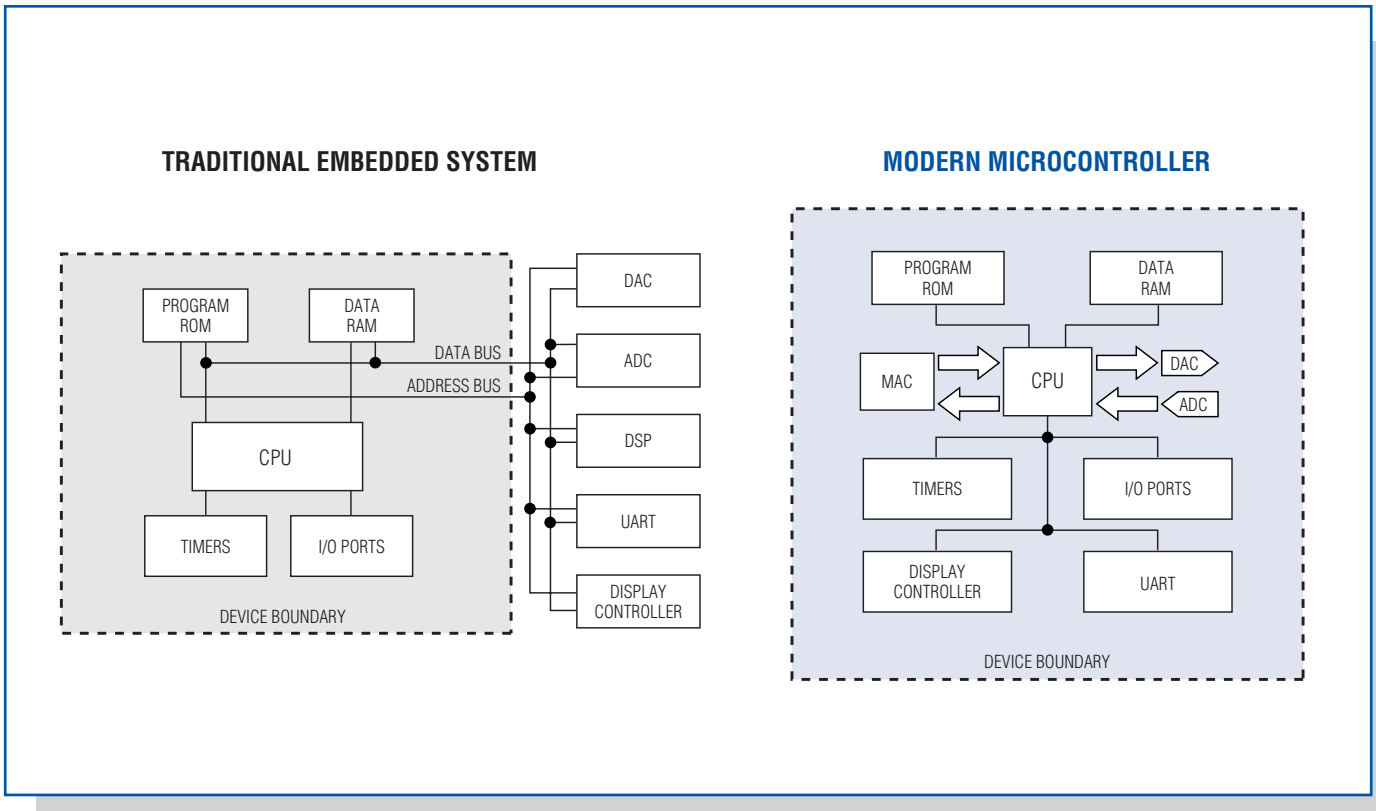
DALLAS SEMICONDUCTOR MAXIM

Engineering Journal

Volume Fifty-Five



A traditional embedded system must expose an address and data bus for the attachment of external I/O devices to complete the system. Modern µCs integrate most of the necessary system functions within the device, including analog peripherals and a multiply-accumulate unit (MAC). (See article inside, page 3.)

News Brief

MAXIM REPORTS 19% INCREASE IN EARNINGS PER SHARE AND 32% INCREASE IN FREE CASH FLOW PER SHARE FOR ITS THIRD QUARTER OF FISCAL YEAR 2005

Maxim Integrated Products, Inc., (MXIM) reported net revenues of $400.2 million for its fiscal third quarter ending March 26, 2005, an 8.2% increase over the $370.0 million reported for the third quarter of fiscal year 2004 and an 8.2% decrease over the $436.1 million reported for the second quarter of fiscal 2005. Net income for the quarter was $125.5 million, a 15.0% increase over the $109.2 million reported last year and a 13.2% decrease from the $144.6 million reported for the second quarter. Diluted earnings per share were $0.37 for the third quarter, a 19.4% increase over the $0.31 reported for the same period a year ago and an 11.9% decrease from the $0.42 reported for the second quarter of fiscal 2005. The Company's free cash flow was $156 million, or $0.45 per diluted share for the third quarter, compared to $120 million, or $0.34 per diluted share, for the third quarter of fiscal year 2004. This is a 32.4% increase in free cash flow per diluted share from the prior year's comparable quarter. For the nine months ended March 26, 2005 free cash flow per diluted share increased 20.8% relative to the comparable period last year. Free cash flow is defined as cash from operating activities (after tax), less additions to property, plant and equipment as reported in the Company's statements of cash flows.

During the quarter, cash and short-term investments increased $116.5 million after the Company repurchased approximately 598,000 shares of its common stock for $24.5 million, paid dividends of $32.7 million, and acquired $13.8 million in capital equipment. At quarter end, cash and cash equivalents totaled $1.4 billion. Accounts receivable increased $12.7 million to $192.4 million, and inventories increased $12.9 million to $159.5 million in the third quarter.

Research and development expense was $83.1 million or 20.8% of net revenues in the third quarter, compared to $81.0 million or 18.6% of net revenues in the second quarter of fiscal year 2005. Selling, general and administrative expenses decreased slightly from $25.3 million or 5.8% of net revenue in the second quarter to $24.7 million in the third quarter, or 6.2% of net revenue.

In the third quarter, the Company recorded a $5.0 million discretionary employee bonus to be paid out at the end of calendar 2005, which increased cost of goods sold by $1.6 million, increased research and development expense by $2.7 million, and increased selling, general and administrative expenses by $0.7 million. The effect of this bonus reduced earnings by $0.01 per share.

Third quarter bookings were approximately $373 million, a 6% increase from the second quarter's level of $353 million. Turns orders received in the quarter were approximately $156 million or 42% of net bookings, a 28% increase over the $122 million or 35% of net bookings received in the prior quarter (turns orders are customer orders that are for delivery within the same quarter and may result in revenue within the same quarter if the Company has available inventory that matches those orders). Bookings increased in 15 out of the Company's 20 end equipment segments and in all geographic locations except Europe. Third quarter ending backlog shippable within the next 12 months was approximately $328 million, including approximately $284 million requested for shipment in the fourth quarter of fiscal year 2005. The Company's second quarter ending backlog shippable within the next 12 months was approximately $370 million, including approximately $300 million that was requested for shipment in the third quarter of fiscal year 2005.

Jack Gifford, Chairman, President, and Chief Executive Officer, commented:"As expected, third quarter bookings improved over the previous quarter. It is a positive sign that a significant majority of our business units participated in this bookings increase. Turns orders accelerated in the quarter as compared to the previous quarter leading us to believe that inventory levels have come further in line with consumption levels."

Mr. Gifford continued: "Design-in activity of Maxim's proprietary products remains strong which we believe positions us to gain market share over the next few years. During the third quarter, the Company introduced 738 engineering man-months of new products, a Company record and a 31% increase over the second quarter."

Mr. Gifford concluded: "The Company's Board of Directors has declared a quarterly cash dividend of $0.10 per share. Payment will be made on June 1, 2005 to stockholders of record on May 16, 2005."

For the complete Q305 press release, including safe harbor information, go to: www.maxim-ic.com/NewsBrief

High-Performance Computing with Low-Cost Microcontrollers

More than ever, microcontrollers (µCs) include DSP-like functions, in addition to more conventional ALU cores. Unfortunately, many firmware developers, unfamiliar with DSP algorithms and coding techniques, fail to take full advantage of the capabilities offered in these devices. This article is a firmware developer's primer for the use of signal-processing techniques in low-cost µCs.

The Changing Role of Microcontrollers

It was not so long ago that designing with a µC meant a multitude of peripheral and support chips surrounding the CPU itself. Even if the CPU had sufficient embedded ROM and RAM, any non-trivial project would require functions provided by external support chips.

This was never more true than when the system involved analog signals of any kind. Introduce even the simplest analog-signal requirement, and system complexity would balloon. But, that is the problem—the real world of sounds, temperatures, pressures, and other natural phenomena remains persistently analog.

In actuality, a system capable of operating in the analog world requires:

- A transducer and analog analog-signal conditioner to translate the physical phenomenon to an electrical signal
- A data-acquisition subsystem that would, at a minimum, consist of an analog-to-digital converter (ADC) and a digital-to-analog converter (DAC) of sufficient accuracy and speed
- A digital-signal processor (DSP) capable of processing the sampled data
- A µC with enough throughput to manage the DSP, the data-acquisition subsystem, and other peripherals, and with enough remaining bandwidth left to smoothly execute control and user-interface functions; the µC may also possibly contain program ROM and data RAM

These complex devices raised the cost of the end product, pricing it out of the range of lower-end applications. However, a new class of µC has emerged that eases the interface to the analog world by embedding high-performance data-acquisition subsystems, DSP-like functions, and a RISC CPU core on the same die. An illustration of the integration capable in modern µCs is shown in **Figure 1**.

An example of the new generation of µC is the MAXQ3120. In addition to the usual array of UARTs, timers, and I/O ports, the MAXQ3120 integrates a set of peripherals that make analog interfacing easy:

- A 16-bit, 1-cycle RISC core
- A pair of 16-bit, sigma-delta ADCs
- A 16-bit timer capable of operating in PWM mode
- A 16 x 16 multiplier with a 40-bit accumulator

The MAXQ3120 also includes a time-of-day clock, an LCD controller, and hardware to ease interfacing to an IR communications channel. In the following examples, cycle counts or storage requirements will refer specifically to the MAXQ3120 µC.

