



Designing Modern USB Audio Systems

By Kendall Castor-Perry, Architect – Principal, Cypress Semiconductor Corp.

USB audio is a ubiquitous interface supported by all but the most ancient personal computer hardware and operating systems. With its robust connection and data rate, one might believe that delivering high quality audio over this interface is simple. However, today's successful USB-based audio products are the result of a lot of chip- and system-level attention to solving the thorny problem of clock recovery.

In essence, the problem is that the final output device that delivers audio to the speakers, headphones or line-out socket needs a 'master clock' to pace the audio conversion cleanly. This master clock needs to have two independent attributes: 1) it must be at *exactly* the correct multiple of the underlying audio sample rate (so that you never have to lose or duplicate an audio sample through timing failure); and 2) it must have low enough jitter (or, equivalently, phase noise) that the performance of the digital-to-analogue process isn't compromised. The challenge lies in meeting both these requirements simultaneously.

Part of the difficulty comes from the fact that the receiving end for the digital data streaming over the USB cable *doesn't know the exact sample rate*. In fact, it can only infer the nominal sample rate. What's more, the data that comes in over the USB 'pipe' *isn't accompanied by any form of clock*. This is in significant contrast to most other serial interfaces, which either send a clock, or structure the data so that such a clock can always be extracted from the link when it is running.

The only clocking information available on the USB interface is that every millisecond, a specific type of data packet announces the start of a frame, and this event is detected by the receiving hardware. This millisecond is derived in a known way from the system clock at the transmitting end – and so is the original audio sample rate (well, we'll briefly discuss an exception later).

A simple solution appears to be that one could put this 1 kHz pacing clock into a PLL-based multiplier and multiply it up by whatever factor is required to create the audio master clock and all the sub-clocks that depend on it. However, in a system handling CD-derived audio, the sample frequency is 44.1 kHz and a typical conventional audio DAC requires a master clock of 256x this, or 11.2896MHz. The fact is that multiplying an input reference frequency by such a large factor in a single-stage PLL gives inherently rotten performance. It takes a hit from multiple constraints: loop bandwidth, reference spur rejection, and VCO jitter. What's more, the number we need to multiply the 1 kHz by in this case isn't an integer, making the task that much harder.

Cascading two fairly complex multiplier loops can be made to work from a phase noise and spurious rejection standpoint. This approach, however, tends to be power-hungry, to demand high levels of chip- and system-level analogue design smarts, and to be rather slow to respond to changes in demanded clock frequency. The nominal sample rate being used on a USB audio link can change rapidly between tracks, and waiting for a large fraction of a second for things to stabilize can result in unreliable performance. Such links are primarily found in fixed-frequency studio-based digital audio links, where cost and size aren't important.

Over the years, various ways of creating the necessary audio master clock without suffering from PLL multiplier problems have been integrated into the dedicated chipsets that have gone into innumerable USB speakers, headsets, and external sound cards. These parts do what's needed of them, but don't spend extra silicon area or pin count on "what if" capability. This certainly keeps the cost down, and everybody is happy.

But, what do you do if your next generation USB interfacing needs can't be met by any of these special-function chips? Mobile devices, such as media players and the latest tablets, are built on new platforms and are running new operating systems that are increasingly standardizing on USB as the wired link of choice for a wide range of accessories and enhancements. Some of these systems have combinations of requirements that aren't met by existing USB audio chipsets, causing a 'shock' to the component supply infrastructure. USB Audio is one of the capabilities increasingly being demanded in these small mobile devices.

Extracting audio in digital form from a mobile device has several benefits. The analogue audio interface no longer needs to be the limiting factor in the system's sound quality. This allows the manufacturer of an audio system or accessory connected to the player to aim for higher levels of measured and sonic performance through their own circuit design. Just as important is

the digital audio link's improved resistance to TDMA interference coupled from the cellular modem of the mobile device into the analogue circuits of the audio replay path of the system.

There are plenty of microcontrollers on the market that integrate a USB device port but none have been designed to also have the necessary clock generation and recovery circuitry that is needed to deliver the high standard of audio reproduction that's now demanded. Sometimes this issue can be resolved by using external 'clock cleaning' chips or more sophisticated audio converters that integrate PLLs or sample rate converters, in an attempt to bridge the master clock accuracy/quality gap. However, this moves a system back into the territory of high cost, high current consumption, high component count, or all of these. In addition, the technique of 'under-clocking' audio into a very long memory buffer is unusable in any system where video images (even PowerPoint slides) need to be time-aligned to the audio.

USB clock recovery

The resolution of this issue has recently been much simplified through the availability of highly versatile mixed-signal devices that combine microcontroller, programmable digital logic and versatile configurable analogue circuitry onto a single device. An example of such a device family is Cypress's new PSoC3 (Programmable System on Chip).

