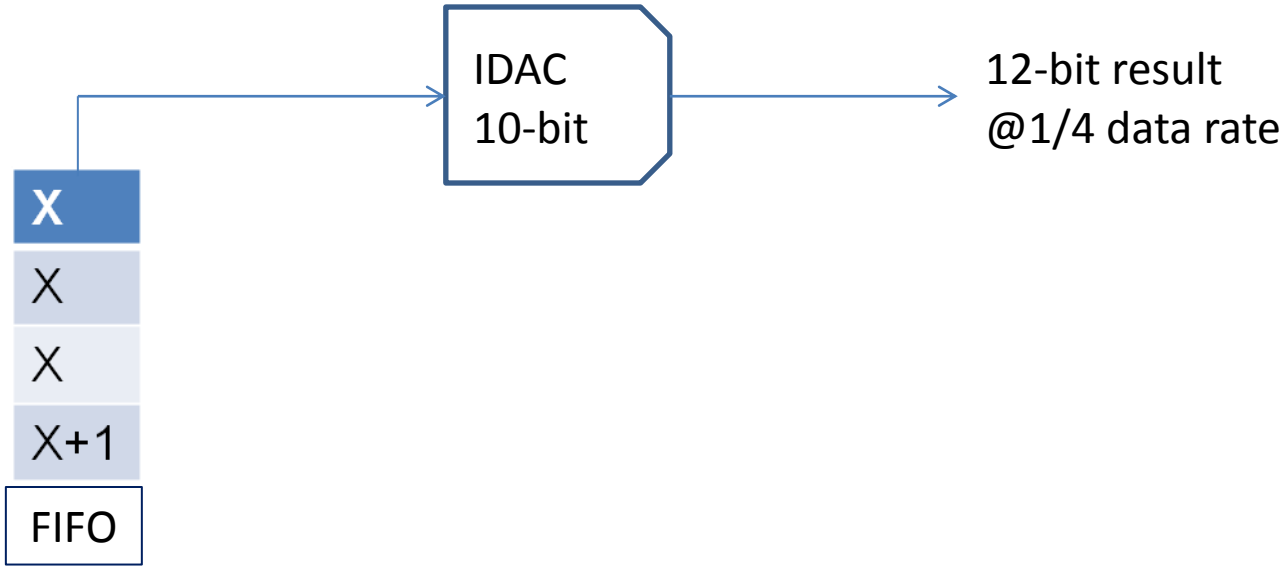

Figure 2.12-Bit IDAC Output

Conclusion

Techniques that extend the precision of standard analog capabilities increase the value of 32-bit mixed-signal MCUs in applications that involve radio and motor control technologies. The high-precision PWM engine, fine PLL adjustment and high-precision IDAC exemplify extended precision capabilities that provide 32-bit microcontroller devices with more capabilities in applications that the these devices could otherwise not serve. DMA-to-everything is a capability that provides the MCU with more granular control and peripheral usage to further enhance value. Higher precision implemented in today's 32-bit microcontroller devices will almost always lead to higher value through better usability and applicability.

# # #