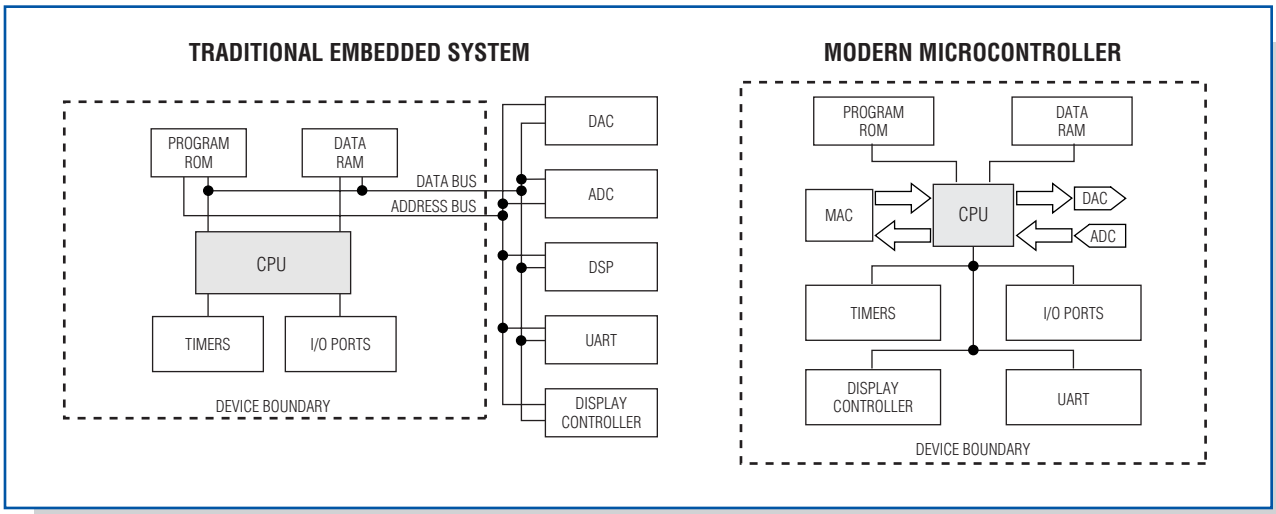


Figure 1. A traditional embedded system must expose an address and data bus for the attachment of external I/O devices to complete the system. Modern µCs integrate most of the necessary system functions within the device, including analog peripherals and a MAC.

Thinking about the Problem

Problems that involve signal processing are best approached in three phases:

1. **What do you want to do?** Carefully dissect the problem and break it into the smallest functional units possible. Does the problem involve filtering, signal discrimination, or generation of a particular type of signal? When you have divided the problem into a series of simpler subtasks, you are ready for the next step.
2. **How do you do it?** Now search the literature, the web, or other resources to find the algorithms that pertain to each task. You will likely find that most of your conceptual problems have been solved already. All you have to do is the last step.
3. **Code it.** The task of converting the algorithms into code is a familiar one, and the hardware resources available make it easier than ever.

In the following examples, we adhere to three steps: describing the problem, presenting algorithms and methods, and providing a code example for solving the problem.

Power Monitoring and Measurement Example

One common example of analog-to-µC interfacing is measuring the voltage and current of an AC circuit to determine the power consumed by a load. At first, this seems relatively straightforward—if the voltage and current waveforms are sinusoidal, the RMS voltage and current are just $1/\sqrt{2}$ times the peak voltage or current. Multiply the RMS voltage and current together, and the result is power in watts. What could be simpler?

There are two problems with this analysis. First, while the voltage delivered from the a power company is often very nearly sinusoidal, the current waveform is not. Lamp dimmers, switching power supplies, and fluorescent lighting all present a nonsinusoidal current profile to the power line. Simple multiplication by a constant will *not* give the RMS current value. Second, even if the current waveform is sinusoidal, unless the voltage and current waveforms are precisely in phase, simple multiplication of the RMS values does not provide the actual power usage in watts. In general, real-world loads contain either capacitive or, more often, inductive reactance. Therefore, reactive power must be considered. We must return to fundamentals to find a better way.

At each instant in time, the product of instantaneous voltage and current is instantaneous power. **Figure 2** illustrates how this instantaneous power rises and falls, and may even dip below zero for part of the cycle, depending on the current flow. The total real power is simply the time average of the product of instantaneous voltage and instantaneous current. Finding the real power is simple—multiply the voltage sample and the current sample, and add the result to an accumulator. After a sufficient number of samples have been accumulated, just divide the sum by the number of samples to obtain the power in watts. To convert this to energy in watt-seconds, multiply by the time in seconds over which the samples were accumulated.

Calculation of reactive power is not as straightforward. First, note that, for sinusoidal voltage and current, the reactive power is defined as:

$$Q = VI\sin\theta$$

where V is the RMS voltage, I is the RMS current, and θ is the phase difference between the current and voltage waveforms.

The power due to reactive components can be determined at any time by calculating a difference:

$$Q_{+\tau} = \int V_t I_{t+\tau} dt = VI\cos(\theta + 2\pi\tau/T)$$

and

$$Q_{-\tau} = \int V_t I_{t-\tau} dt = VI\cos(\theta - 2\pi\tau/T)$$

where V_t and I_t are the instantaneous values for voltage and current at any time, τ is any arbitrary delay, and T is the period of one line cycle. The difference is:

$$Q_{-\tau} - Q_{+\tau} = VI(\cos(\theta - 2\pi\tau/T) - \cos(\theta + 2\pi\tau/T))$$

Using the trigonometric identities cos(A + B) = cosAcosB - sinAsinB and cos(A - B) = cosAcosB + sinAsinB changes the equation to:

$$Q_{-\tau} - Q_{+\tau} = VI(\cos(\theta)\cos(2\pi\tau/T) + \sin(\theta)\sin(2\pi\tau/T) - \cos(\theta)\cos(2\pi\tau/T) + \sin(\theta)\sin(2\pi\tau/T))$$

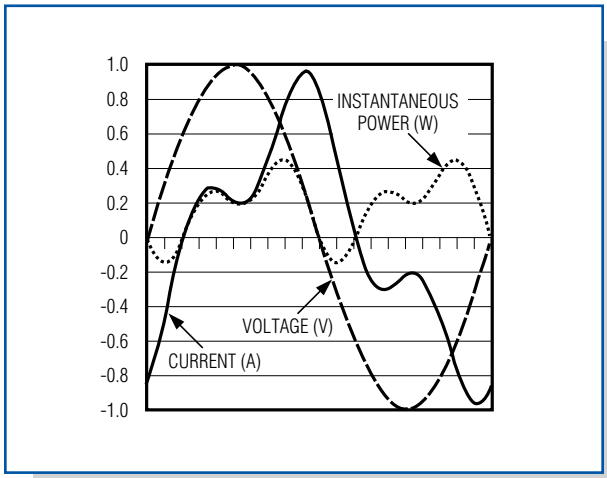


Figure 2. While the voltage waveform (dashed line) is often very nearly sinusoidal, the current waveform (solid line) may vary significantly from a pure sinusoid. Additionally, the current waveform may be shifted in phase, causing instantaneous power (dotted line) at some parts of the line cycle to become negative.

Note that the cosine terms cancel and the sine terms reinforce, leaving only:

$$Q_{-\tau} - Q_{+\tau} = VI2sin(\theta)sin(2\pi\tau/T)$$

But VI sin(θ) is Q, and that is what we are trying to find. So, rearranging the terms leaves:

$$Q = \frac{Q_{-\tau} - Q_{+\tau}}{2sin(2\pi\tau/T)} = \frac{1}{2sin(2\pi\tau/T)}\int(I_tV_{t-\tau} - I_{t-\tau}V_t)dt$$

This means that the reactive power can be calculated by accumulating the difference between the product of the most recent current sample and the previous voltage sample, and the most recent voltage sample and the previous current sample. The denominator in the above formula is a constant if τ is set to the sample period, so it can be precalculated.

Once the real power and reactive power are known, the *apparent* power (the power that would be calculated if the RMS voltage and current were multiplied) can be easily calculated as:

$$P_{app} = \sqrt{P^2 + Q^2}$$

Finally, the power factor is found by dividing the real power by the apparent power.

Assuming, for the moment, that tracking of all of these factors is desirable, an application would need to keep track of the sum of the product of the voltage channel and the current channel, and the sum of the product differences described in the previous discussion on reactive power. The code for accumulating real power would look like:

```
void accumulateRealPower(int i_sample, int
v_sample)
{
       static long real_power;
       initMAC(MULTIPLY_ADD);
       preloadMAC(real_power);
       real_power    =    getMAC(v_sample,
i_sample);
}
```

In this example, the `initMAC` function takes a byte that sets operational parameters in the MAC. The `preloadMAC` function loads a long variable into the accumulator prior to the multiply operation.

To demonstrate how this can translate into actual machine instructions, consider the following set of operations based on the above code:

1. Initialize the multiply-accumulate unit, or MAC (one cycle)
2. Set the memory pointer to sum the storage area (one cycle)
3. Load the old sum into the accumulator (two cycles)
4. Load the voltage sample into the multiplier (one cycle)
5. Load the current sample into the multiplier (one cycle)
6. Wait one cycle for the accumulate operation to complete (one cycle)
7. Save the accumulator to memory (one cycle)

Thus, eight cycles are required to accumulate real power.

Accumulating reactive power is similar:

```
void accumulateReactivePower(int i_sample, int
v_sample)
{
       static long reactive_power;
       initMAC(MULTIPLY_ADD);
       preloadMAC(reactive_power);
       reactive_power = getMAC(prev_v_sample,
i_sample);
       initMAC(MULTIPLY_SUB);
       reactive_power = getMAC(prev_i_sample,
v_sample);
}
```

Note that this discussion assumes that the input samples have zero DC offset. If this is not true, it will be necessary to provide additional accumulators for the voltage and current channels by themselves. If there is zero DC offset, these sums will be zero. If not, the power represented by the DC offset must be subtracted from the accumulated real power.

Filtering Example

Filtering is one of the most common tasks to be performed in the digital realm. The reason is simple—ideal filters, unrealizable in the analog world, are relatively simple to implement in digital logic.

In this section, a lowpass filter and a bandpass filter are presented. Lowpass filters are commonly used to remove unwanted, high-frequency components from a signal prior to sampling for elimination of aliasing artifacts. Bandpass filters are often used to restrict a communication channel to a particular range of frequencies. An FSK modem may, for example, use a bandpass filter to eliminate both high- and low-frequency noise components, leaving only the frequencies of interest to be processed.

