STM32 Journal

Volume 1, Issue 2

In this Issue:

Developing High-Quality Audio for Consumer Electronics Applications

By Paul Beckmann, CEO/CTO, DSP Concepts
Dragos Davidescu, Chief System Architect, STMicroelectronics
John Knab, Application Engineer, STMicroelectronics

With the falling cost of high-performance MCUs, manufacturers are considering adding digital audio functionality to more and more consumer devices and other embedded applications. Their goal is to support the wide variety of media sources users want to access such as an iPhone, Internet radio, external USB devices, and SD cards.

Achieving high-quality sound output, however, is non-trivial. Sound quality depends greatly upon the final system configuration, making it difficult to design even when prototype hardware is available. In addition, implementing real-time digital signal processing algorithms introduces a whole new set of concerns for developers used to MCU-based design. These include implementing advanced filters and processing algorithms, handling fixed-point issues, using DSP-like instructions, and optimizing complex algorithms for speed, MIPS, memory, and power.

In this article, we'll show how developers can leverage MCU-based digital signal processing (DSP) and floating-point unit (FPU) capabilities to enable real-time audio playback, implement enhanced algorithms, convert between multiple clock domains, manage high-speed communications without impacting audio quality, optimize designs to balance quality and cost, and manage other system tasks such as a graphical user interface, all with a single microcontroller.

Consumer Audio

Traditionally, introducing audio to an embedded application requires digital signal-processing capabilities beyond the capabilities of most MCUs. Even a "simple" product like an iPod speaker dock requires a significant number of advanced audio algorithms to achieve full performance:

Spatial enhancement: In an iPod docking station, the speakers may be only 12-18 inches apart. To create a more spacious, rich sound, spatial enhancement is required is to compensate for the close proximity of the speakers.

Multi-channel audio: For systems supporting more than two speakers, the stereo input signal requires processing to create the additional audio channels.

Equalization: Speakers need to be equalized to achieve better sound quality. If the speakers in use change, the equalization needs to be adjusted as well. Developers can employ a variety of equalization methods, including graphic and parametric equalization. For higher-end applications, developers may even want to design their own equalization algorithms using a tool like MATLAB.

Peak limiting: Speakers exhibit nonlinearity at louder sound levels. By applying a time-varying gain and carefully controlling the peak levels, the system can play louder with a minimum amount of distortion.

Boost: When listening to music at low volume levels, much detail, and therefore depth, can be lost. Boosting of the bass and certain other frequencies at low volume levels using loudness compensation or perceptual volume-control techniques can significantly improve perceived sound quality.

Level matching: Level matching eliminates the need for users to adjust the volume for each song

when shuffling through a large library of albums.

Digital audio has commonly been implemented in consumer electronics and embedded applications using a second processor dedicated to this task. To meet market cost pressures, however, manufacturers need to be able to process audio on the host CPU.

In general, it is easier to implement audio on an MCU than it is to implement real-time responsiveness and connectivity on a DSP. DSPs, while excellent at processing audio, don't have the peripherals or interrupt responsiveness required for real-time systems. DSP architectures are also typically designed for high-end signal processing and massive parallelism that exceeds the requirements of the typical consumer application. In addition, DSPs are not designed to support communication interfaces like USB, SD cards, or Wi-Fi, so a DSP-based docking station would still require a second processor to handle connectivity.

With the introduction of DSP capabilities to MCU instruction sets, MCUs now have the advanced math processing capabilities required to handle not only basic audio processing but the advanced algorithms required to improve quality as well. In addition, rather than requiring developers to hand-code assembly as is typical for DSP-based designs, MCUs offer ease-of-use and faster time-to-market through C programming and application libraries. MCUs are also specifically architected to provide short and deterministic interrupt latency as well as ultra low-power operation for battery-powered applications.

