

Edited by Bill Travis

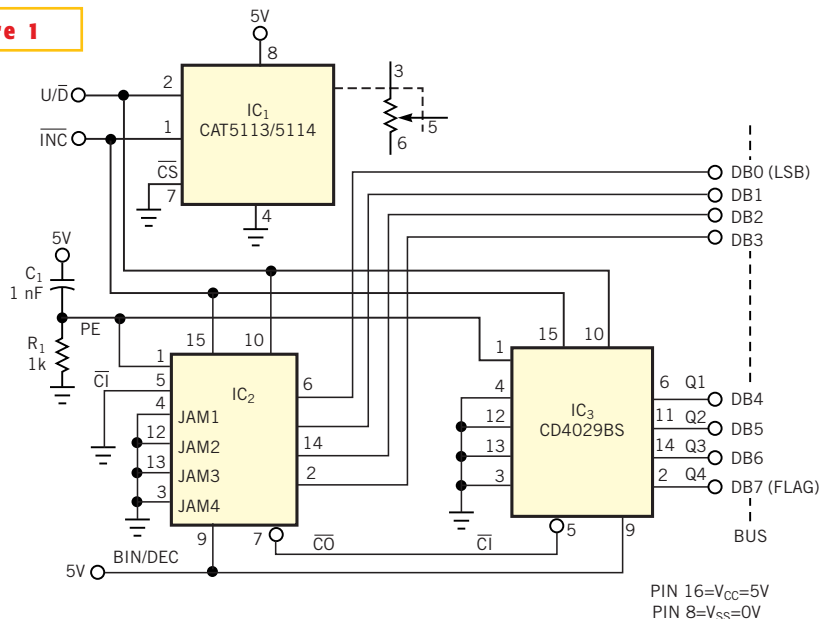
Where is the wiper?

Chuck Wojslaw and Dave Gillooly, Catalyst Semiconductor, Sunnyvale, CA

YOU ENJOY SIGNIFICANT advantages when using DPPs (digitally programmable potentiometers) with increment/decrement serial interfaces. Programming the serial interface is simple and fast, and you can adjust the potentiometer in real-time applications. The interface, however, provides no information about wiper position, and this information is important in some applications. If, for example, you use the potentiometer to control a parameter in a closed-loop, real-time application, the data reflecting the final wiper settings can be valuable in evaluating both the product performance and the circuit design. The circuit in **Figure 1** keeps a digital record of the DPP's wiper position by using two presettable CD4029 up/down counters, IC₂ and IC₃. The counters monitor the control signals $\overline{\text{INC}}$ and $\text{U}/\overline{\text{D}}$ of the DPP, IC₁.

During power-up, the wiper assumes position $(00)_{10}$, which it takes from previously programmed nonvolatile memory. Also during power-up, R₁ and C₁ differentiate the 5V power supply; this

Figure 1



Two up/down counters keep track of a digitally programmable potentiometer's wiper position.

differentiated signal serves to preset the binary counters to $(0000\ 0000)_2$. Thus, the DPP and the external IC₂/IC₃ counters are at the same point after power-up. The level-sensitive up/down signal establishes the direction of movement of the DPP's wiper and the direction of the count. The edge-sensitive $\overline{\text{INC}}$ signal advances both the wiper and the counter. The $\overline{\text{INC}}$ pin of the DPP responds to negative-edge triggering, and the clock input of the counter responds to positive-edge triggering. If the signal driving the $\overline{\text{INC}}$ line is a pulse (a common occurrence), the two inputs are compatible.

The Q outputs of the counters (DB0 to DB7) indicate in binary notation the location of the wiper. You can use the same circuit, using two counters, for

DPPs having as many as 256 taps. The DPP does not "wrap around" when the wiper advances to its upper or lower limit. The counters, however, do wrap around. To identify the case in which the digital counter is not in synch with the DPP, you can use the MSBs of the counters as flags. DB7 can serve as a flag for the 32-tap CAT5114 and the 100-tap CAT5113. You can change the initial count during power-up to something other than zero by preprogramming the DPP and setting high and low levels on the JAM inputs of the digital counters to the desired value.