Volumes have been written on the topic of digital filters, possibly leading one to believe that the topic is difficult to fathom. In reality, the basics are simple, and one need not be a DSP expert to make use of filtering functions.

By inspecting the lowpass filter diagram in **Figure 3**, it can be seen that:

$$Y_n = Y_{n-1} + b_0(X_n - Y_{n-1})$$

where Y_n is the current output sample, Y_{n-1} is the previous output sample, and X_n is the current input sample. The filter constant, b_0, is precalculated to be:

$$b_0 = \pi \Delta t f_0$$

where f_0 is the desired half-amplitude corner frequency, and Δt is the sample period. Therefore, for a sample rate of 8kHz and a desired corner frequency of 100Hz, b_0 would be equal to π x 125µs x 100Hz = 3.93 x 10^{-2}

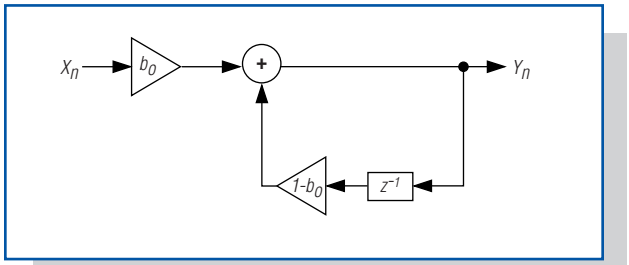


Figure 3. A single-pole, lowpass filter is used prior to signaling to remove unwanted high-frequency components from a signal.

Implementing the filter algorithm is quite simple:

```
long lpf(int input_sample)
{
        static long prev_out;
        initMAC(MULTIPLY_SUB);
        preloadMAC(prev_out);
        prev_out = getMAC(b0, prev_out);
        initMAC(MULTIPLY_ADD | ONE_OP);
        prev_out = oneopMAC(input_sample);
        return prev_out;
}
```

The `oneopMAC` function loads only one operand into the multiplier, expecting the multiplier to maintain the second operand. This is done to save execution time. This filter, if efficiently compiled, requires only ten cycles per sample. With an 8MHz clock, this means that a 1st-order lowpass filter would only consume about 1% of the MAXQ3120's processing bandwidth (without considering interrupt overhead.)

A 2nd-order bandpass filter is conceptually somewhat more complex, though practically similar, adding only a few steps to the algorithm. The signal-flow diagram for a simple, 2nd-order bandpass filter is shown in **Figure 4**, and its formula is:

$$Y_n = b_0(X_n - X_{n-2}) - a_1 Y_{n-1} - a_2 Y_{n-2}$$

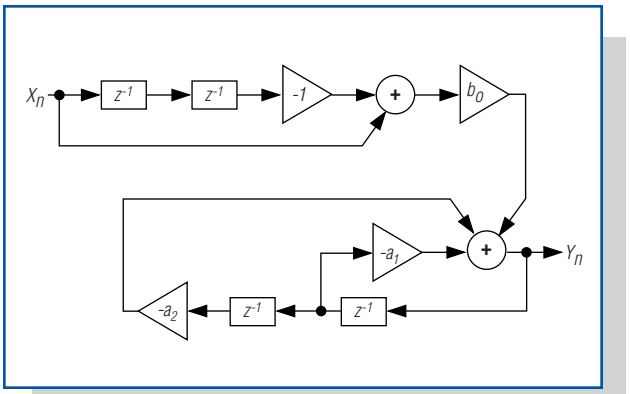


Figure 4. A 2nd-order bandpass filter can restrict a communication channel to a particular range of frequencies.

As in the previous example, the parameters b_0, a_1 , and a_2 are precalculated. The a_1 parameter defines the Q-factor of the filter—a measure of how effectively the filter rejects of out-of-band frequencies near the center frequency. Specifically,

$$a_1 = \frac{1}{p} - 2$$

with p typically in the range of 100 to 1,000. Larger values of p represent narrower passband, but longer filter settling times. Smaller values of p provide a wider passband and faster settling times.

The other parameters, a_2 and b_0, are calculated in terms of a_1. First, we define two intermediate variables:

$$k_0 = 2\pi f_0 \Delta t$$

$$k_1 = a_1(\cos k_0 + \frac{1}{\cos k_0})$$

Now we can define the other two parameters:.

$$a_2 = \frac{1}{2}(\sqrt{k_1^2 - 4a_1^2} - k_1) - 1$$

$$b_0 = \frac{\sqrt{1 + a_1^2 + a_2^2 + 2a_1(1 + a_2)\cos k_0 + 2a_2\cos(2k_0)}}{2\sin k_0}$$

Note that once the center frequency, the sampling rate, and the filter Q-factor are selected, the parameters are constants. The necessary code is:

```
long bpf(int input_sample)
{
        static long x[1], y[1], output;
        initMAC(MULTIPLY_ADD | CLEAR_ACC);
        output = getMAC(y[1], a2);
        initMAC(MULTIPLY_SUB);
```

```
        output = getMAC(y[0], a1);
        output = getMAC(x[1], b0);
        initMAC(MULTIPLY_ADD | ONE_OP);
        output = oneopMAC(input_sample);
        x[1] = x[0];
        x[0] = input_sample;
        y[1] = y[0];
        y[0] = output;
        return output;
}
```

This routine (exclusive of aging of the input and output samples) requires fourteen cycles, if efficiently compiled. Aging is not included in the cycle estimate as the samples will often be stored in a circular buffer, requiring no aging.

Tone Generation and Detection Example

In communications applications, it is often necessary to generate/detect combinations of tones in an audio channel. In telephone applications, for example, it is not uncommon to want to synthesize or detect dual-tone multifrequency tones (DTMF) in a telephone line for dialing on a telephone line. In trunk cards, standard multi-frequency (MF) tones are often used for in-band signaling. Call-progress signaling (dial tone, ringback, busy, and reorder, among others) is often presented as a combination of two or more tones.

A two-pole digital resonator (**Figure 5**) is a simple way of generating tones. The formula for the resonator is:

$$Y_n = kY_{n-1} - Y_{n-2}$$

where k is:

$$k = 2\pi f_0 \Delta t$$

and can be precalculated. Note that there is no input to this resonator—it operates continuously without intervention. To start the resonator, however, Y_{n-1} must be set to zero and Y_{n-2} set to -Asin(k), where A is the desired amplitude of the signal.

The algorithm to produce a sine wave is the simplest presented so far:

```
long gen_tone_sample()
{
        static long y[1];
        initMAC(MULTIPLY_ADD);
        preloadMAC(-y[1]);
        y[1] = y[0];
        y[0] = getMAC(y[0], k);
        return y[0];
}
```

The estimate of code size for this routine is about eight machine cycles per sample of sine wave to be generated. This has similar CPU intensity to generating a simple square wave, which must then be filtered by external analog circuitry to produce the spectral purity that many applications require.

Tone detection is somewhat more complicated, though not that daunting when pared to the essentials. To detect a tone in an audio channel, we select the Goertzel algorithm. This algorithm utilizes a 2nd-order filter and a power-measurement formula to determine the presence of energy in the filter passband.

The formula for the 2nd-order filter in **Figure 6** is:

$$Y_n = X_n + 2cos(k)Y_{n-1} - Y_{n-2}$$

where k is defined the same as in the tone generator described previously. The algorithm is:

```
long *tone_filter(int input_sample)
{
        static long y[1], output;
        initMAC(MULTIPLY_ADD);
        preloadMAC(-y[1]);
        y[1] = y[0];
        y[0] = getMAC(y[0], twocosk) +
input_sample;
        return y;
}
```

Notice that this routine returns an array rather than a

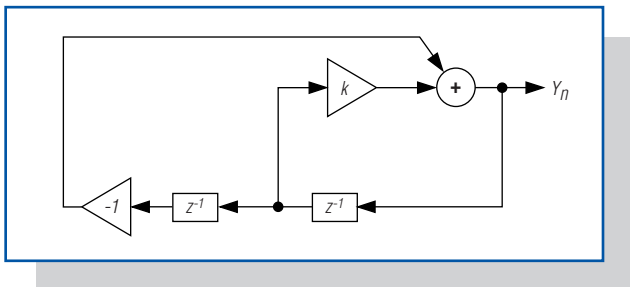


Figure 5. A two-pole digital resonator is used for sine-wave generation.

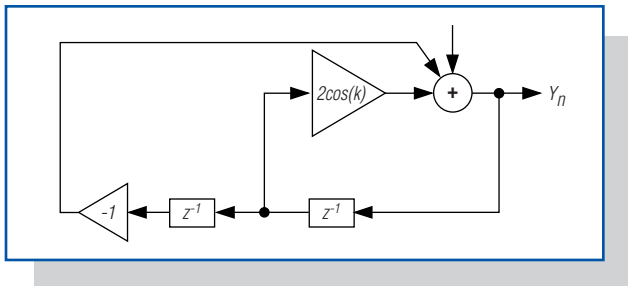


Figure 6. Tone detection can be performed by a 2nd-order filter.

scalar value. This is because the next step needs both the most recent result and the previous result. The estimate for the cycle count for one tone-detector channel is 12 cycles, or, at a sampling rate of 8kHz, about 1.5% of the CPU horsepower. In general, it will require more than 100 sample periods before the filter settles and provides a reliable indication of tone presence.

To test for the presence of a tone, it is necessary to calculate the power in the filtered signal. This can be done using the following equation:

$$P^2 = Y_n^2 + Y_{n-1}^2 - 2cos(k)Y_nY_{n-1}$$

Because the coefficient 2cos(k) is the same as that precalculated for the filter loop, we can use it here to calculate the power:

```
power_squared = y[0] * y[0] +
                y[1] * y[1] -
                twocosk * y[0] * y[1];
```

After a sufficient number of sample periods, tone presence is shown by a squared power indication that will be at least one order of magnitude above the measurement in the absence of a tone. In tests, presence of a tone might demonstrate a squared amplitude of more than 1,000 with unity-signal amplitude at the input. An off-frequency tone, or a combination of off-frequency tones, would typically demonstrate a squared amplitude of less than 50.

Conclusion

As data-acquisition hardware and DSP-like building blocks become more common on inexpensive μCs, firmware engineers must find ways to pull DSP functions into the main CPU to reduce cost and improve performance. With the wealth of resources available in literature and on the web, however, this task is much easier than expected.