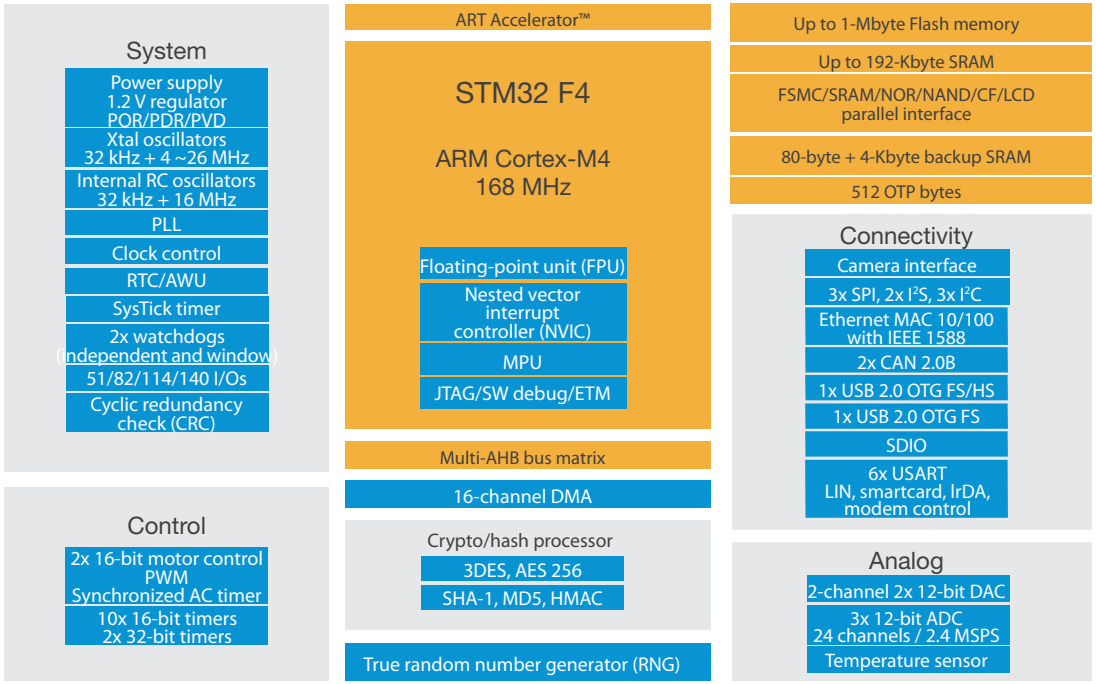

Figure 1 ST has expanded its STM32 MCUs beyond the base Cortex-M architecture with a variety of integrated peripherals to create a wide range of MCUs that optimize performance, memory, and cost for nearly every embedded application.

The STM32 MCU + Audio Architecture

The STM32 architecture from ST has been designed to bring 32-bit MCU capabilities to a wide range of consumer audio applications, including multimedia speakers, docking stations, and headphones. The STM32 F4, based on an ARM Cortex-M4 core operating at up to 168 MHz, also integrates capabilities such as DSP instructions and a floating-point unit to allow manufacturers to produce consumer audio applications offering quality playback at the lowest cost (see Figure 1).

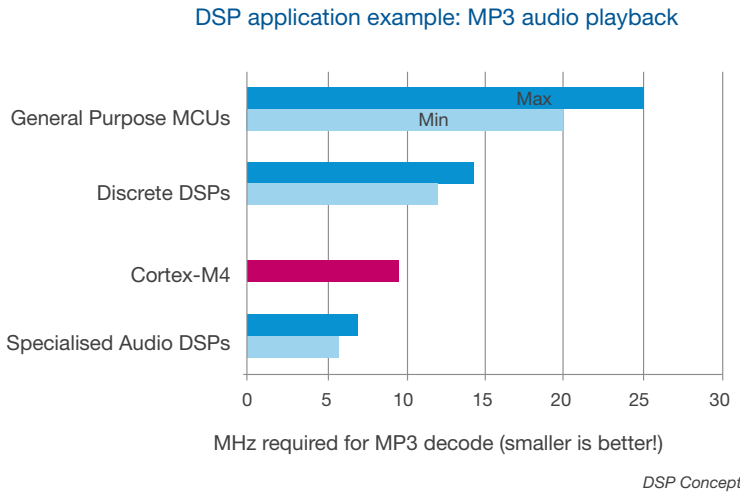

Figure 2 With its Cortex-M4 core, the STM32 F4 offers excellent audio processing capabilities that exceed the performance of many general-purpose MCUs and discrete DSPs.

The STM32 F4 offers excellent audio processing capabilities (see Figure 2). With its rich peripheral integration, a single STM32 F4 can provide a cost-effective, single-chip solution for implementing embedded audio that combines performance, ease-of-use, connectivity, and signal processing to achieve quality audio playback. Key capabilities of the STM32 F4 for accelerating audio design, enhancing performance, and lowering system cost include:

Digital Signal Processing Instructions: With the STM32 F4, developers have access to up to 105 DSP-specific instructions. These instructions include single-cycle multiply-and-accumulate (MAC), saturated arithmetic, and both 8- and 16-bit SIMD integer operations. Its architecture is designed to enable high-quality audio in consumer electronics and embedded applications in a more cost-effective manner than is possible with DSPs.