Is this the best Design Idea in this issue? Select at www.edn.com.

Where is the wiper?	107
PLD code creates PWM generators.....	108
Square-wave modulator has variable frequency and pulse width.....	110
Expanded-scale indicator revisited	112
Butterworth filter has adjustable group delay	114
Single transistor sequences multiple supplies.....	116

Publish your Design Idea in EDN. See the What's Up section at www.edn.com.

PLD code creates PWM generators

Clive Bolton, Bolton Engineering Inc, Melrose, MA

THE PLD (programmable-logic-device) code in Listing 1 creates arbitrary-resolution, pulse-width-modulated (PWM) generators. PWM generators are useful as low-bandwidth D/A converters in hardware of micro-processor-based systems. When you pass it through a simple RC lowpass filter, a PWM waveform becomes a voltage that's approximately equal to the PWM duty cycle times the supply voltage. In practical systems, the driving hardware is imperfect, so the minimum value is never zero, and the maximum value never equals the positive-voltage rail.

The software module in Listing 1 automatically generates the required hardware from two compile-time parameters: PWM_WIDTH and AVALUE. PWM_WIDTH sets the number of possible steps in the PWM comparison. For example, 6 bits yields 2^6 , or 64, steps. AVALUE sets the value at which the PWM initializes upon power-up or reset (set to one-half scale in the example in Listing 1).

The module has two major sections: a holding register, which stores the PWM value, and a counter, which generates the PWM waveform. You can update the holding register independently of the PWM counter. The holding register's value automatically strobes into the PWM counter when the counter overflows. The module takes the CLOCK, ACLR, ENABLE, WRITE, and DATA[PWM_WIDTH-1..0] inputs. CLOCK is the master system clock; all signals other than ACLR must be synchronous with the clock's rising edge. ACLR initializes the hardware to the power-up state and loads AVALUE into the holding register. When ENABLE=0, the PWM output becomes 0 (off); when ENABLE=1, the PWM generator produces the PWM waveform at the Q output. Asserting WRITE for one clock cycle strobes the data presented on DATA[PWM_WIDTH-1..0] into the holding register. The PLD code uses Altera's (www.altera.com) AHDL high-level design language; you can directly compile the code

LISTING 1—AHDL CODE FOR PWM GENERATOR

```

INCLUDE "LPM_COUNTER.INC";
INCLUDE "LPM_COMPARE.INC";
INCLUDE "LPM_FF.INC";

PARAMETERS
(
    PWM_WIDTH      = 6,      -- Bits (set to 6 for testing)
    AVALUE         = B"100000" -- Async reset value
);

SUBDESIGN pwm
(
    clock          : INPUT;
    aclr           : INPUT;
    enable         : INPUT = VCC;      -- zeros
    PWM output     : INPUT;           -- writes into
    holding register
    data[PWM_WIDTH-1..0] : INPUT;
    Q              : OUTPUT;
    -- period_pulse : OUTPUT;       -- for debug and ext
    sync           : OUTPUT;
    -- cout        : OUTPUT;
)
VARIABLE
    pwm          : LPM_COUNTER WITH (LPM_WIDTH =
    PWM_WIDTH,
    LPM_DIRECTION
    = "DOWN");
    cnt          : LPM_COUNTER WITH (LPM_WIDTH = PWM_WIDTH,
    LPM_DIRECTION
    = "DOWN");
    pwm_ff       : SRFF;
    pwm_reg      : LPM_FF WITH (LPM_WIDTH = PWM_WIDTH,
    LPM_AVALUE = AVALUE);
    pwm_preout   : NODE;
    period_pulse : NODE;           -- for debug and ext sync
    cout        : NODE;

BEGIN

    ASSERT
    REPORT "PWM_WIDTH: %" PWM_WIDTH
    SEVERITY INFO;

    % PWM Holding Register %
    pwm_reg.clock = clock;
    pwm_reg.aset = aclr;
    pwm_reg.enable = write;
    pwm_reg.data[] = data[];

    % PWM Counter %
    cntr.clock = clock;
    cntr.aclr = aclr;
    cntr.cnt_en = enable;
    period_pulse = DFF(cnt.cout, clock, !aclr, VCC);

    % PWM Counter %
    pwm.clock = clock;
    pwm.aclr = aclr;
    pwm.sload = cnt.cout;
    pwm.data[] = pwm_reg.q[];
    pwm.cnt_en = enable;
    cout = pwm.cout;

    % PWM Output F/F %
    pwm_ff.clk = clock;
    pwm_ff.clrn = !aclr;
    pwm_ff.s = !cout AND period_pulse;  -- turn FF on at beginning of
    interval
    pwm_ff.r = cout OR !enable;  -- turn off at carry overflow.
    pwm_preout = pwm_ff.q;

    Q = DFF(enable AND pwm_preout,
    clock, !aclr, VCC);

END;