When systemic 'shocks' happen, programmable microprocessor-based designs can quickly be adapted, since new code and new circuit boards can be spun so much more rapidly than new silicon. Sometimes, however, an application requires dedicated peripheral or processing support that hasn't yet been incorporated into any of the readily available microprocessors. First solutions to new problems of this nature can end up as 'bitty' combinations of microprocessors with FPGAs, PLDs and various dedicated fixed-function chips that are often only half-utilized to implement some specialized but required behavior. The resulting large circuit boards and inefficient BOMs can threaten to snuff out a nascent new market.

Highly programmable system-on-chip architectures offer an alternative approach. In such devices, chip design effort is initially devoted – often without a clear picture of specific usage scenarios – to creating a more configurable, more flexible architecture, on both the analogue and digital sides. Digital flexibility comes from including blocks (Universal Digital Blocks, or UDBs) that can implement complex combinatorial and sequential logic functions, independently of the main processor core. Dedicated coprocessors for frequently encountered generic signal processing tasks such as filtering can also be included. On the analogue side, the ubiquitous op-amps and comparators can be enhanced with a rich network of switches and on-chip components, to deliver a range of analogue blocks whose interconnection is limited only by users' imaginations. Flexible multi-domain clock trees add a further layer of versatility.

These highly versatile devices can't always match the raw silicon cost of dedicated, single-function devices. However, as soon as something slightly different is needed, the programmable device usually offers the most competitive BOM cost, compared to the patchwork solution required from less flexible parts. And the rapidity of the product design – *and redesign* – process has ensured that programmable "systems-on-chip" have made a significant contribution to electronic product design in the past few years.

Programmable SoCs have already been proven to support all the elements required for a complete contemporary consumer music appliance requiring USB digital audio capability. The programmable digital logic and versatile clock capability deliver a no-external-components approach for generating the required audio master clock and synchronizing it rapidly and exactly to the incoming USB frame structure. The heart of the solution is the USB audio clock recovery process; the basic configuration used is outlined in figure 1:

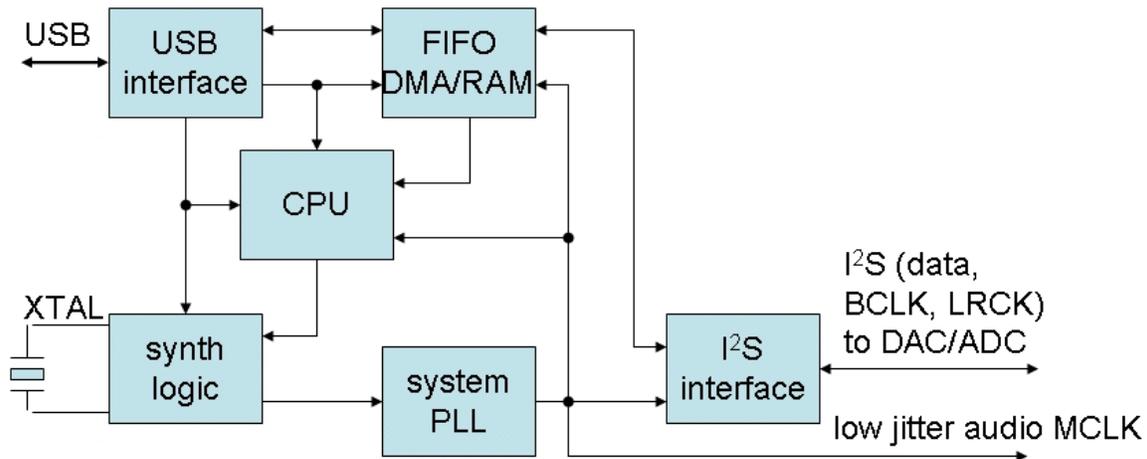


Figure 1: Example USB audio clock recovery architecture

The flexible USB interface of a programmable SoC allows several audio and control protocol endpoint functions to be combined. The matrix of programmable digital logic blocks implements a frequency synthesis system that derives any standard audio sample rate master clocks *exactly* from a single stable crystal source. In normal operation, the clock is locked to the received USB timebase using the start-of-frame token pulse timing. The system clock PLL is integrated into this synthesizer through the flexible clock routing framework. The whole system exactly tracks the source sample rate and delivers a high quality, audio master clock for the system's audio converters, at a jitter level that's commensurate with the needs of modern quality audio systems.