Floating-Point Unit: All STM32 F4 devices also have an integrated floating-point unit (FPU). While signal-processing algorithms can be implemented using fixed-point arithmetic, this approach adds complexity in that underflow and overflow need to be manually managed. In addition, fixed-point processing offers less dynamic range than floating-point, which impacts many audio functions. With the integrated FPU, there is no penalty for retaining this precision. Code based on floating-point can also be substantially faster and requires less memory than fixed-point code.

32-bit Efficiency: The bus size of the processor has a tremendous impact on both performance and power efficiency. Even if audio samples are streaming at 16 bits, the system still needs 32-bits to store intermediate computations. A 16-bit MCU or DSP, for example, requires seven operations (4 multiplies and 3 additions) just to complete a single 32 x 32 multiplication. The STM32 F4 can execute a 32-bit MAC (multiply and accumulate) with only one single-cycle operation.

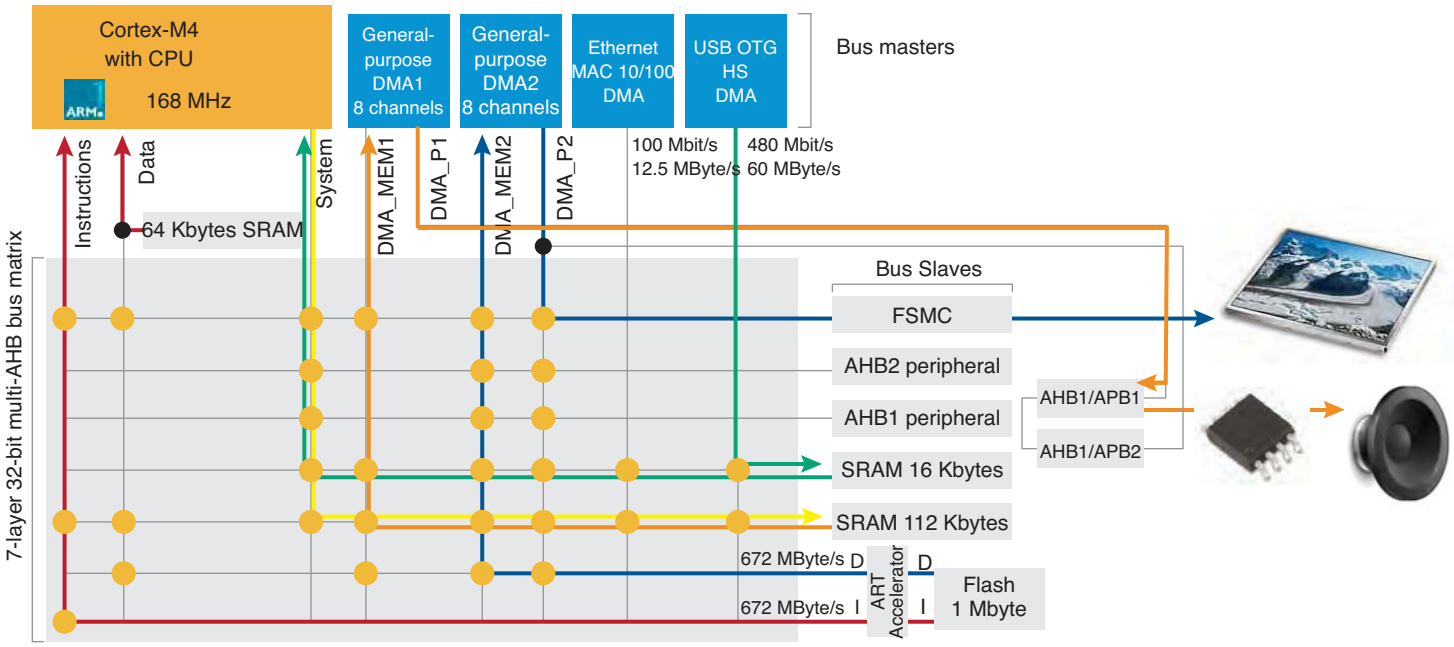

Figure 3 The multi-layer matrix that interconnects STM32 MCUs with peripherals and memory enables simultaneous transfer between multiple masters and slaves without requiring involvement from the CPU. This provides STM32 MCUs with a tremendous interconnect capacity that eliminates peripheral and memory access bottlenecks for the highest operating performance.