```

into any of Altera's PLDs. Using an EP1K10TC100-3 PLD, a design with parameters set to the default values in **Listing 1** operates as fast as 139 MHz. Although we wrote the code for Altera's

devices, you can readily translate the design structure and flow into VHDL or Verilog. You can download **Listing 1** from the Web version of this Design Idea at www.edn.com.

Is this the best Design Idea in this issue? Select at www.edn.com.

Square-wave modulator has variable frequency and pulse width

Michael Fisch, Agere Systems, Longmont, CO

A FEW YEARS AGO, I worked at a disk-drive company. We had a plating facility that required square waves to drive the high-voltage plating operation. The challenge was that the square wave's pulse width had to be variable, along with the duty cycle. Also, the amplitude of the pulses had to be adjustable. The circuit in **Figure 1** satisfies all these criteria. The circuit delivers a unipolar (adjustable from 0 to 12V) pulse with adjustable fre-

quency and pulse width. The first half of a dual, retriggerable monostable multivibrator, IC_{1A}, generates the frequency of the pulse train. The 100-k Ω potentiometer, R₁, along with R₂ and C₁, sets the adjustable frequency. R₃, R₄, and C₂ set the adjustable pulse width in the second section of the multivibrator, IC_{1B}. The ac-coupled op amp, IC_{2A}, running open-loop, delivers a $\pm 12V$ pulse output. D₁ and D₂ clamp the negative-going excursions of the pulse train to ground. The other half of the op amp, IC_{2B}, serves as a level shifter that allows amplitude control over the range 0 to 12V. You can modulate the amplitude at low frequency by varying the amplitude-control voltage.

sions of the pulse train to ground. The other half of the op amp, IC_{2B}, serves as a level shifter that allows amplitude control over the range 0 to 12V. You can modulate the amplitude at low frequency by varying the amplitude-control voltage.

Is this the best Design Idea in this issue? Select at www.edn.com.