Design Challenges for Backlighting LCD TVs

Liquid crystal displays (LCDs) are becoming a very popular display technology for televisions (TVs). According to the research firm IDC™, the number of LCD TVs will grow to 50 million units in 2007 (source: IDC press release dated 1/22/04). LCD panels are actually electronically controlled light valves that require a backlight source to produce a visible image. LCD TVs typically use cold-cathode fluorescent lamps (CCFLs) for this purpose. (See the sidebar CCFLs, *page 14, for further information.) Other technologies, such as light-emitting diodes (LEDs), have been considered, but high cost has limited their adoption.*

This article focuses on the design challenges of driving and controlling multiple CCFLs for backlighting large LCD panels like those in LCD TVs.

Design Challenges

Because LCD TVs are consumer items, the overriding design factor is cost—yet a certain minimum level of performance must be maintained. The CCFL inverter must drive the lamps in a manner that will not dramatically shorten the lamp life. Also, because high voltages are generated in driving the lamps, safety must be considered. This article focuses on three key design challenges in driving multiple CCFLs in LCD-TV applications: picking the best drive architecture, driving multiple lamps, and tight control over the lamp- and burst-dimming frequencies.

Picking the Best Drive Architecture

Creating the high-voltage AC waveform needed to drive CCFLs can be realized through a number of architectures including Royer (self-oscillating), half bridge, full bridge, and push-pull. **Table 1** details the advantages and disadvantages of these four architectures.

Royer Architecture

The best application for Royer architecture (**Figure 1**) is in designs where the lamp frequency and brightness do not need to be tightly controlled. Because Royer architecture is a self-oscillating design containing variations in component values, it is difficult to control the exact lamp frequency and lamp current, both of which directly affect lamp brightness. For these reasons, Royer architecture is almost never used in LCD-TV applications, although its implementation is the least expensive of the four architectures presented here.

Full-Bridge Architecture

Full-bridge architecture is best suited for applications that require a very wide DC supply-voltage range (**Figure 2**). This is why almost all notebook PCs use a full-bridge approach. The inverter's DC supply is tied directly to the notebook main DC power source, which can vary from 7V (low battery) to 21V (AC charging). Some full-bridge implementations do require p-channel MOSFETs, which are more expensive than n-channel MOSFETs. Also, p-channel MOSFETs are less efficient due to their inherent higher on-resistance.

Half-Bridge Architecture

The biggest advantage of the half-bridge architecture is that it requires two fewer MOSFETs per drive channel than full-bridge architecture (**Figure 3**). It does, however, require a transformer with a higher turns ratio, thus increasing the cost of the transformer. Also, like full-bridge architecture, half-bridge architecture may require p-channel MOSFETs.

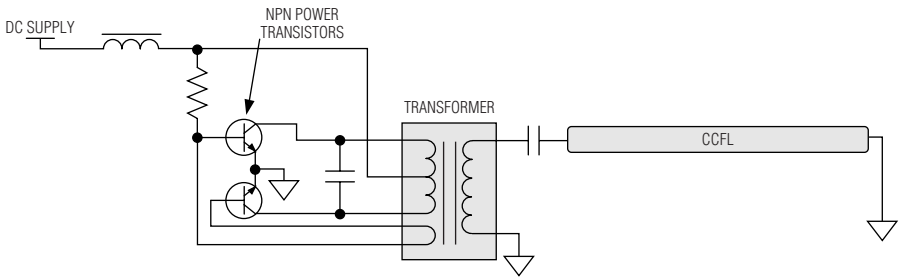


Figure 1. Royer drives are simple, but not very accurate.

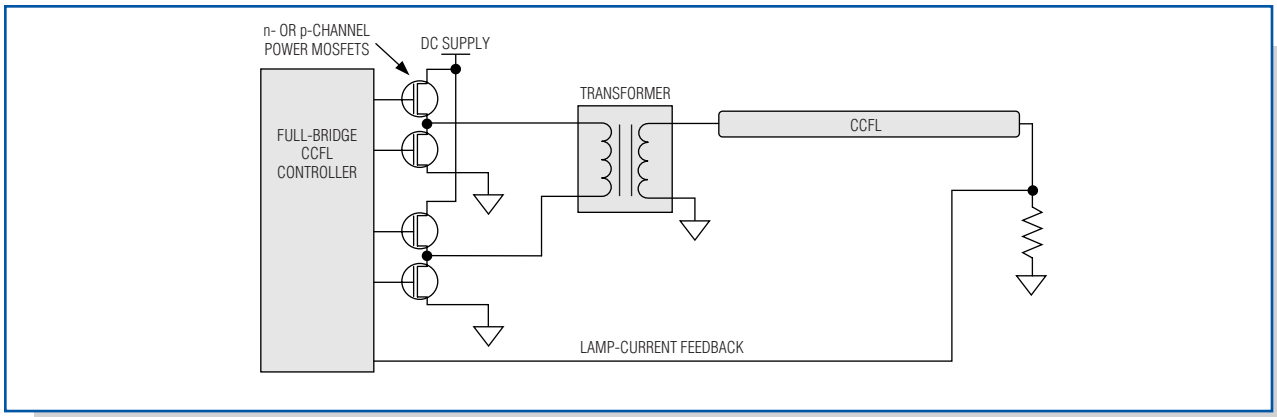


Figure 2. Full-bridge drives work well over a large DC-inverter supply range.

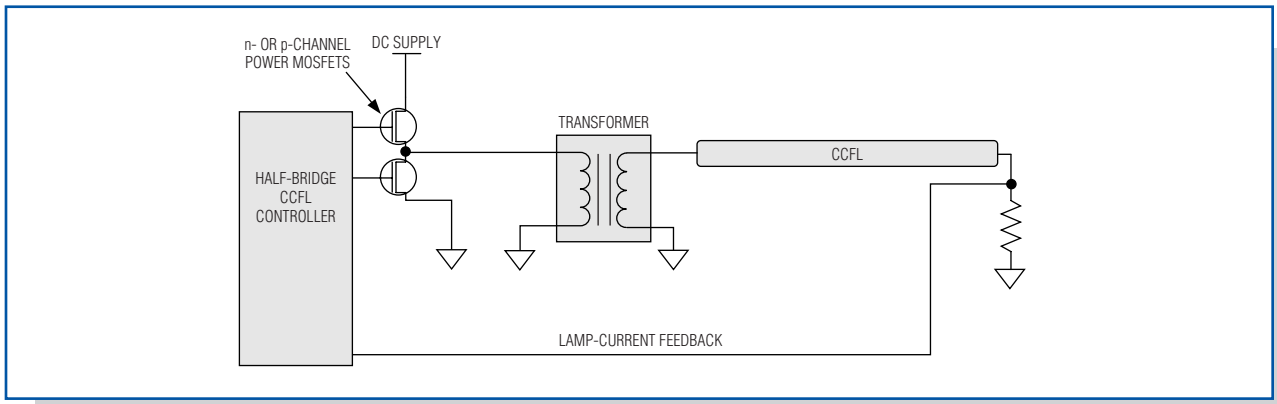


Figure 3. Half-bridge drives require two fewer MOSFETs than full-bridge drives.

Table 1. CCFL Drive Architecture Comparison

Drive Architecture	Advantages	Disadvantages
Royer	• Least expensive	• Cannot tightly control the lamp current or frequency • Requires tight DC-supply regulation • Requires a special transformer winding • Requires a ballast capacitor • Low efficiency
Full Bridge	• Does not require a center-tapped transformer • Works over a wide DC-supply range (greater than 3:1)	• Requires four MOSFETs • May require p-channel MOSFETs, which are higher cost and less efficient
Half Bridge	• Requires only two MOSFETs	• May require p-channel MOSFETs, which are higher cost and less efficient • Requires a higher turns ratio transformer, which increases cost
Push-Pull	• Requires only two n-channel MOSFETs, which are lower in cost and more highly efficient than p-channel MOSFETs • Easily scales to higher DC supply voltages (up to 120V) • Low transformer turns ratio	• Lower efficiency when the DC supply goes beyond a 2:1 range

Push-Pull Architecture

The final architecture to consider is a push-pull drive, which has a number of advantages. This architecture only uses n-channel MOSFETs (**Figure 4**), the use of which reduces cost and raises the efficiency of the inverter. It easily scales to higher DC-inverter supply voltages. Using a higher DC-inverter supply voltage only requires selecting a MOSFET with the appropriate drain-source breakdown voltage. The same CCFL controller can be used regardless of the DC-inverter supply voltage. This is not the case for full- and half-bridge architectures using n-channel MOSFETs.

The biggest disadvantage of push-pull architecture is that the DC-inverter supply range should be less than 2:1. Otherwise, the system efficiency will drop due to the AC waveforms' high crest factor when the DC-inverter supply voltage is high. This makes the push-pull architecture inappropriate for notebook PCs, but ideal for LCD TVs in which the DC-inverter supply voltage is regulated to within ±20%.

Driving Multiple Lamps

CCFLs have been used for a number of years in notebook PCs, digital cameras, navigation systems, and other equipment with smaller LCD screens. These types of equipment typically have only one CCFL, for which the traditional technique uses one CCFL controller. With the advent of large LCD panels where many CCFLs are needed, new approaches are necessary. One possible approach is to use a single-channel CCFL controller to operate multiple lamps (**Figure 5**). In this approach, the CCFL controller monitors the lamp current through only one of the lamps, and then drives all the lamps in parallel with approximately the same AC waveform. There are, however, several shortcomings with this approach.

The first problem is maintaining equal brightness across all the lamps so no bright or dull spots are apparent to the viewer. Driving all the lamps with the same waveform causes each lamp to have a different current driven through it, hence a different brightness. Also, using the same waveform can cause uneven brightness due to differences in the lamp impedances. Furthermore, the brightness of CCFL varies with temperature (**Figure 6**). Because heat

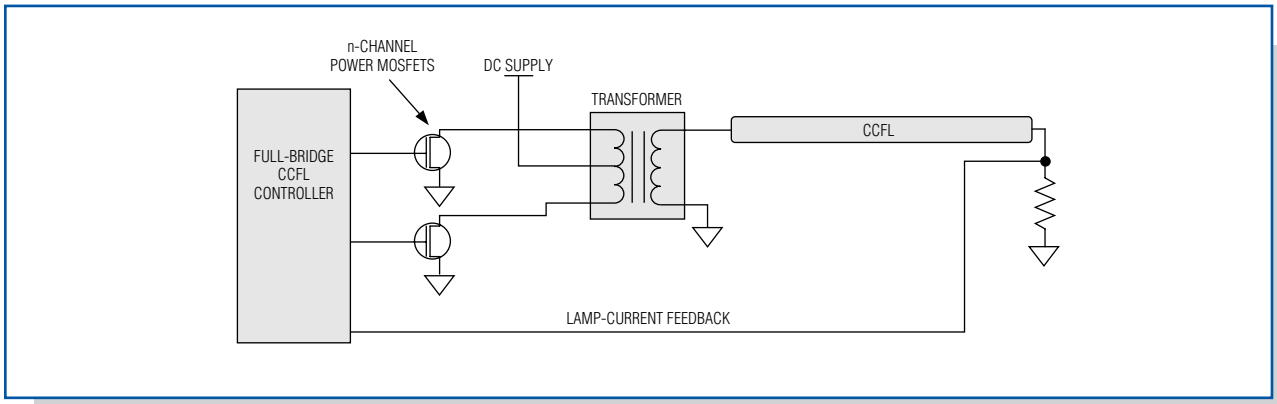


Figure 4. Push-pull drives are simple, yet provide accurate control.

