Implement a stepper-motor driver in a CPLD

Stephan Roche, Santa Rosa, CA

Based on the Motorola (now Freescale, www.freescale.com) heavily used but obsolete SAA1042 stepper-motor-driver IC, this Design Idea describes a CPLD (complex-programmable-logic-device)-based implementation of a stepper-motor driver that can also replace the driver in SAA1027- or UCN5804B-based designs. The design uses only six macrocells of a Xilinx (www.xilinx.com) XC9536 CPLD and thus can implement multiple stepper-motor drivers in one small-capacity CPLD. The CPLD stepper-motor driver requires clock, direction, step-size, and reset inputs. The clock input accepts logic-level pulses and goes active on the pulse's positive edge.

The direction, or CW/CCW (clockwise/counterclockwise), input determines the motor's rotational direction. Depending on the motor's electrical connections, holding this input at 0V normally produces CW rotation, and a logic-1 input produces CCW rotation. The step-size—that is, full- or half-step—input determines the motor's angular rotation for each clock pulse. Holding this input low commands the motor to execute a full step for each applied clock pulse, and a high input

TABLE 1 DRIVER OUTPUTS FOR EACH MACHINE STATE

Outputs	Step 0	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7
A	0	0	1	1	1	1	0	0
A_n	1	1	1	0	0	0	0	1
B	0	0	0	0	1	1	1	1
B_n	0	1	1	1	1	0	0	0

produces a half-step. A high level on the reset input puts the motor in a previously defined state and commands the CPLD to ignore any incoming clock pulses.

The CPLD's outputs comprise A and A_n and B and B_n phases, each of which controls one of the motor's two coils through external power drivers IC_2 and IC_3, which operate at the motor's nominal voltage (**Figure 1**). A pair of Schottky diodes at each driver's output protects the drivers' outputs during inductive-voltage transients induced by reversing the windings' currents. Using MOSFET drivers with internal diodes, such as Microchip's (www.microchip.com) TC4424A dual driver, may eliminate the requirement for external diodes.

The CPLD's program comprises an eight-state Moore finite-state machine that corresponds to the motor's eight half-step states. **Table 1** shows the driver's outputs for each machine state. In full-step state mode, the state machine executes only Step 0, Step 2, Step 4, and Step 6. At each clock pulse's rising edge, the machine state changes from Step(n) to Step(n+1) if CW/CCW is high or from Step(n) to Step(n−1) if CW/CCW is low. You can download a generic VHDL implementation of the stepper-motor-driver firmware from this Design Idea's online version at www.edn.com/070215di2. Although written for an XC9536 CPLD, the code is also suitable for any CPLD or FPGA target device.

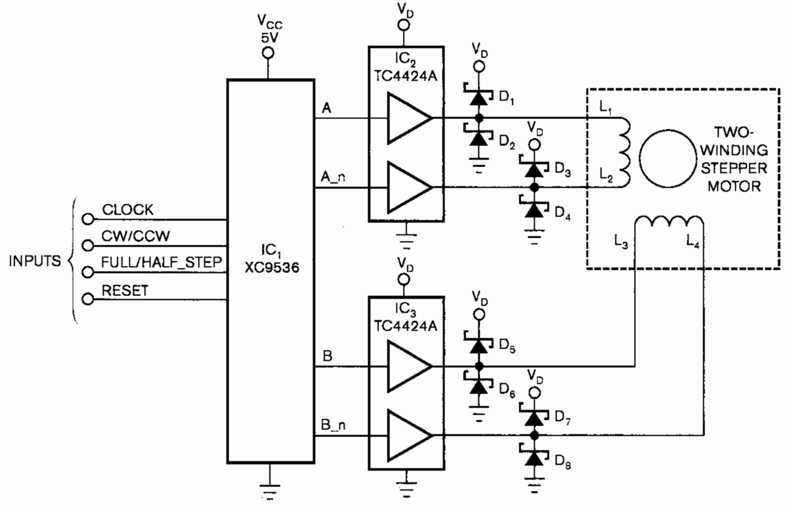


Figure 1 Emulating a dedicated stepper-motor controller, a programmable-logic device, IC_1, applies stepper-motor signals to motor drivers IC_2 and IC_3.