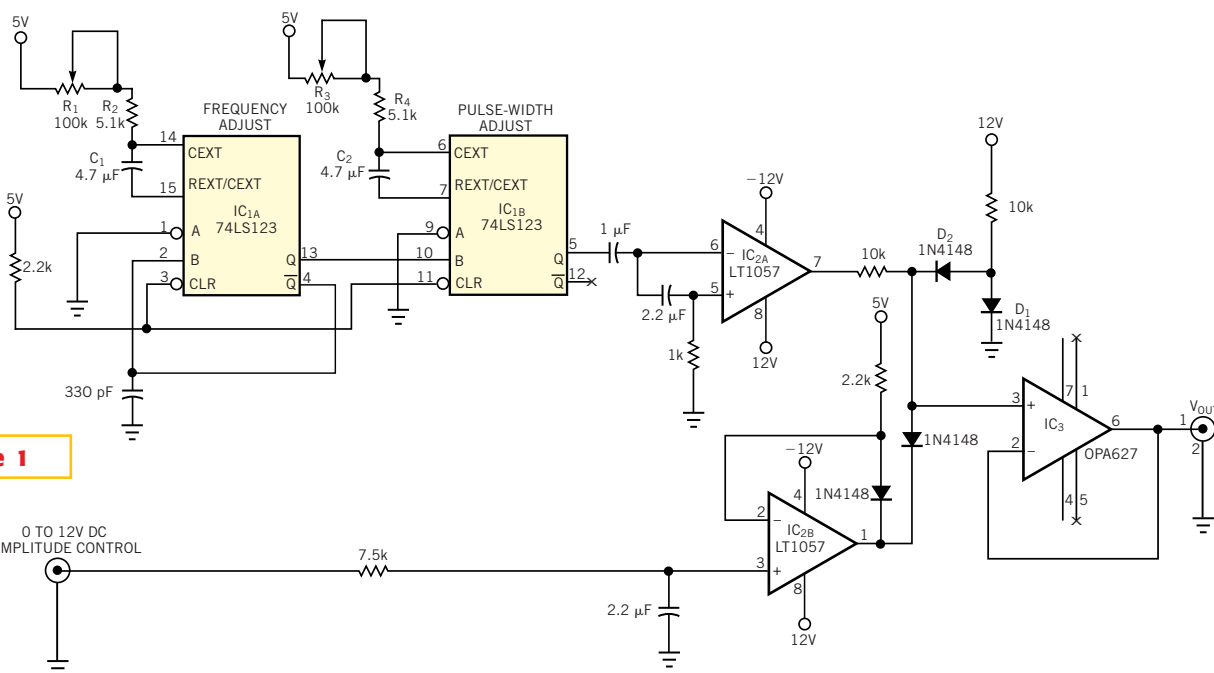


Figure 1

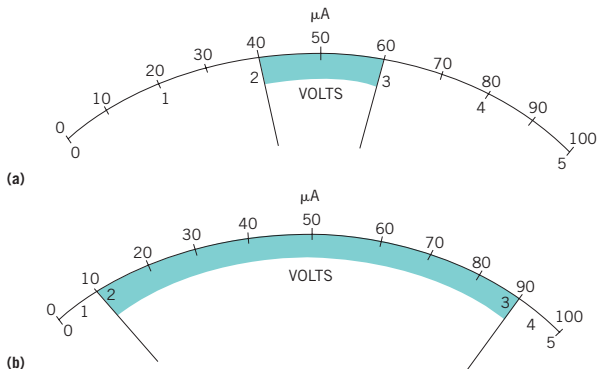
This variable-frequency circuit allows amplitude modulation of its pulse-train output.

Expanded-scale indicator revisited

Abel Raynus, Armatron International Inc, Melrose, MA

THE VISUALIZATION AID that a previous Design Idea describes allows only the expansion of the upper end of the scale (Reference 1). But what can you do if, according to your project requirements, you need to expand the middle region of the scale? **Figure 1a** illustrates the challenge. A voltmeter comprises a 100- μ A dc meter and a series resistor. The voltage under test, V_{TEST} , ranges from 0 to 5V. The voltage changes between 2 and 3V (the “green zone”) are of interest. But at the same time, you cannot ignore the voltages from 0 to 2V and from 3 to 5V, and you need to be able to observe these voltages. With a linear scale, the green zone consumes only 20% of the full-scale range. Your objective is to expand this zone to 80%, leaving 10% at the lower end and 10% at the upper end of the scale (**Figure 1b**). The circuit in **Figure 2** solves the problem. The window comparator, IC₁, controls the variable impedance of the voltmeter. Analog switches S₁ and S₂ provide a contact-logic AND function and put resistor R₂ in parallel with R₁ only upon closure of both switches. This closure occurs when V_{TEST} is between the threshold voltages V_{T1} and V_{T2} (**Figure 3**). You can calculate the re-

Figure 1



In a, the 2 to 3V “green zone” occupies only 20% of the scale; in b, this zone expands to 80%.

sistor values as follows:

$$R_1 = \frac{2V}{10\mu A} = 200k\Omega;$$

$$R_1 \parallel R_2 = \frac{3-2V}{90-10\mu A} = 12.5k\Omega;$$

$$R_2 = \frac{R_1 \cdot R_1 \parallel R_2}{R_1 - R_1 \parallel R_2} = \frac{200 \cdot 12.5}{200 - 12.5} = 13.3k\Omega.$$