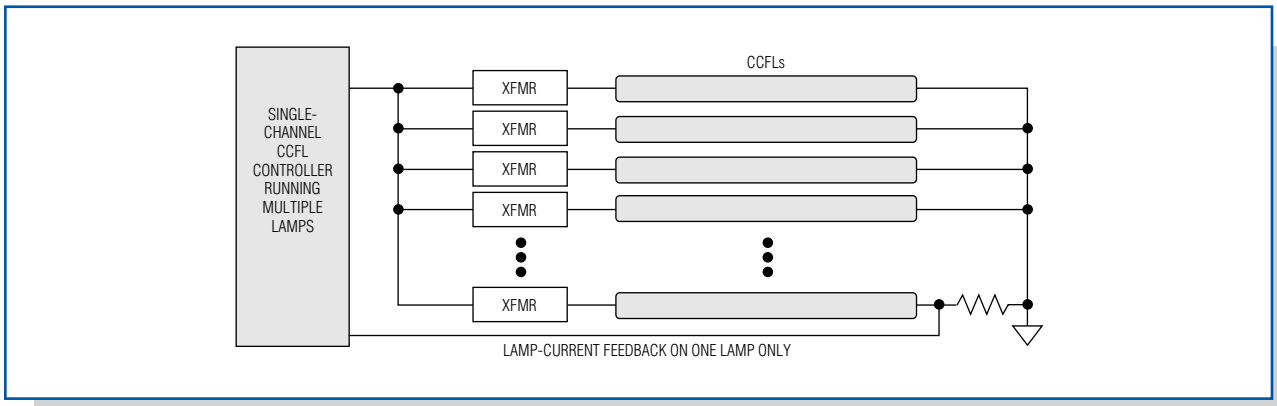


Figure 5. Controlling multiple lamps with a single-channel CCFL controller is not ideal due to the potential for uneven brightness, among other concerns.

rises, the lamps at the top of the panel (sidebar, **Figure 12**) will be hotter than the lamps at the bottom of the panel, again causing uneven brightness.

The second disadvantage to using a single-channel CCFL controller to drive multiple lamps is that a single lamp failure (like lamp breakage) will cause all the lamps to turn off. A third disadvantage is that, because all the lamps are driven in parallel and turned on and off simultaneously, the DC inverter supply must be more heavily decoupled with bulk capacitance. This raises the cost of the inverter.

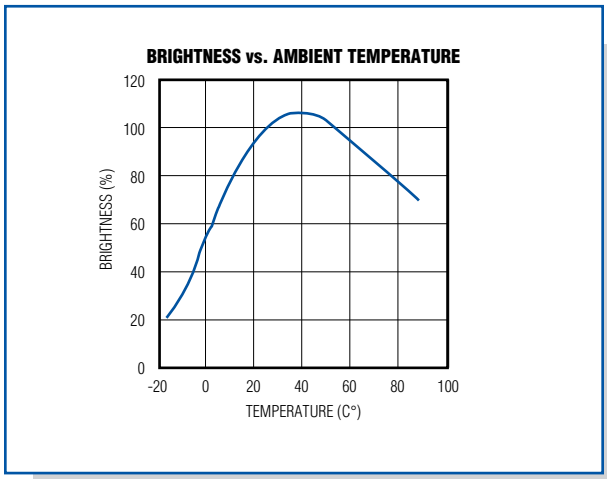


Figure 6. The brightness of CCFLs varies with ambient temperature.

One way to solve these several problems is to use a separate CCFL controller for each lamp (**Figure 7**). The major disadvantage to this approach, however, is the extra cost due to additional CCFL controllers.

The ideal solution for backlighting LCD panels is a multichannel CCFL controller that can drive and monitor each lamp independently (**Figure 8**). The multichannel CCFL controller thus resolves the issues of uneven brightness and single-lamp failure, reduces the required decoupling, and is cost-effective.

Tight Control over the Lamp and Burst-Dimming Frequencies

Because LCD-TV displays are dynamic and continually showing moving images, they have some additional requirements that do not exist in static display applications, like computer monitors and notebook PCs. The frequency at which a CCFL is driven can interfere with the image displayed on the LCD screen. Slow-moving lines or bars can be created if the lamp frequency is near a constant multiple of the video update rates. Eliminating these artifacts can be accomplished by tight control of the lamp frequency to within ±5%.

The same tight control is required for the burst-dimming frequency, used to adjust the brightness of the lamps. Typically a pulse-width modulated (PWM) signal in the 30Hz to 200Hz range turns the lamps off for a short period of time, though not long enough for them to lose ionization. If the burst-dimming frequency is near a multiple of the vertical sync rate, rolling lines can be created. Again, control of the burst-dimming frequency within a ±5% level eliminates this problem. Also, in some LCD-TV applications, it is necessary to synchronize the slow burst-dimming frequency of the CCFLs to the vertical sync rate to improve the image response of the LCD panel.

A Solution for the LCD TV Backlighting Challenges (DS3984/DS3988)

The DS3984 (four channel) and the DS3988 (eight channel) CCFL controllers solve all the design challenges raised in this article. These devices can be configured to drive one lamp per channel (**Figure 9**), or multiple lamps per channel (**Figure 10**), which allows the user to adapt the design to meet their cost vs. performance target. Multiple DS3984/DS3988s can be easily cascaded to support as many lamps as necessary for backlighting LCD TV panels.

The DS3984/DS3988 use a push-pull drive, which allows the use of lower cost, higher efficiency n-channel

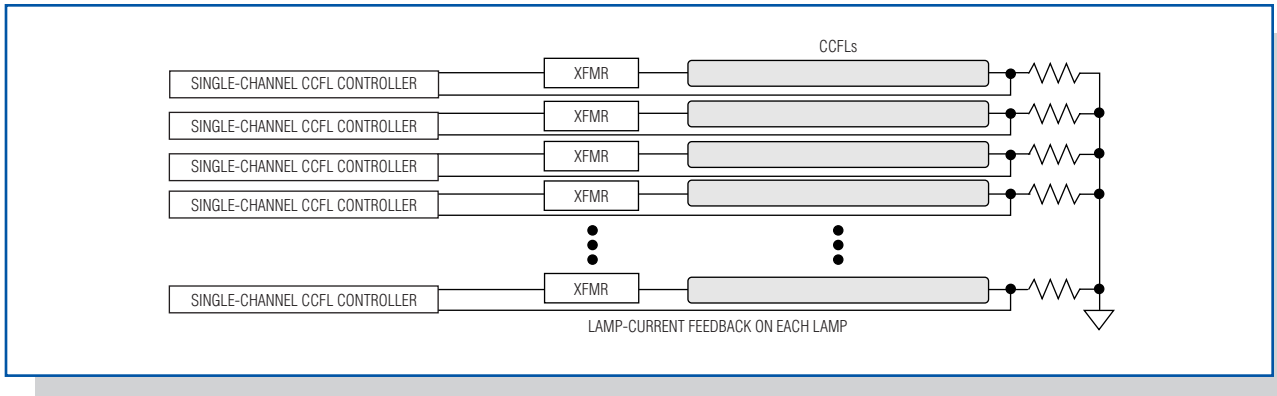


Figure 7. Using a single-channel controller for each CCFL is not cost effective.

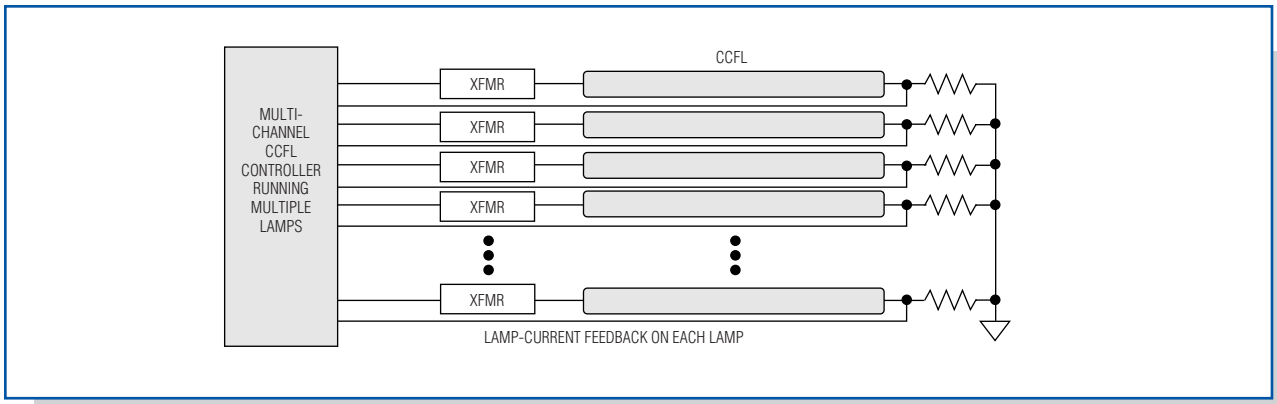


Figure 8. Controlling multiple lamps with a multichannel controller is the ideal approach.

MOSFETs. Also, the DC-inverter supply voltage can be easily scaled to a higher voltage. The individual lamp control and monitoring provides even lamp brightness, and reduces the overall component count in the inverter design. With individual lamp control, if a lamp fails, only the failing lamp is disabled. The other lamps continue to operate. Onboard lamp- and burst-dimming oscillators specified to a tight ±5% level eliminate visual artifacts, and can be synchronized to external clock sources.