The audio data is typically clocked out of the buffer into one or more standard I²S interfaces with the required number of channels, implemented again with the programmable digital blocks. This interface can connect to a standard audio DAC, processor or 'digital amplifier'. Other custom interfaces can also be implemented in these blocks, for example, S/PDIF transmission. The whole process can operate bidirectionally, with data from an ADC being transmitted back out through the USB port.

Some USB audio modes require that the local clock is able to 'slip' against the incoming clock, for instance from a source that's relaying a remotely-synchronized audio stream. A programmable SoC architecture can run in an adaptive mode that finely trims the local clock to provide the required 'slip'.

Fixed-function microcontrollers just can't meet the tough performance requirements on this clock generation process. Their inflexible clock generation systems can't be adjusted to be both exactly correct and low in jitter, and they usually fall back on a crude "add/lose samples" approach. This might be workable for telephony but is completely unacceptable for high quality audio. Meanwhile, dedicated USB audio interface devices (necessarily another entry on the BOM, in addition to the control microprocessor) can't simultaneously manage the critical bidirectional control protocol traffic that delivers innovative new functionality in these latest media players.

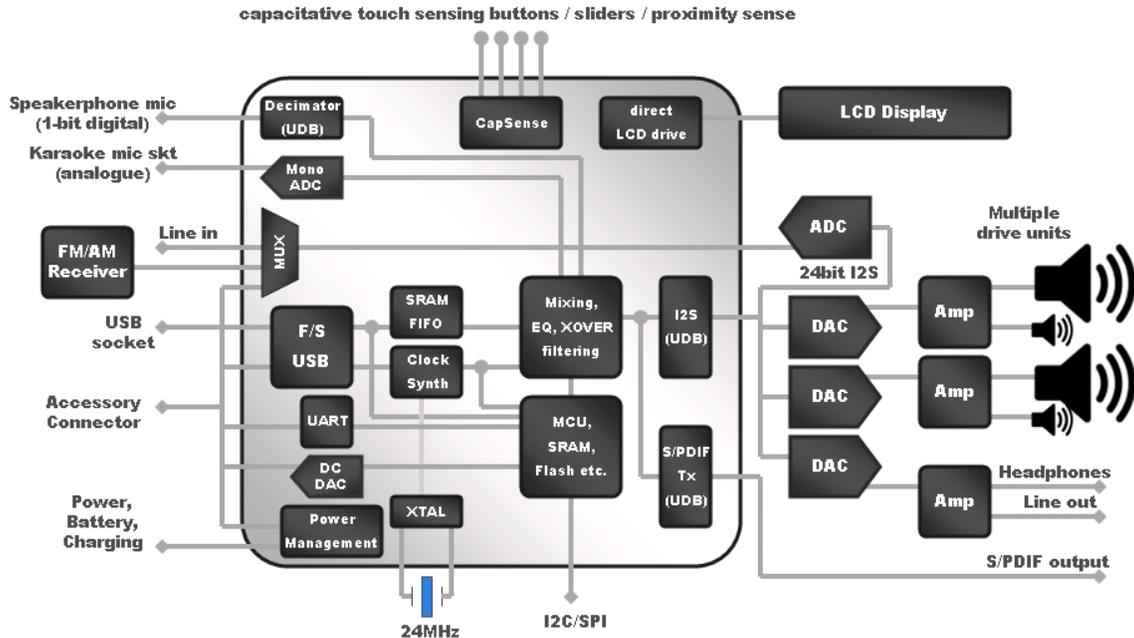


Figure 2: Block diagram of a high-end consumer music appliance using PSoC3

Programmable SoCs can also provide advanced functionality in a cost-effective manner. For example, additional value is provided by some family members of Cypress’s PSoC3 family in the form of an embedded audio filtering engine (the Digital Filter Block). This DFB can post-process the recovered USB audio, for instance, for response equalization and crossover filtering. Sufficient performance is available to render additional digital processing devices redundant; at least ten second order biquad filters can be implemented on each channel of a stereo pair, giving very fine control over frequency response.

The versatility of a highly configurable solution inevitably helps designers to ‘mop up’ most of the glue logic and analogue housekeeping circuitry. Direct LCD drive lowers display cost, and capacitive button sensing ensures both an elegant industrial design and a contemporary human interface. An example of the result is shown in figure 2: a single device that can form the heart of a new generation of mobile device audio accessory or consumer audio appliances that deliver all the benefits of fully digital audio and data exchange. The flexibility of the programmable device means that the required functions and interfaces can be integrated on a mix-and match basis.

Eventually fixed function devices will catch up to programmable SoCs. But by adopting a design architecture based around highly configurable SoC architectures, developers can keep at least one step ahead and be able to act when the next “How do we do *that?*” shock arrives.



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
<http://www.cypress.com>

© Cypress Semiconductor Corporation, 2007. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.