You can calculate resistors R₃, R₄, and R₅ from the equations for the threshold voltages:

$$V_{T1} = \frac{R_5}{R_3 + R_4 + R_5} V_{CC};$$

$$V_{T2} = \frac{R_4 + R_5}{R_3 + R_4 + R_5} V_{CC}.$$

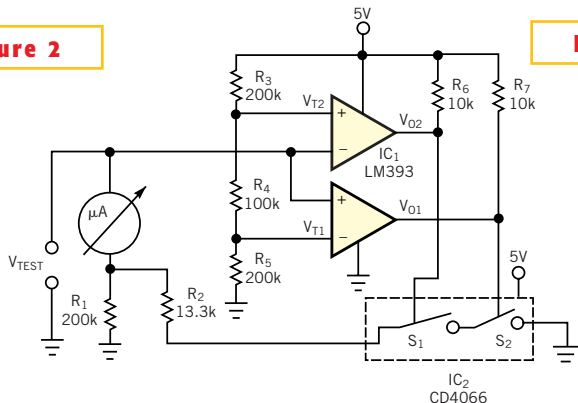
In this case, $V_{CC} = 5V$, $V_{T1} = 2V$, $V_{T2} = 3V$; hence, $R_3 = R_5 = 200k\Omega$, and $R_4 = 100k\Omega$.

REFERENCE

1. Raynus, Abel, “Indicator features expanded scale” *EDN*, Feb 21, 2002, pg 86.

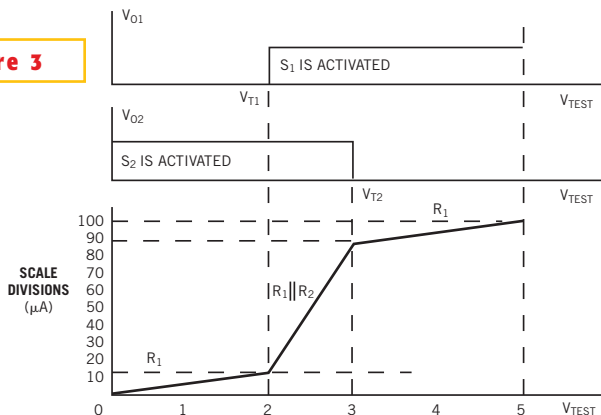
Is this the best Design Idea in this issue? Select at www.edn.com.

Figure 2



A window comparator and two analog switches collaborate to expand the green zone in Figure 1.

Figure 3



Within the thresholds of the window comparator, R₂ connects in parallel with R₁ to expand the scale of the voltmeter.

Butterworth filter has adjustable group delay

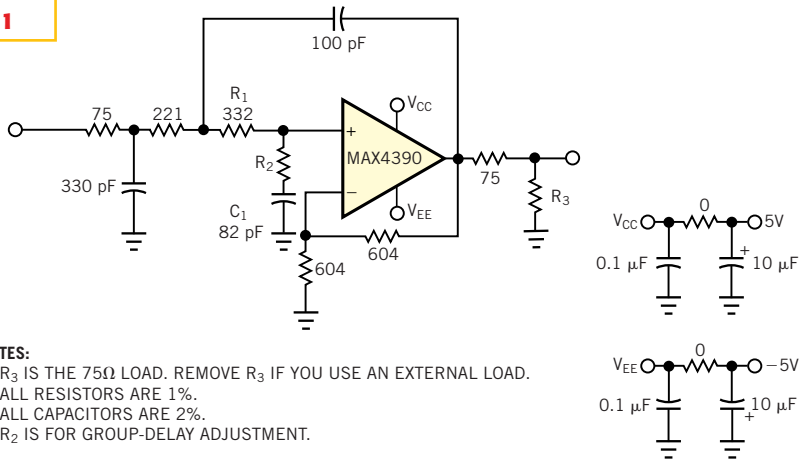
William Stutz, Maxim Integrated Products, Sunnyvale, CA