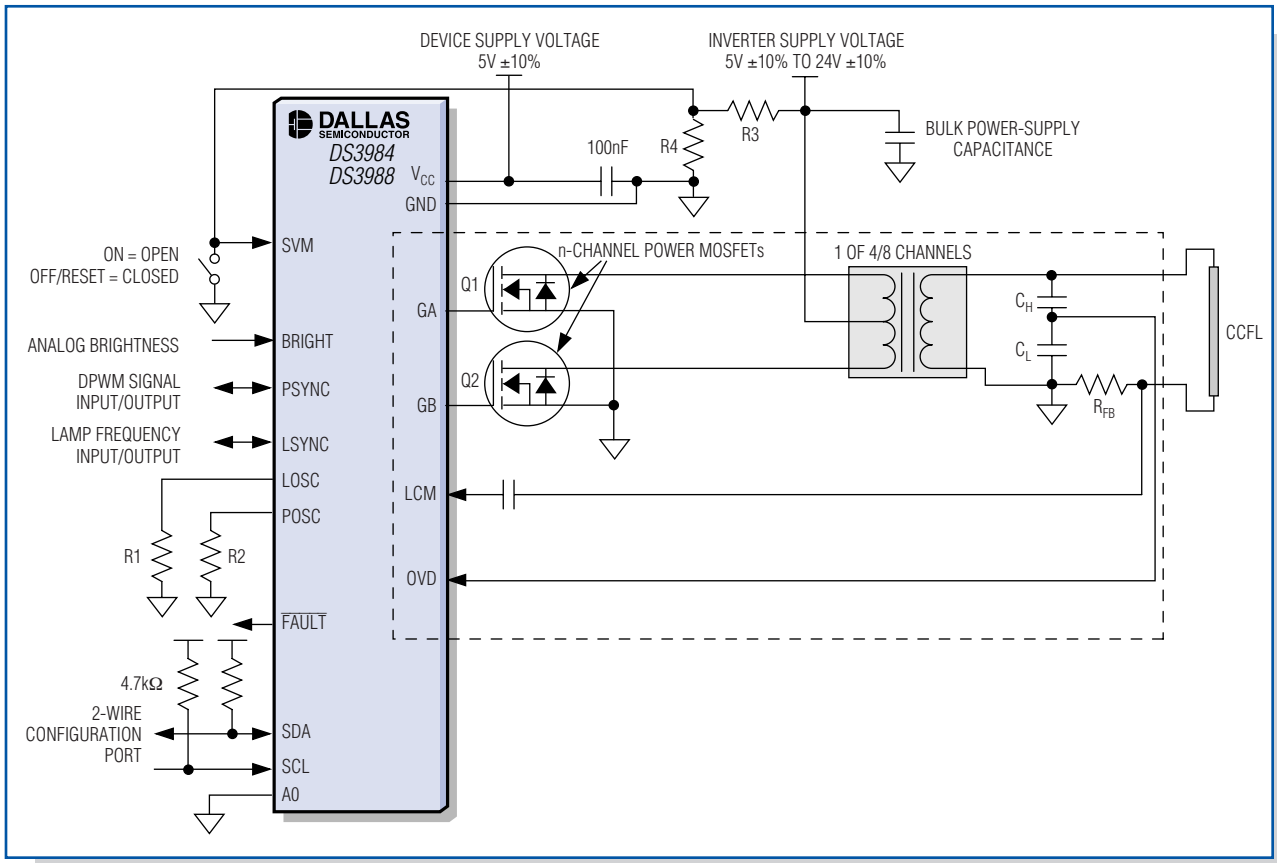


Figure 9. The DS3984/DS3988 drive and monitor each lamp independently for even brightness in LCD TVs and PC monitors.

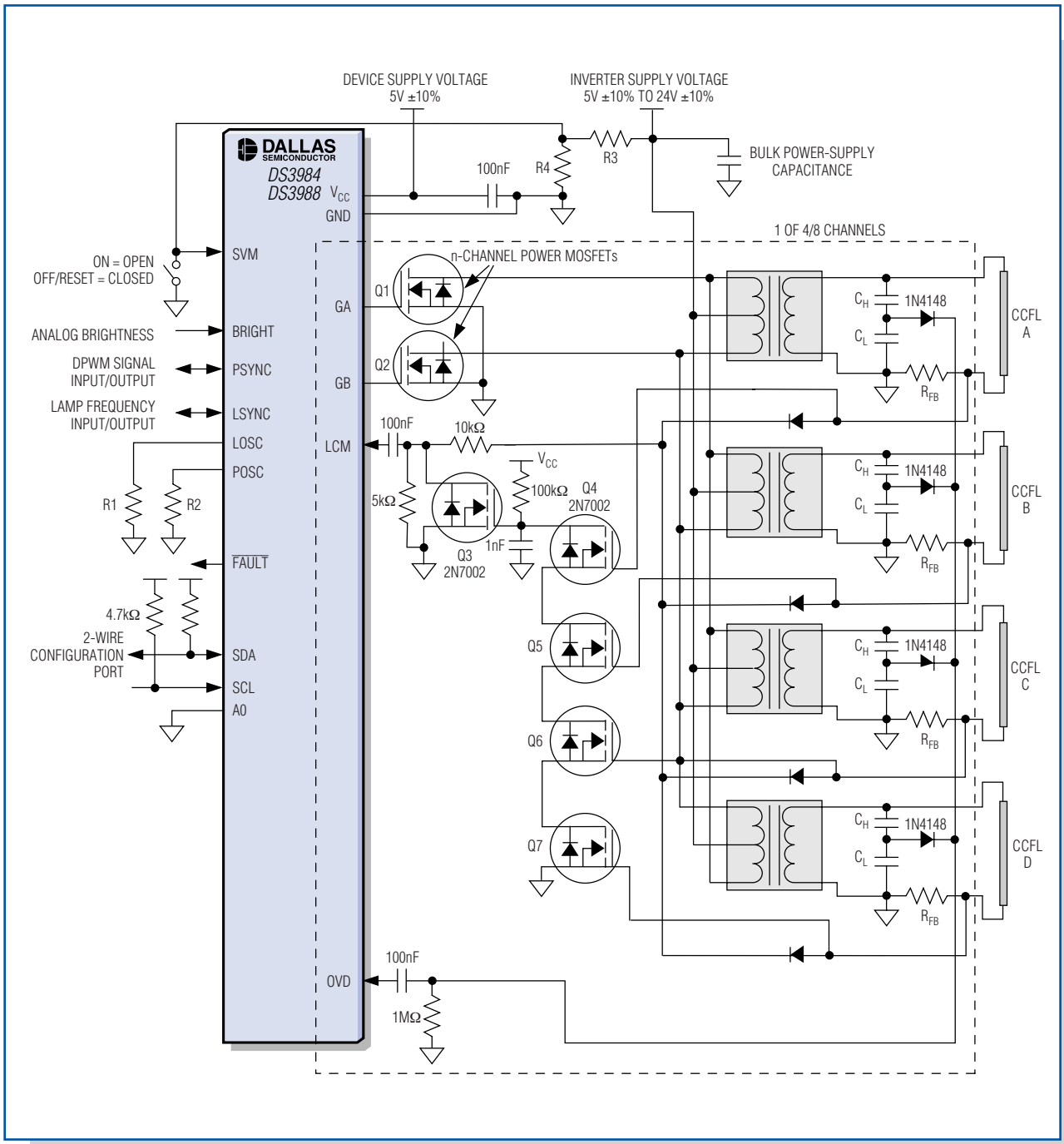


Figure 10. The DS3984/DS3988 can also drive multiple lamps per channel.

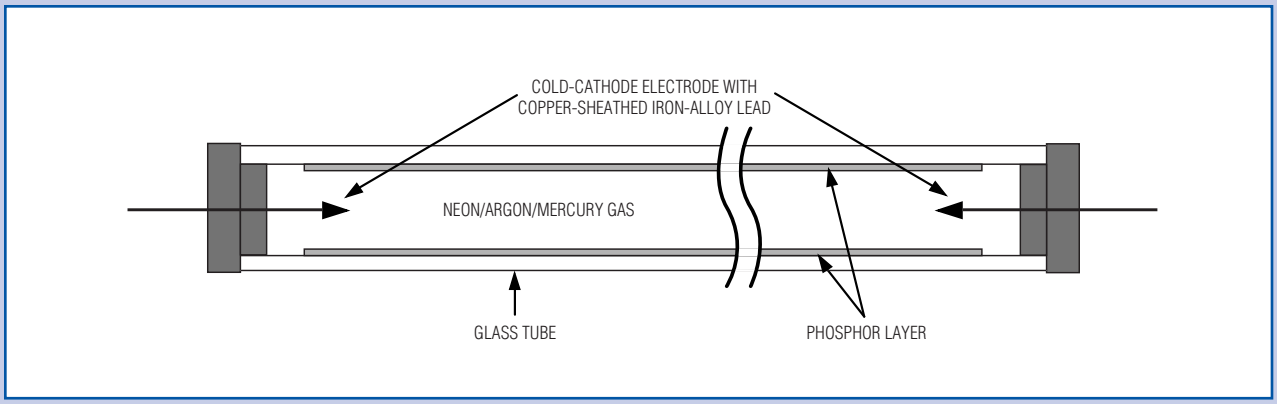


Figure 11. CCFLs are glass tubes filled with inert gasses.

CCFLs

Cold-cathode fluorescent lamps (CCFLs) are long, sealed glass tubes of small diameter filled with inert gases (**Figure 11**). When a high voltage is sent across the tube, the gases ionize, creating ultraviolet (UV) light. This, in turn, excites an inner coating of phosphor, creating visible light.

CCFLs have many desirable features, including:

- Excellent white light source
- Low cost
- High efficiency (electrical power in to light out)
- Long life (>25kilohours)
- Stable and predictable operation
- Easily varied brightness
- Light weight

CCFLs have some unique characteristics that must be accounted for in order to maximize their efficiency, life span, and usefulness. These characteristics, however, do cause design challenges. For example, to maximize the life of the lamps, CCFLs need to be driven with an AC waveform. Any DC component can cause some of the gasses to accumulate at one end of the tube and create an irreversible light gradient where one end of the tube is brighter than the other. Also, to maximize their efficiency (power in to light out), the lamps need to be driven with near-sinusoidal waveforms. To accomplish these tasks, CCFLs typically require a DC-to-AC inverter that transforms a DC-supply voltage to a 40kHz to 80kHz AC waveform, with an operating voltage of $500V_{RMS}$ to $1000V_{RMS}$.

