

Connecting MSP430s to the DRV8824

LT Thomas

4/1/11

Executive Summary:

Texas Instruments (TI) offers many motor controlling solutions, the IC we will be involving ourselves in is the DRV8824. The DRV8824 is a motor control for Stepper motor applications, this chip makes controlling a stepper motor more efficient and pain free.

Keywords: MSP430, PWM, DRV88xx

Table of Context:

Background	3
Understanding the DRV8824	4
Applications with the MSP430	
Simple MSP430F2013 Example	6
Adding more functionality	9
References	10

Background

The stepper motor came about around the 1960s, it was developed as a replacement for servos. The Stepper motor is unique from all other motors for many reasons:

1. There is no feedback required to control the motor
2. The rotation motor at stator never touch so there is no mechanical wear
3. The polarity is axial

Because this motor has a unique form of no feedback in the sense that you can predict the location of the motor ahead of time simply by knowing the number of steps you are sending it. The torque of the motor can be controlled by the number of phases that are turned on at a time which relates to the amount of degrees the motor spins. The smaller steps the motor takes the more torque that can be generated. The only real downside to stepper motors is the size and also the amount of power they require to run a simple step.

The DRV8824 is a TI solution for controlling a Stepper motor. This IC allows for easy control of a motor by a microcontroller. This motor controller allows for a 8-45 Volt @ 1.5A max motor to be controlled. Full, half, quarter and microsteps can all be implemented with this chip through selectable pins. There is also a feature that takes the motor voltage and outputs a 3.3 Volt power pin that can be used for microcontrollers. There is a safety feature built in and output can be seen by a pin on the DRV. Also the DRV provides variable current and decay speeds.

Understanding the DRV8824

Before coding the MSP430 we must first understand what is needs to be connected. Below in figure 1, the functional block diagram can be found. So we have here 2 stages, a power stage and a control stage. One really great feature of the DRV8824 is that only 1 outside power supply is needed, from this an internal regulator creates a 3.3v output that can then be used to power a microcontroller and the AVREF & BVREF. Another great feature is all the PWMs are handled inside the DRV8824, this means that the only inputs it needs is simple rising edge oscillator and general input outputs.

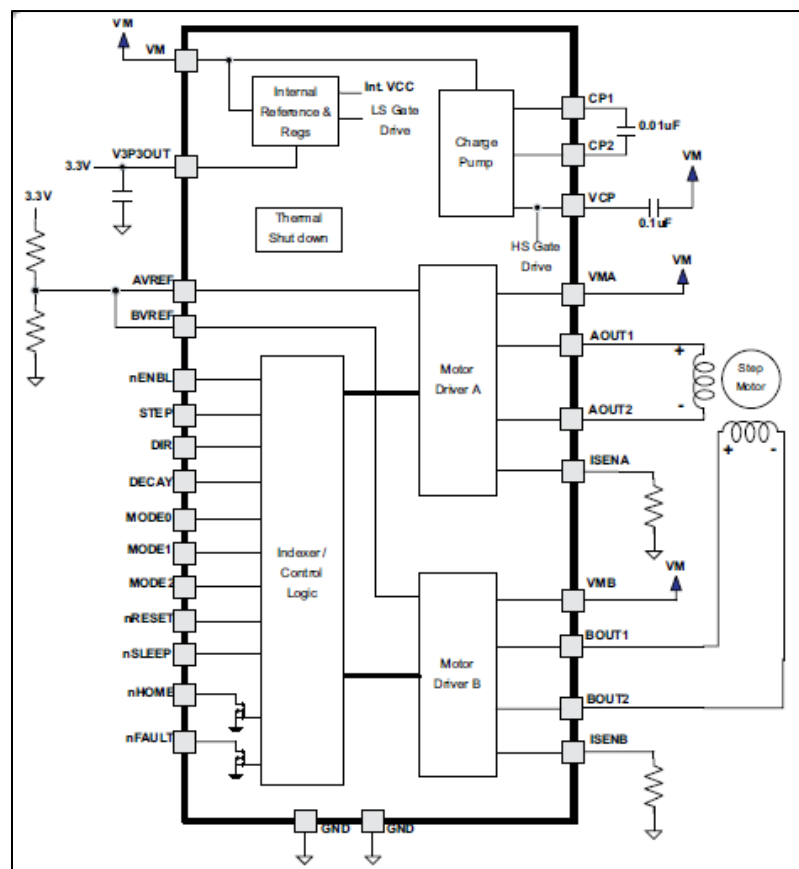


Figure 1. Function Block Diagram

There is four different terminal sections as seen below in Figure 2; power and ground, control, status, and output. Power and ground will be wired just as described in the table, the motor voltage

(VM) should be supplied according to the voltage rating on the motor being driven. Next the control has 7 general inputs that are either high or low and do not need to be changed often. There are 3 analog inputs, 2 are for current control of each phase of the motor, and 1 is for the decay rate. Next comes the 2 status ports, nHOME is used to detect when the motor has made 1 complete rotation (this can be used to calculate speed), and nFAULT allows to know the user to know when an overcurrent or overtemp condition occur and can shut off the motor. The output stage is directly connected to the motor and these are the ports that move the motor, there are also current sense ports that can be used for feedback.