THE SALLEN-KEY REALIZATION of a 5.25-MHz, three-pole Butterworth filter has a gain of

2V/V and can drive 75Ω back-terminated coax with an overall gain of 1 (Figure 1). Used to reconstruct component-video (Y, Pb, Pr) and RGB signals, this filter has an insertion loss greater than 20 db at 13.5 MHz and greater than 40 db at 27 MHz (Figure 2). Like the antialiasing filter before an ADC, this filter removes the higher frequency replicas of a signal following a DAC. To preserve quality in the video waveform, you should minimize group-delay variations in the filter and any group-delay differential between filters. That requirement mandates a means for adjusting the filter's group delay without affecting its bandwidth. In Figure 1, the addition of R_2 in series with C_1 and R_1 creates a lag-lead network.

Keeping the sum of R_1 and R_2 constant and equal to the original R_1 value preserves bandwidth by preserving the dominant-pole frequency. Increasing the R_1 value, on the other hand, introduces a "lead" term that lowers group delay by reducing the rate of change in phase. For $R_2=0\Omega$ and $R_1=332\Omega$ in the circuit shown, the average group-delay variation over the filter bandwidth is about 25 nsec. Raising R_2 to 31.6Ω and lowering R_1 to 301Ω decreases the variation to approximately 15 nsec, and setting $R_2=59\Omega$ with $R_1=274\Omega$ decreases it to approximately 7 nsec. The last case has a less-than 0.5-dB effect on band-edge selectivity but does not change the filter's 3-dB bandwidth (Figure 3).

Figure 1

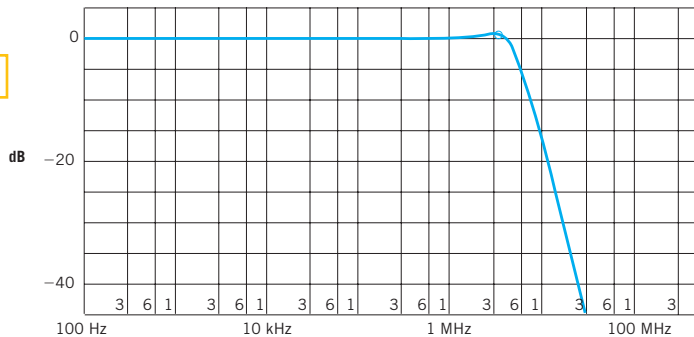


NOTES:

1. R_3 IS THE 75Ω LOAD. REMOVE R_3 IF YOU USE AN EXTERNAL LOAD.
2. ALL RESISTORS ARE 1%.
3. ALL CAPACITORS ARE 2%.
4. R_2 IS FOR GROUP-DELAY ADJUSTMENT.

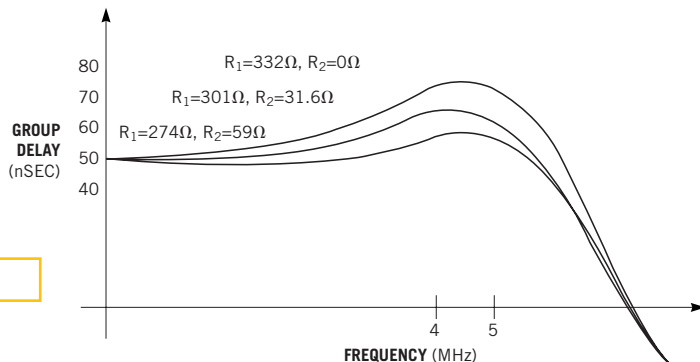
This three-pole Butterworth video-reconstruction filter has adjustable group delay.

Figure 2



The typical filter response for the circuit of Figure 1 is $R_1 + R_2 = 332\Omega$.

Figure 3



Selected values of R_1 and R_2 allow control of group-delay variation over the filter's passband.

Is this the best Design Idea in this issue? Select at www.edn.com.