Multi-Layer Bus Fabric: The key to real-time signal processing is maintaining efficient data flow. In a consumer audio device, however, the MCU must move not only signal data but manage program memory, communication ports, and other system tasks. The complexity and real-time nature of audio algorithms also requires them to be integrated with application code to ensure that neither core application tasks nor audio playback compromise each other.

The STM32 architecture is designed to minimize this problem so that developers do not need to spend time resolving potential conflicts. This is achieved through the low interrupt service overhead of STM32 devices combined with the multi-layer bus fabric that allows multiple DMA transactions to occur simultaneously without burdening the CPU. Figure 3 shows the high level of parallelism that can be achieved through simultaneous transfers over a multi-layer fabric:

- Program code is executed from Flash with data stored in SRAM (red)
- The compressed audio stream is received over USB and stored in SRAM (green).
- CPU with DSP and FPU functionality accesses the

compressed audio stream for decompression and signal processing (green)

- Decompressed MP3 data is sent from the CPU to SRAM (yellow)
- Audio data is output to I2S through DMA (orange)
- Graphical icons are transferred from Flash to the display through DMA (blue)

Communications Interfaces: Users want to be able to access audio data from different sources and over different interfaces. With the right mix of interfaces—including USB (host and device), Ethernet for Internet Radio, SDIO, and external memory—developers can create flexible devices that support a wide range of usage models.

In addition to being able to receive data without loading the CPU, developers need to be able to address the many issues related to streaming audio, including lost packets and lack of feedback controls. For example, USB feedback controls to prevent under and overflow of the audio buffer are not always used or well implemented. This can result in lost or dropped packets that impact audio quality. To overcome this limitation, developers can utilize sample-rate conversion (SRC). SRC is also useful for converting between audio speeds (i.e., clock domains) while maintaining audio fidelity, compensating for slight mismatches in clock speed, or for mixing audio from different sources. For applications that need SRC, the STM32 F4 requires only 10% utilization, leaving plenty of headroom for other signal-processing tasks.

Multiple Clock Sources: Consumer audio systems require a number of different clock domains—including the CPU, USB, and I2S—that have fixed frequencies and need to be accurate as well as free of jitter. Trying to use the same clock for each of these can impact precision. For example, it is straightforward to achieve a clean clock at 168 MHz for the CPU, 44.1 KHz for an I2S interface or 48 MHz for USB but not for all three using a single clock source.

The STM32 F4 integrates two PLLs for increased clocking flexibility. The main PLL is used to generate the system clock and the second PLL is available to generate the accurate clocks needed for high-quality audio.

Complete audio system

	STM32 F2 CPU load	STM32 F4 CPU load	Flash footprint	RAM footprint
MP3 decoder	17%	6%	23k	12344
MP3 encoder	22.5%	9%	25k	16060
WMA decoder	17.5%	6%	45k	36076
AAC+ v2 decoder	25%	11%	54k	87000
Channel mixer	2.5%	2%	0.6k	16
Parametric Equalizer	16%	12%	2k	300
Loudness Control	4.5%	3.5%	3.25k	632
SRC	22.5%	10%	17.5k	1880

Figure 4 When computing 16- and 32-bit DSP functions, theSTM32 F4 offers a 25–70% improvement. As a result, systems can drop into sleep mode faster to conserve power or run more algorithms to further improve audio quality.

In addition to being able to receive data without loading the CPU, developers need to be able to address the many issues related to streaming audio, including lost packets and lack of feedback controls. For example, USB feedback controls to prevent under and overflow of the audio are not always used or well implemented.