Typical CCFL Arrangement in LCD TVs

Figure 12 shows how CCFLs are typically arranged in an LCD TV. The 12 lamps in this TV are equally spaced across the LCD backplane to provide the best possible light distribution. It is important that all the lamps operate at the same brightness level. Although there is a light diffuser placed between the CCFL lamps and the LCD panel to help evenly distribute the backlighting, uneven lamp brightness can still be visible and detract from the quality of the TV image. Depending on the size of the LCD panel, up to 30 or even 40 CCFL lamps may be required.

Figure 12. LCD TVs contain from 4 to 40 CCFLs.

Feature-Rich, Complete Audio Record/Playback for GSM/GPRS Cell Phones

Short design cycles, coupled with PC-board area and cost pressures, continue to compel higher levels of integration in cell phone circuitry. Basic (low-tier) phone functionality is headed inexorably towards a one-chip solution. Meanwhile, in high- and mid-tier market segments in which model differentiation is critical, demand for high-performance, feature-rich peripheral components become increasingly important. Given that market push and the continually evolving feature set, a complete analog- and digital-audio solution optimized for GSM/GPRS cell phones would provide a solid core for ongoing designs. The optimized solution must also include voice-band audio functionality (microphone, receiver speaker amplifier, ADC, and DAC) and flexible multimedia features (high-resolution ADC and DAC, voice recording, stereo microphones, headphones, and 8Ω speaker amplifiers). With that combination of features, the integration of both cell phone and application-based audio functions would be seamless.

Introduction

The complexity and high circuit density of cell phones challenge the system designer wishing to establish a high-quality, audio record/replay path that meets provider specifications. New models with additional multimedia features, such as a still camera, ring-tone generators, MP3 replay, and voice memo, usually require incremental product changes. This necessitates not only new components, but also PC-board layout changes that can give rise to nonideal grounding and subsequent new noise problems.

Noise and interference issues in a cell phone analog-audio signal path can usually be attributed to demodulation of RF into the audio band or shared/nonideal grounding.

When subjected to high levels of RF energy from the phone antenna, the comparatively low-bandwidth audio circuitry in a phone can unintentionally demodulate the complex RF-transmitted signal. This degrades the noise floor of the audio path. While certain techniques and topologies can be applied to minimize this degradation within the audio amplifier circuitry, suppression components placed adjacent to the input pins are a cost-effective cure. Low-value capacitors-to-GND are often used, due to the designer selecting the minima of the capacitor's impedance to correspond with the carrier frequency of the radio.

One effective audio solution that minimizes shared/nonideal grounding integrates all the typically required analog-audio I/O functions into a single IC. This design transfers most of the problematic grounding issues from the PC-board layout engineer to the IC manufacturer. Besides including the necessary analog-audio I/O functions, that same IC must provide digital-audio interfacing sufficient to support voice band and any multimedia (i.e., application processor) functions. The IC also must provide partitioned shutdown control over the various blocks to maximize battery life.

The following discourse highlights some of the analog-/digital-audio issues that arise in single-chip implementation. The MAX9851* is used as an example of techniques and features that simplify GSM/GPRS cell phone design.

Analog Audio—Minimizing Microphone Noise

High-gain audio circuits, such as microphone amplifiers (mic amps), are subject to degradation from poor grounding. This is particularly true for single-ended circuit topologies, in which small voltage differences between the mic-amp ground reference and the source ground reference (in this case, the GND pin of the mic capsule) are amplified into the signal path. In a complex product, like a cell phone, where audio ground planes are often shared with other circuits, degradation can be problematic, because the copper plane is not "zero ohms" (as is frequently assumed). Consequently, any current flowing through this finite resistance can cause small potential differences across the plane.

The grounding problem can be addressed with a mic amp that features a fully differential input. This approach, which allows remote sensing of the mic's GND pin, is incorporated into the MAX9851. Remote sensing forces any AC-voltage differences between the CODEC reference and the mic GND to appear as a common-mode signal to the mic amp. These differences are then reduced by the common-mode rejection ratio (CMRR) of the amp, thereby significantly attenuating the effective noise contributed to the signal path. The only penalties with this design are an extra PC-board trace to the mic from the CODEC and an extra coupling capacitor.

The MAX9851 also allows stereo, external mic inputs to be switched in to override the internal mic. These inputs typically would be sourced from a car kit or other external headset. In this case, using the amp input's CMRR, the EXTMICGND pin acts as a 'Kelvin sense' for both L and R channels, canceling ground differential noise in the same way as described above. The EXTMICGND PC-board trace should be extended to the GND connector of the car kit jack or headset connector for best results (**Figure 1**).

*Future product—contact factory for availability.

Mic bias circuits can introduce significant noise in the signal path. A large percentage of any bias output-voltage noise appears directly across the mic-amp input. More proficiently designed mic amps, as in the MAX9851, provide a regulated, low-noise bias voltage with output-noise levels matched to the internal mic-amp input noise.

Analog Audio—Stereo DirectDrive™ Headphone and Receiver Outputs

The ability to replay compressed music files at near-CD quality places high demands on headphone audio reproduction. Signal-to-noise ratio (SNR), linearity, and bandwidth must be improved over the basic 300Hz to 4kHz voice-path requirement. Low-frequency extension can be problematic, as headphone drivers typically need a series capacitor to prevent the headphone amp's DC bias from appearing across the headphone transducer. The typical impedance range of common stereo headphones extends down to 16Ω, so that any series capacitor forms a highpass filter to low-frequency content. For extension of the listening response down to 100Hz, for example, two 100µF DC-blocking capacitors would be needed to guarantee 16Ω stereo headphone operation.

Use of Maxim's DirectDrive technique allows headphone operation without series capacitors, due to the amp outputs being referenced to 0V. The low-frequency content is then limited by either a DC-removal filter (digital source), as designed into the MAX9851, or by the input-coupling capacitors on the line or mic inputs (analog source). A further advantage of the DirectDrive design is the inherently low click/pop levels when bringing the device into or out of shutdown. As there are no series capacitors to charge or discharge, no net turn on/off current flows through the headphones.

The stereo headphones outputs of the MAX9851 are also capable of bridged-mono operation, (**Figure 2**) which enables compatibility for different headsets and accessories. The same socket could accommodate stereo headphones or a mono (mic plus hook switch and speaker) headset. The outputs remain ground referenced in this mode, so no DC voltage appears on the headset cable. Therefore, fault or short-circuit conditions are less problematic.

The receiver speaker output also uses the on-board charge pump in the DirectDrive design, so the output is single ended and the negative speaker connection is GND (0V). The output still has nearly the same voltage swing of a more typical BTL (differential) output, because the inverting charge pump provides a negative rail almost equal to that of the applied AV_{DD}. The resulting peak-to-peak output across the receiver speaker is almost 2 x AV_{DD}.

Analog Audio—Class D Speaker Amplifiers

The MAX9851 incorporates Maxim's third-generation class D technology to drive 8Ω (or 4Ω) speakers. The main advantage of class D (switching) amps over class AB (linear) amps is efficiency. Class AB amps dissipate significant power in the output devices unless the amp is driven into clipping. However, because class D topologies have their output devices either on or off, their thermal dissipation is less and battery life can be extended. Extended battery life can be significant if a cell phone is used in speakerphone mode frequently, or supports push-to-talk (PTT) operation.

There are caveats to using class D topologies, however, especially in a product whose core function is RF transmission/reception, such as a cell phone. The fast-switching edges associated with efficient class D amp operation can lead to RF-emission problems, especially with long PC-board traces and speaker leads. To counter the RF-emission problem, the MAX9851 stereo, class D speaker amps incorporate a proprietary EMI-reduction topology (active emission limiting) that suppresses high-frequency RF emissions from speaker leads/traces at the expense of slight efficiency degradation. State-of-the-art IC construction techniques also minimize any interaction between the class D switching output stages and any of the sensitive, low-noise analog circuitry on the CODEC.

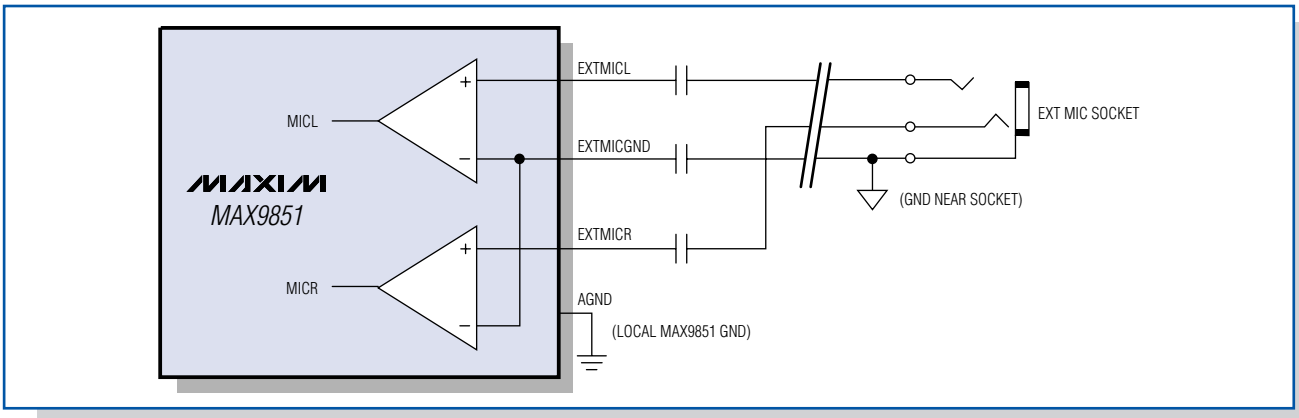


Figure 1. The use of differential input amplifiers allows remote sensing of the socket "GND" reference. Any AC voltages between local and socket grounds are largely rejected, and not amplified by the mic amp gain.

