TI TechNotes

Stepper Motor Control Using MSP430™ MCUs

Texas Instruments

Introduction

Stepper motors are a form of brushless DC electric motor that convert input pulses into specifically defined increments as each pulse creates rotation toward a fixed angle. By dividing full rotation into equal steps, the motor's position can then be controlled with advanced precision. Offering several advantages including high reliability, excellent response times, and a wide range of torque and speed options, stepper motors are used in consumer electronics, industrial systems, and factory automation.

Stepper motor drivers, such as the DRV8825 used for the purposes of this document, employ a simple step and direction interface to allow easy interfacing to microcontrollers (MCUs). By simply adjusting a timer pulse width modulation (PWM) output frequency and incorporating an additional general-purpose output pin, it is possible for a MSP430™ MCU to regulate the speed and direction at which a stepper motor is driven. Commands are also simple enough such that a stepper motor can be controlled by the MSP430FR2000 MCU, a cost-effective MCU with 512 bytes of main memory. To get started, download project files and a code example demonstrating this functionality.

Implementation

The DRV8825 manages all protection features necessary when interfacing with a brushless DC motor, and therefore, all that is required from the MSP430 device are the step, direction, and ground connections. The RESET and SLEEP pins are pulled high, because they do not need to be directly controlled, and the M2 pin is also pulled high so that 1/16th microstepping mode is used. Operating supply voltage ranges, maximum drive currents, timing requirements, and allowed modes vary between stepper motor drivers, so see the device-specific data sheet. A DRV8825-based breakout board is used for the purpose of this demonstration, and Figure 1 shows the connections for this board. The DRV8886AT is a suitable alternative that includes updated features like autotune and internal current sensing.

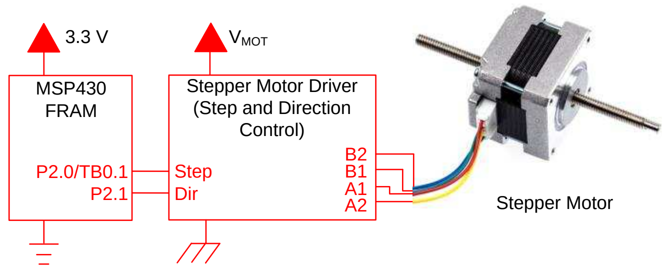

Figure 1. Stepper Motor Circuit Diagram

Timer_B0 is used to generate the PWM output to the DRV8825 STEP pin through P2.0. MCLK is initialized at 16 MHz then divided by two to avoid ferroelectric random access memory (FRAM) wait states, The resulting 8-MHz frequency shared with SMCLK is used to source the timer operating in up mode. TB0CCR0, which controls the output frequency, varies depending on the desired stepper motor speed. TB0CCR1 maintains a 50% duty cycle as is expected by the DRV8825.

A UART host interface must connect to P1.6/UCA0RXD to send commands to the eUSCI_A0 peripheral of the MSP430FR2000 device. An MSP-FET programmer and debugger and MSP-TS430PW20 target development board are used for evaluation. A baud rate of 9600 with one stop bit and no parity is provided as the default. However, this can be easily altered to meet application requirements. A hexadecimal input value of 0x00 stops the motor movement. An input value of 0x0A inverses the polarity of the P2.1 output connected to the DIR pin, which causes the motor to change direction. Other values from 0x01 to 0x09 change the motor speed from slowest to fastest, respectively. Valid frequency and timing requirements depend on the specific stepper motor driver being used. Table 1 lists the default frequencies used in the example.

Table 1. Hexadecimal to Frequency Mapping

Received	TB0CCR0	Frequency (kHz)
0x00	0	0
0x01	8000	1
0x02	4000	2
0x03	2000	4
0x04	1000	8
0x05	500	16
0x06	250	32

Table 1. Hexadecimal to Frequency Mapping (continued)

Received	TB0CCR0	Frequency (kHz)
0x07	125	64
0x08	64	125
0x09	32	250

Performance

Figure 2 shows three examples of PWM waveforms generated from hexadecimal terminal entries (in red), validating the entries provided in Table 1. Faster entries were not tested with a physical stepper motor setup, as it was limited by the power supply's maximum current draw.

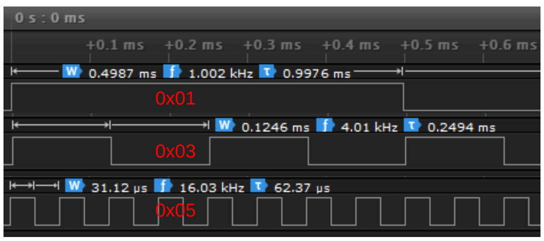

Figure 2. Frequency Modulation

The code example uses approximately 250 bytes of main memory. This leaves more space for additional application code, detailed movement sequences, or dual motor control through P2.1/CCR2. An increase in code development might require migrating to MSP430 MCUs with larger memory footprints. If the eUSCI peripheral is not required, more timer outputs can be generated through the P1.6 and P1.7 pins. However, one timer (Timer_B0) and two capture/compare registers (CCR1 and CCR2) limit the number of distinct frequencies that can be generated at any time.

Because the subsystem master clock (SMCLK) is required as the timer clock source, low-power mode 0 (LPM0) can be accessed during inactivity and consumes 300 µA on average. If only using frequencies of 8 kHz or below, then auxiliary clock (ACLK) can supply the timer instead, and LPM3 mode can be used to achieve power consumption of 1 µA (LFXT) or 17 µA (REFO).

Device Recommendations

The device used in this example is part of the MSP430 Value Line Sensing portfolio of low-cost MCUs, designed for sensing and measurement applications. This example can be used with the devices shown in Table 2 with minimal code changes. For more information on the entire Value Line Sensing MCU portfolio, visit www.ti.com/MSP430ValueLine.

Table 2. Device Recommendations

Part Number	Key Features
MSP430FR2000	0.5KB FRAM, 0.5KB RAM, eComp
MSP430FR2100	1KB FRAM, 0.5KB RAM, 10-bit ADC, eComp
MSP430FR2110	2KB FRAM, 1KB of RAM, 10-bit ADC, eComp
MSP430FR2111	3.75KB FRAM, 1KB RAM, 10-bit ADC, eComp

MSP430 is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265