Abstract
I propose to design and build an audio playback device for use in broadcast radio. It will permit the operator of a radio station to instantly play back audio elements, and manipulate them in real-time. While it can function with any audio element, its primary application is for “donut” elements. Donuts are in wide use in the radio industry today, but are currently rudimentarily implemented. A donut is a pre-produced element that begins with a recorded announcer speaking called an intro, transitions into a music or sound bed for a live operator to speak over, and finally returns to a recorded announcer speaking called an outro before finishing.

My device will implement a new concept, which I call the Dynamic Donut. The Dynamic Donut allows the operator to manipulate the length of the music bed inside the donut in real-time. This means instead of being restricted to a specific length of time to talk, or alternatively having sharp and mismatched transitions between the bed and outro, the operator can speak as little or as long as he or she requires, while still maintaining a consistent flow throughout the whole element.

Background and Benefits
Systems to provide quick playback of audio in a broadcast radio environment exist today in both hardware and software form. A popular hardware example is made by 360 Systems, the Instant Replay 2 shown in Figure 1. Audio is stored on an internal 3.5” hard disk, and transferred onto and off of the system via FTP to its integrated FTP server. The fifty hot keys on the front each provide instant playback of one audio element. Users can switch between 10 banks of hot keys, providing access to 500 audio elements. The Instant Replay 2 has a looping feature, but its use is limited by the fact that looping is restricted to entire audio elements, looping portions of elements is impossible.

Comparable software systems are usually built into radio automation systems, software suites which handle a wide variety of playback responsibilities. A popular system is Google Radio Automation, shown in Figure 2. Audio is stored on the automation system server. Hot keys grow and shrink to fit available space on the screen, and unlimited banks provide access to unlimited audio elements. This particular system has no support for looping. While some others do, support is always limited to the entire audio element.

In both of these types of systems, donuts are handled one of two ways. The first is to pre-produce the donut with a specific length music bed sandwiched between the intro and outro. This has the advantage of allowing the entire element to seamlessly flow from beginning to end as one congruent piece. The downside is the operator now has a fixed length of time to fill. Speak too little and the transition sounds awkward, with a significant delay between the operator and announcer. Speak too much and the operator has overrun the bed, and is now speaking at the same time as the announcer.
The second solution is to separate the intro and bed from the outro. The operator first plays the first audio element, which includes intro and bed. The operator talks over the bed for as long as necessary. Finally, the operator starts the outro and turns down the bed at the same time. The clear advantage is that now the operator can speak as little or as long as needed, easily accommodating different situations. The downside is the bed and outro are now disjoint, and the entire element sounds more scattered.

In contrast to traditional donut implementations, my Dynamic Donut will maintain a smooth flow throughout the audio element, while allowing the operator to adjust the length of the bed in real time. Audio will be stored on external solid state storage, allowing for easy copying to and from the device, as well as allowing capacity to be easily upgraded. Audio elements will be organized hierarchically in folders, each folder supporting an unlimited number of audio assets. Folders and audio elements will be navigated using traditional up, down, enter, and back keys. Hot keys will be provided, and can either be assigned globally across the entire file system, or locally per-folder.

**Design and Implementation**

The magic behind the Dynamic Donut is its looping scheme. The system seamlessly loops the bed inside the donut to provide a never ending bed. Since the in and out points of the loop are specified by the producer when designing the element, the loop is undetectable to listeners. In addition to loop in and out points, the producer also specified break points. These are points in the bed which the producer has indicated will flow perfectly to the loop outro. Therefore, if the audio element plays to the loop out point or any of the break points, and then jumps to the outro, the result will sound natural and uninterrupted.

Large, clear displays will give feedback to the operator as to how soon a break or out point is. Since break points can reasonably be placed as little as one to four seconds apart, it’s easy for operators to time their talkset to end naturally. The operator can tell the Dynamic Donut to exit at the next break point either by pressing a button on the control pad, or through their console, automation system, or external button through external logic connections.