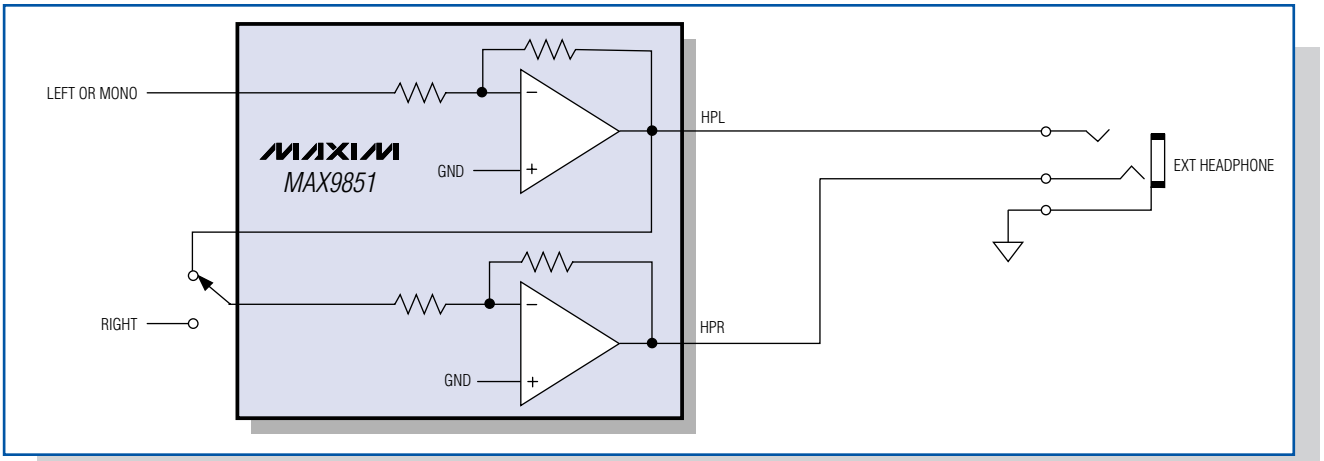


Figure 2. The DirectDrive headphone outputs are capable of bridged mono and stereo operation. The Maxim proprietary GND-referenced output means no series coupling capacitors are necessary, saving cost and PC-board area.

With the ability to connect to an unregulated, single-cell Li+ battery, the stereo amps are capable of 1W output from a 4.2V supply into an 8Ω speaker (**Figure 3**). More power is available if lower impedance speakers are used, but 4Ω speakers are not generally found in the smaller diameter drivers commonly used in cell phone designs.

Digital Audio—General Architecture, Signal Flow

To allow conversations, the basic functionality of the GSM/GPRS cell phone must be supported by an 8kHz (or optional 16kHz) sampling ADC/DAC path, with a 16-bit depth in either direction. In the MAX9851, this I/O function is fully synchronous to the 13MHz (or 26MHz for f_S = 16kHz) MCLK input to ensure no dropped or repeated samples. The S1 digital I/O lines in GSM voice mode (**Figure 4**) are used to access this basic functionality; the S1 digital interface can operate in MASTER or SLAVE mode.

Many mid- and high-tier phones are commonly required to provide additional DAC functions at higher bit depths and sample rates. Examples of these functions are WMA/MP3 replay or generating WAV file ring tones. Combining digital-to-analog conversion for these functions with existing voice converters allows high integration, and gives one 'point source' for all data conversion. These features can be useful in product design, in which ground loops and audio level differences can prove problematic when trying to sum the two functions in the analog domain.

Therefore, having one converter to combine voice and multimedia data seems an ideal solution. The main difficulty with this approach is that any voice conversion must remain synchronous to the GSM/GPRS rate dictated by the MCLK input. The multimedia replay, moreover, often demands an unrelated sample rate: 44.1kHz or 48kHz, for example. The MAX9851 solves these challenges by implementing an algorithm similar to sample-rate conversion (SRC) on the digital-input data, thus allowing a single DAC to transform the summed voice and multimedia data in a synchronous manner.

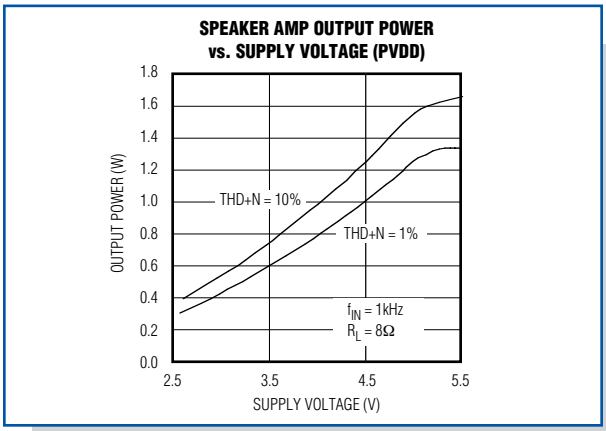


Figure 3. The MAX9851 stereo class D speaker amplifier can operate from a raw battery supply, delivering 1W continuous (at 10% THD+N, 1kHz signal) from a 4.2V supply.

In SLAVE mode, the incoming sample rate for the GSM voice data is necessarily sample-rate accurate (as dictated by MCLK). However, an internal digital PLL locks to the incoming LRCLK on the S2 digital input, allowing a precise (averaged over many samples) replication of nonsynchronous multimedia audio data. In MASTER mode, again the voice data is correctly aligned to the desired integer division of MCLK, but the S2 LRCLK data rate is approximated with a slight f_S error, which is usually insignificant. Sample rates from 8kHz to 48kHz are supported on either S1 or S2 inputs.

The MAX9851 S2 digital I/Os have an interface that supports I²S and popular minor variations thereof. When not operating in GSM voice mode, the S1 interface can be programmed to support I²S, maximizing the interface flexibility often needed in feature-rich, high-end phones.

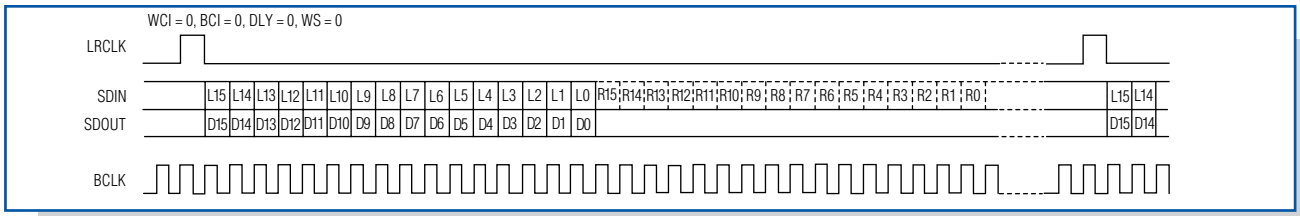


Figure 4. Basic GSM speech conversion functionality is supported by a generic, GSM voice-mode protocol on the S1 outputs. This can be operated as a MASTER or SLAVE; SLAVE mode expects both BCLK and LRCLK to be provided by the host.

Digital Audio—GSM Filters

As can be seen in **Figure 5**, the S1 digital I/Os have extra filters that can be enabled in GSM voice mode. These digital blocks are an efficient implementation of tightly specified lowpass and highpass filters. This implementation suppresses energy near the Nyquist band edge and at low frequencies. The filters can prove beneficial to meet noise and signal-leakage envelopes when a phone undergoes testing and qualification. **Figure 6** gives the frequency response of enabled filters.

Summary

The previous examples highlight only a few issues that cell phone system designers/architects must address. Design cycles are notoriously short for this end application and feature sets continue to mature and change, almost on a per-model basis. Therefore, investing time in a well-engineered, flexible, comprehensive core chipset architecture is a worthwhile decision.

Controlling low-noise analog circuitry, which interfaces with multiple replay/record systems at different sample rates, is only part of the overall design task. It is also important to integrate the following features in one solution:

- Analog functionality and high performance
- A one-point, digital-/analog-audio interface
- Digital interfacing flexibility
- Comprehensive power management and partitioned shutdown

These features address a significant number of system-design, architecture, and topology issues. The MAX9851 is a 48-pin, 7mm x 7mm single-chip solution that resolves these issues, and forms the basis of either mid- or high-end GSM/GPRS cell phone audio designs.

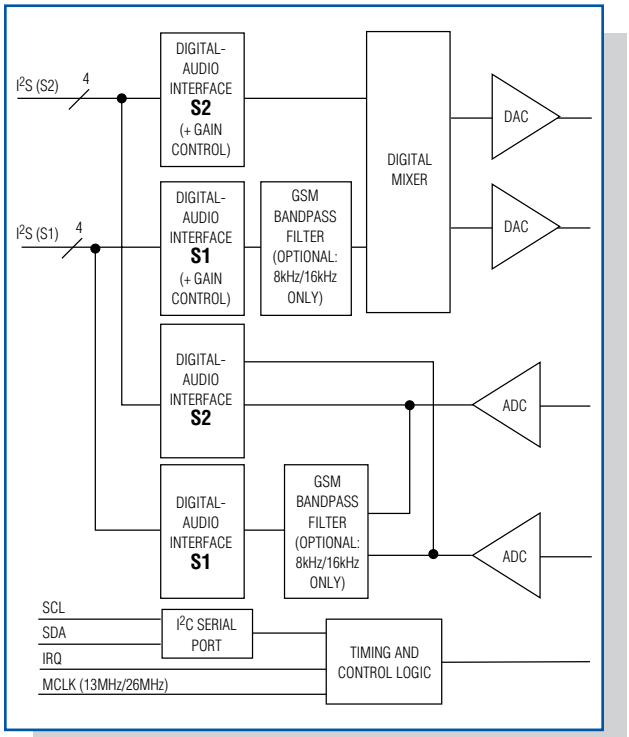


Figure 5. The MAX9851 integrates two independent sets (S1 and S2) of digital-audio interface I/Os. For DAC replay, each interface can run at differing and noninteger-related sample rates in either MASTER or SLAVE modes.

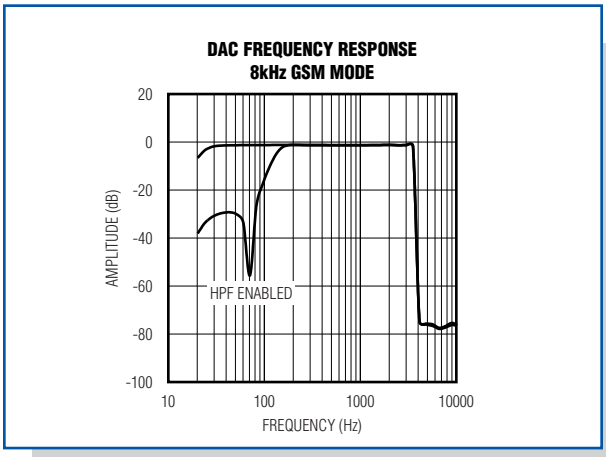


Figure 6. The frequency response of the GSM replay path has enabled GSM filters. At f_S = 8kHz, note the steep roll-off just before the Nyquist frequency (4kHz). The highpass filter (HPF) can be optionally defeated.