Single transistor sequences multiple supplies

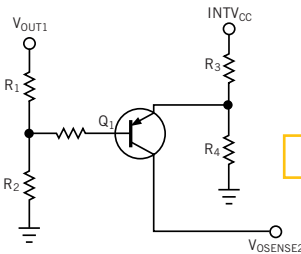
David Chen, Linear Technology Corp, Milpitas, CA

MANY DSP CHIPS, microprocessors, FPGAs, and ASICs require multiple power supplies that must deliver different voltages in a specific start-up sequence. Out-of-sequence voltages can cause excessive input current, logic errors, or even device failure. To sequence different supplies, a common approach is to regulate a lower voltage from a higher voltage using a linear regulator. Another approach is to use a series of Schottky diodes. Although simple in concept, these approaches can be expensive and difficult to implement in applications that require more than two power supplies. **Figure 1** shows a simple, low-cost approach that requires

only one pnp transistor to provide the necessary logic. **Figure 2** shows a dual power supply that uses the described circuitry to sequence the outputs.

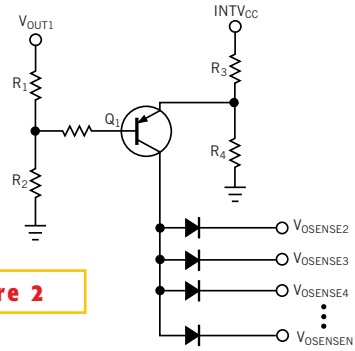
When V_{OUT1} is low, $V_{OSENSE2}$, the voltage

Figure 1



A simple one-transistor circuit synchronizes two outputs.

Figure 2



You can obtain multiple-output sequencing by adding ORing diodes.