The Dynamic Donut will be implemented as an embedded system with a microcontroller at its core. It will be packaged in a sturdy case that takes the approximate footprint of a computer keyboard, with additional height. The microcontroller receives input from a control pad, which is a matrix of buttons on the case. The microcontroller provides output to displays and indicators on the case, and a digital audio stream to one or more CODECs. The microcontroller both receives input from and provides output to a controller for storage, and a logic driver for external control. A block diagram of the system is provided as Figure 3.

---

**Figure 2 - Google Radio Automation**

---

Matthew Cohn  
10/31/2008  
Dynamic Donut  
ETEC 471 Project Proposal
Radio studios are tightly integrated to permit a fast workflow. To facilitate this, my system will contain a suite of industry standard connectors. Audio will be output in either digital AES/EBU, in balanced analog on XLR3 connectors, or both. External logic input and output will be provided on RS-232, contact closures, or both. All of these formats are industry standard and allow my device to communicate with the widest possible array of external equipment.

**Social and Global Impacts**

The radio industry has faced some of the worst effects of consolidation in the past several years. One company, Clear Channel Communications, owns and operates more than 900 individual radio stations in the United States. Because of their massive size, they are able to invest in expensive technology and infrastructure allowing them to reduce per-station costs. My Dynamic Donut is an inexpensive way to provide small market and independent stations resources to sound competitive to consolidated station groups.
The device itself will be RoHS compliant, consisting of RoHS compliant components. This has economic benefits, allowing my device to be sold in European markets, health benefits, allowing for safe disposal after the life of the product, and environmental benefits, eliminating the use of many environmentally hazardous materials. Additionally, I will configure my processor to enter a low power mode during unused clock cycles. Less energy consumption also means less unnecessary heat generated, which further reduces energy consumption by not needlessly taxing studio air conditioners.

**Development and Demonstration**

After extensive research, I have chosen the MCF5223x microcontroller family from Freescale Semiconductor. My system should be able to decode as wide a variety of compressed audio formats as possible, to avoid generational quality loss by transcoding audio to a compatible format. MPEG1 Layer 2 is the most common lossy audio compression used in broadcast, and lossless is also commonly used. Freescale provides both MPEG1 Layer 2 (.mp2) and MPEG1 Layer 3 (.mp3) decoders for their ColdFire family. The MCF5223x has between 128kb and 256kb of Flash, and 32kb of SRAM. While not excessive, these resources should comfortably meet my requirements. The MCF5223x family includes a built-in Ethernet MAC, which allows for future expansion to an Ethernet enabled device. Freescale has a low-cost development board for the MCF52233 available, the M52233DEMO. I have purchased and become familiar with this board. I have also installed, configured, and successfully used the following collection of development tools:

- Eclipse Platform 3.4.0.M20080911-1700
- Eclipse CDT 5.0.1
- Zylin Embedded CDT 4.5.1
- CodeSourcery GNU Toolchain for ColdFire Processors 2008q1
- Freescale CF Flasher 3.1.9

Finally, I have written a collection of tools including embedded operating system, startup code, makefile, linker script, and vector file which will facilitate rapid development on the MCF5223x.

I will construct a prototype of the system for demonstration. It will feature everything listed in the block diagram above including controls, displays, and external interfaces. I will use the following equipment to help demonstrate my product:

- Audio mixer with balanced analog inputs (One mic and two stereo input minimum)
- Microphone (Shure SM58 or equivalent)
- Headphone distribution amplifier (Four channel minimum)
- Headphones (Sennheiser HD202 or equivalent)
- Stereo monitor amplifier
- Stereo monitors
- Laptop with RS-232 interface
- Broadcast automation software (Demo, OMT iMediaTouch v3 or equivalent)

I have or should be able to borrow all of this equipment. Demonstrating my product with professional broadcast equipment not only portrays a typical installation environment, but showcases its interfaces. Because many products will be demonstrated at the same time in the same lab, I will keep noise to a minimum. Headphones will allow my visitors to clearly and comfortably hear the output of my product, as well as generate interest from passing observers.