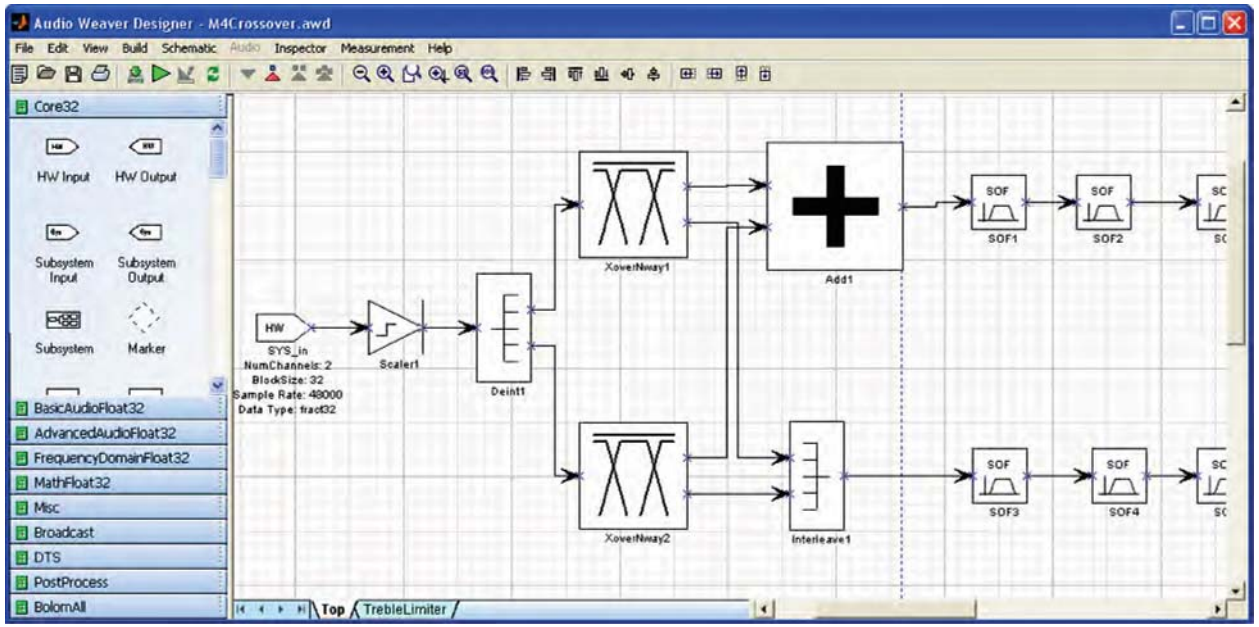

Figure 5 Audio Weaver from DSP Concepts offers a GUI-based development environment that enables developers to design the signal flow for their application by selecting processing blocks and connecting them using a drag-and-drop editor.

The ability to source different clock domains enables designs based on the STM32 architecture to maintain a permanent USB connection and avoid audio synchronization issues.

Integrated Audio Interfaces: The STM32 F4 has two full-duplex I2S standard stereo interfaces offering less than 0.5% sampling frequency error. There is also an external clock input to the I2S peripheral if an external high-quality audio PLL is preferred. In addition to simplifying design, integrating the I2S interfaces reduces component count, board size, and system cost.

MCU Peripherals: The STM32 architecture includes all of the real-time peripherals required for even the most demanding MCU-based application.

The combination of the STM32 F4's capabilities brings a new level of performance to audio applications. Performing a long 32-bit multiply or multiply-accumulate (MAC) operation on an STM32 F2, for example, takes 3-7 cycles. With the STM32 F4, this operation is performed in a single cycle. When computing 16- and 32-bit DSP functions, the STM32 F4 offers a 25-70% (see Figure 4) improvement. As a result, systems can drop into sleep mode faster to conserve power or run more algorithms to further improve audio quality.

In addition to the integrated DSP capabilities of the STM32 F4, developers have access to the CMSIS DSP library to accelerate development. The CMSIS DSP library includes a large number of DSP and floating-point functions optimized for the algorithms commonly used in audio applications. This library is supplied by ARM for processors built around the Cortex-M4 processor. DSP Concepts is the company that wrote the CMSIS DSP library. They have leveraged their intimate knowledge of the library to create the audio blocks that make up their signal processing design tool, Audio Weaver.

Audio Algorithm Design

Audio Weaver enables developers to quickly design the audio processing portion of their system; i.e., everything that goes on between receiving an audio signal and outputting it. Audio Weaver offers a GUI-based development environment that enables developers to design the signal flow for their application by selecting processing blocks and connecting them using a drag-and-drop editor (see Figure 5). Each block has hand-optimized code behind it, and the tool automatically creates the required data structures.

Because complex functions are built from base audio functions, the final code executes with no performance or efficiency losses compared to hand-coding from scratch.

When algorithm code is written by hand, each design iteration requires substantial time investment since the code must be optimized and tuned to see what its actual impact on sound quality and processing load are. With Audio Weaver, the design cycle is much faster, giving developers the ability to explore more configurations in their efforts to increase sound quality while reducing system cost. Code is highly optimized for MIPS and memory usage, supports floating-point processors such as the STM32 F4, offers flexible deployment modules, and does not require an RTOS to operate. The library includes over 150 different audio blocks, including third-party IP.