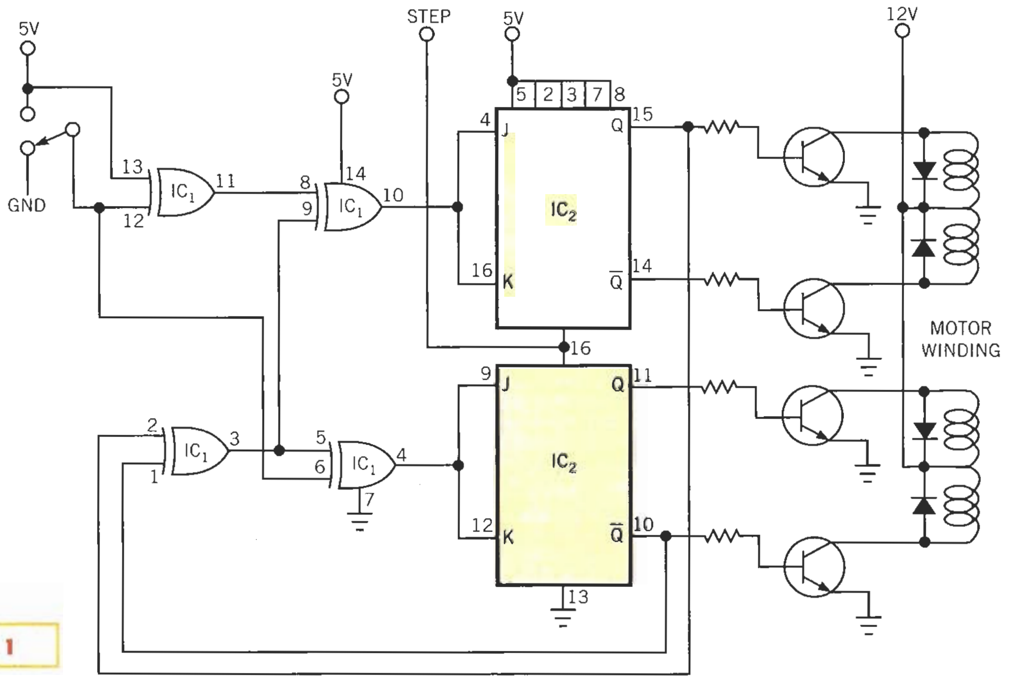
Simple circuit controls stepper motors

Noel McNamara, Analog Devices, Limerick, Ireland

Stepper motors are useful in many consumer, industrial, and military applications. Some, such as personal-transportation systems, require precise speed control. Stepper-motor controllers can be simple (**Figure 1**), but they require a variable-frequency square wave for the clock input. The AD9833 low-power DDS (direct-digital-synthesis) IC with an on-chip, 10-bit DAC is ideal for this task, because you need no external components for setting the clock frequency (**Figure 2**). The device contains a 28-bit accumulator, which allows it to generate signals with 0.1-Hz resolution when you operate it with a 25-MHz MCLK (master clock). In addition, the circuit can easily stop the motor if you program a 0-Hz output frequency.

Figure 3 shows the complete system. The most significant bit of the on-chip DAC switches to the V_{OUT} pin of the AD9833, thus generating the 0-to-V_{DD} square wave that serves as the clock input to the stepper-motor controller. Writing to the frequency-control registers via a simple, three-wire interface sets the clock

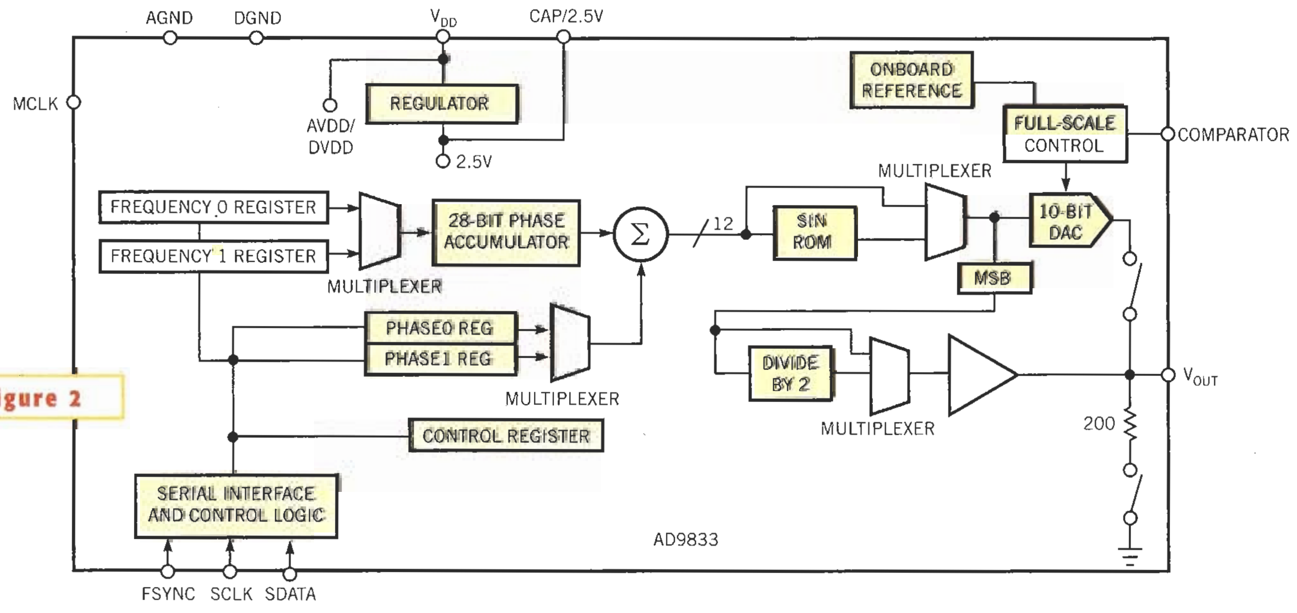
Figure 1



A stepper-motor controller requires only a few logic circuits.

NOTES: IC_1: SN74HC86. IC_2: SN74LS76A.

Figure 2



The AD9833 DDS IC generates frequencies with 0.1-Hz resolution.

frequency. Writing a 0 to the frequency register stops the clock, thereby stopping the stepper motor. When you are not using the DAC, you can power it down by writing to a control register. This power-down action results in the AD9833's drawing only 2 mA from the supply. Reducing the MCLK frequency can further reduce the supply current. The AD9833 is available in a tiny, 10-lead package, so you can assemble the complete control system on a very small pc board.□

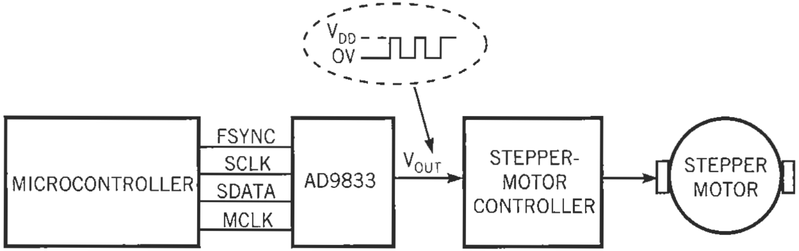


Figure 3

The complete stepper-motor controller uses a DDS IC to generate the variable frequencies for the circuit in Figure 1.