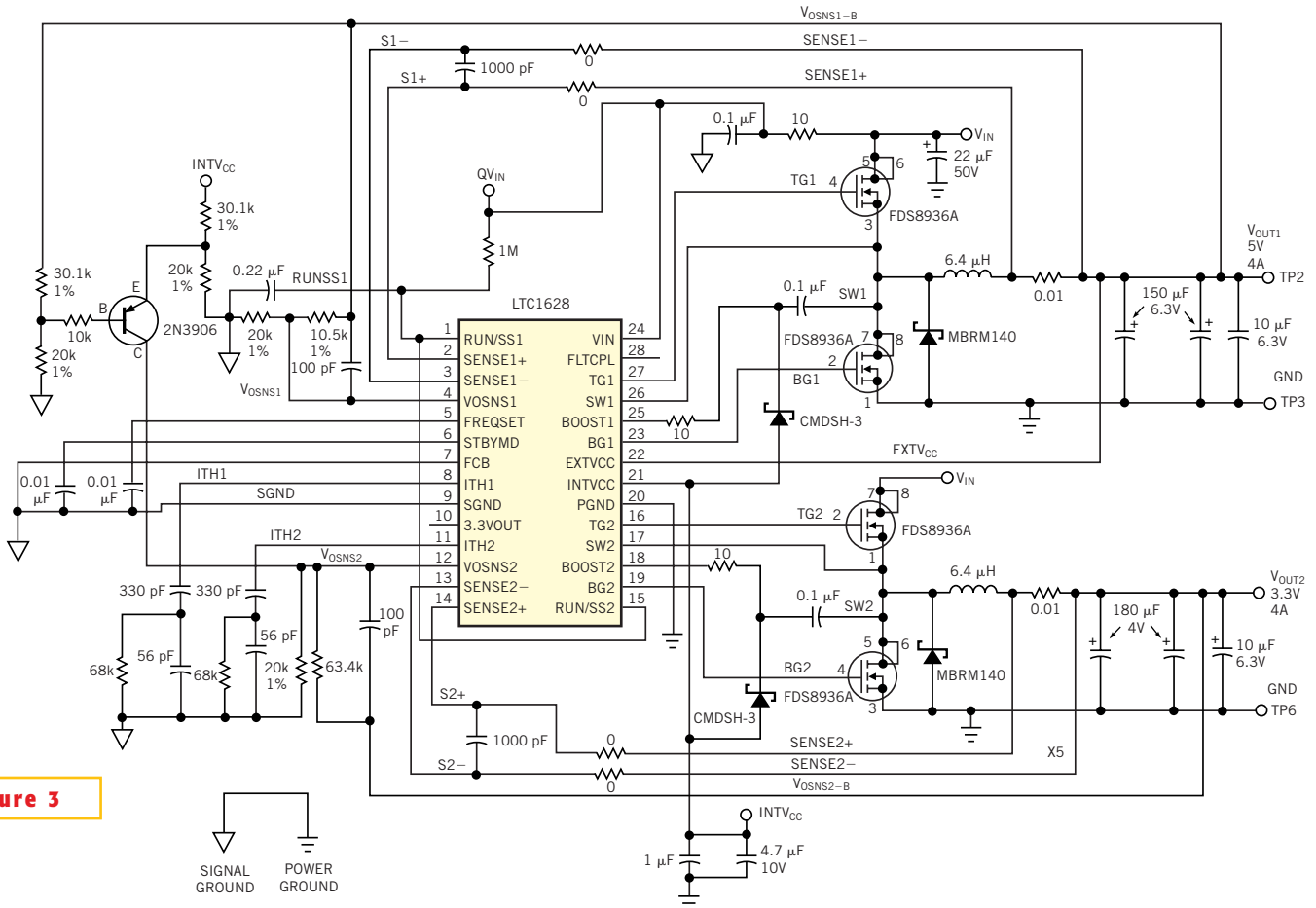


Figure 3

This dual-output supply uses the simple circuit in Figure 1.

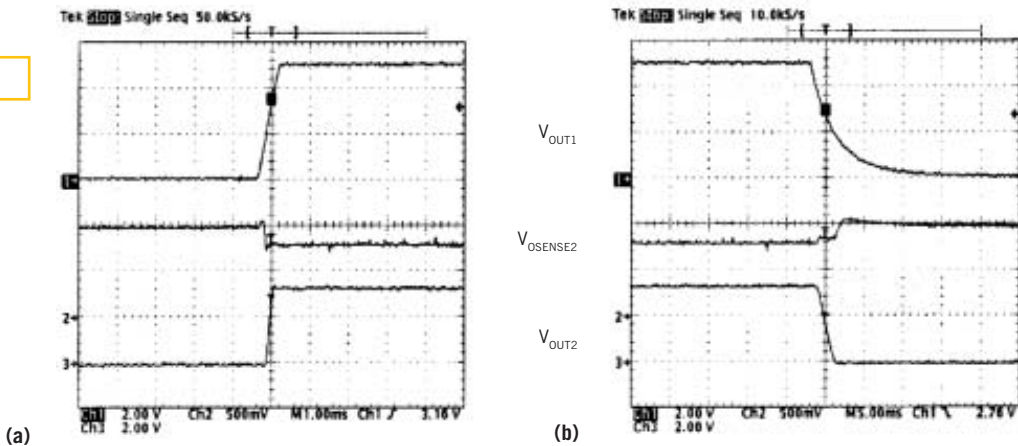
feedback for V_{OUT2} , goes high, and the second supply, V_{OUT2} , shuts off (Figure 3). When V_{OUT1} approaches its nominal level, Q_1 turns off. Q_1 then relinquishes control of $V_{OSENSE2}$, and V_{OUT2} resumes its normal start-up process. The process is similar for power-down sequencing. When V_{OUT1} is high, V_{OUT2} operates nor-

mally. When V_{OUT1} goes from high to low, $V_{OSENSE2}$ goes high and shuts off V_{OUT2} . More specifically, R_3 and R_4 set the clamping voltage for the $V_{OSENSE2}$ pin when V_{OUT1} is low, and R_1 and R_2 determine the V_{OUT1} voltage level at which Q_1 turns off. In cases of multiple supplies, you need only add ORing diodes at the collector of

Q_1 (Figure 3). The design uses an LTC1628 dual-output controller. You can see the sequenced-output waveforms in Figures 4a (at turn-on) and 4b (at turn-off).

Is this the best Design Idea in this issue? Select at www.edn.com.

Figure 4



These supply-voltage waveforms occur at turn-on (a) and turn-off (b).