With tools like Audio Weaver, it has become possible to create highly tuned audio applications without engineers needing to have a deep knowledge of audio processing. For companies new to audio, complete reference designs are available, with assistance from DSP Concepts to tune them for the final production system. Companies that are comfortable with audio processing can work with individual audio blocks that provide basic functionality and build them into higher-level processing algorithms. Even sophisticated companies can accelerate design using Audio Weaver as it provides a framework with core components that not only jumpstarts design with highly optimized code but provides a development environment that facilitates fast prototyping and tuning. For these companies the value-add of Audio Weaver is faster time-to-market.

Accelerating Optimization

To speed design, Audio Weaver supports cross-platform development. The ability to run the same algorithms on a PC as on the STM32 F4 gives engineers a powerful environment in which to design and tune the software in parallel with hardware development. Once target hardware is available, the code can be retargeted for the STM32 F4 and final optimizations made, resulting in significant time-to-market savings.

During final optimization, developers can profile the MIPS and memory required by each audio processing stage. This enables engineers to measure how much a particular improvement in sound quality will cost in terms of CPU utilization to determine the most efficient use of processing resources when many functions have to operate simultaneously.

When algorithm code is written by hand, each design iteration requires substantial time investment since the code must be optimized and tuned to see what its actual impact on sound quality and processing load are. With Audio Weaver, the design cycle is much faster, giving developers the ability to explore more configurations in their efforts to increase sound quality while reducing system cost.

Consider the use of different order filters to equalize the speakers. A lower-order filter, for example, may provide a frequency response that is 3 dB off of the ideal response while a higher order filter is off by only 1 dB. The relative difference in CPU utilization between these two filters can be used

to determine where to allocate CPU resources to maximize sound quality.

At the end of the day, however, audio quality is not about response graphs but how it actually sounds to people. With many development systems, engineers have to make adjustments to code, recompile, and download code before they can hear a new configuration. However, to assess the impact of a lower-order filter on quality, for example, developers need to be able to hear both configurations right after each other.

Audio Weaver solves this problem by supporting a tuning interface that can change filter characteristics in real-time. With the ability to configure and switch between multiple settings with the click of a button, developers can compare two sets of speaker equalizations or different spatial processing. Note that the tuning interface is seamless and transparent, compared to instrumenting code that can impact quality because of extra loading on the CPU.

The ability to tune quickly and easily without recompiling can substantially shorten the time it takes to optimize a system. Flexible tuning also simplifies the optimization process for developers new to audio.

Note that audio applications are not comprised solely of audio processing. To accelerate system design, DSP Concepts also provides an extensive range of software functionality beyond its extensive audio module library, including:

- Real-time kernel
- Audio I/O management
- PC/host control interface
- Boot loader
- Update manager
- Flash file system

System-level Design

One of the challenges to adding audio to embedded designs is that while many MCU manufacturers offer reference designs, audio is typically not one of the applications supported.

To address this shortcoming, ST has invested significantly in creating digital audio resources for its customers in order to offer complete audio reference designs as well as tools that enable the design of quality audio optimized for the STM32 architecture. For example, ST and its partners offer a variety of evaluation boards with audio capabilities. ST also offers several docking station reference designs that provide a representative design that can be used in a wide range of embedded applications.

For Apple Made for iPod (MFI) licensees, ST offers the Apple iAP application, a complete solution based on STM32 F2 and STM32 F4 devices to deliver a high-quality music experience. The Apple iAP application support both simple accessory and audio streaming accessory for iPod, iPhone, and iPad devices. Components include:

- Either the STM322xG-EVAL or STM324xG-EVAL board to which developers connect their Apple Authentication Coprocessor (ACP) circuit
- Free Apple “iPod Accessory Protocol” (iAP) firmware with Lingoes for authentication and control/information data
- Free USB Host Library with USB Host HID class for control and information data

For audio streaming accessories, the Apple iAP application also supports:

- Free USB Host Library USB Host Audio classes
- Remote iPod/iPhone/iPAD control
- Digital audio streaming
- Music tag extraction
- Flash card reader capabilities, such as using an SD card or MMC, that can decode audio files from this media. Optimized decoders are provided for this purpose free of charge

Today’s consumer audio devices are complex systems that require both high performance to support quality playback and flexibility to meet rapidly changing market expectations. With its high performance core, efficient multi-layer bus fabric enabling simultaneous data transactions, and the right mix of MCU peripherals and connectivity, the STM32 F4 is an ideal architecture for many embedded and consumer audio applications. Developers can now design systems offering synchronized digital audio playback of the highest quality using a single MCU.