NAME	PIN	I/O ⁽¹⁾	DESCRIPTION	EXTERNAL COMPONENTS OR CONNECTIONS
POWER AND GROUND				
GND	14, 28	-	Device ground	
VMA	4	-	Bridge A power supply	Connect to motor supply (8 - 45 V). Both pins must be connected to same supply.
VMB	11	-	Bridge B power supply	
V3P3OUT	15	O	3.3-V regulator output	Bypass to GND with a 0.47- μ F 6.3-V ceramic capacitor. Can be used to supply VREF.
CP1	1	IO	Charge pump flying capacitor	Connect a 0.01- μ F 50-V capacitor between CP1 and CP2.
CP2	2	IO	Charge pump flying capacitor	
VCP	3	IO	High-side gate drive voltage	Connect a 0.1- μ F 16-V ceramic capacitor to VM.
CONTROL				
nENBL	21	I	Enable input	Logic high to disable device outputs and Indexer operation, logic low to enable
nSLEEP	17	I	Sleep mode input	Logic high to enable device, logic low to enter low-power sleep mode
STEP	22	I	Step input	Rising edge causes the indexer to move one step
DIR	20	I	Direction input	Level sets the direction of stepping
MODE0	24	I	Microstep mode 0	MODE0 - MODE2 set the step mode - full, 1/2, 1/4, 1/8/ 1/16, or 1/32 step
MODE1	25	I	Microstep mode 1	
MODE2	26	I	Microstep mode 2	
DECAY	19	I	Decay mode	Low = slow decay, open = mixed decay, high = fast decay
nRESET	16	I	Reset input	Active-low reset input initializes the indexer logic and disables the H-bridge outputs
AVREF	12	I	Bridge A current set reference input	Reference voltage for winding current set. Normally AVREF and BVREF are connected to the same voltage. Can be connected to V3P3OUT. A 0.01- μ F bypass capacitor to GND is recommended.
BVREF	13	I	Bridge B current set reference input	
NC	23		No connect	Leave this pin unconnected.
STATUS				
nHOME	27	OD	Home position	Logic low when at home state of step table
nFAULT	18	OD	Fault	Logic low when in fault condition (overtemp, overcurrent)
OUTPUT				
ISENA	6	IO	Bridge A ground / Isense	Connect to current sense resistor for bridge A.
ISENB	9	IO	Bridge B ground / Isense	Connect to current sense resistor for bridge B.
AOUT1	5	O	Bridge A output 1	Connect to bipolar stepper motor winding A.
AOUT2	7	O	Bridge A output 2	Positive current is AOUT1 → AOUT2
BOUT1	10	O	Bridge B output 1	Connect to bipolar stepper motor winding B.
BOUT2	8	O	Bridge B output 2	Positive current is BOUT1 → BOUT2

Figure 2. Terminal Functions

If a user wanted to use as little ports as possible they would only need to connect to the STEP port and hardwired the rest.

First we will look at just controlling the speed of a stepper motor in this first application seen below in figure 3. If you notice I have wired most the general inputs for the DRV to either ground or power, this is done to reduce the amount of connections between the micro-controller and the DRV8824. With the application below just a 4-pin micro-controller could operate the motor. Ideally we would never hardwire the general ports to specific signals, instead wire to the micro-controller so we could have better control of the motor. But to reduce the learning curve we will go through the code for this application first.



6

clockwise by 1 full step at full current reference every time the STEP port receives a rising edge. Now on the micro-controller side we connect STEP to pin 13, the reason we are doing this is because pin 13 has a special function and that is Pulse Width Modulation (PWM). To drive the motor we need to send a continuous pulse, this can be done perfectly with the PWM registers, the reason we use this over just hard coding a pulse signal in the code is because PWM runs consistently in the background allowing the micro-controller to run other functions, PWM is also very easy to implement.

Now let's step through how to implement a pulse that runs in the background of the MSP430. The first line of code you will see in the code for the MSP430F2013 in figure 4 on the next page contains the information for the watchdog timer (WDT). A watchdog timer is commonly used in low power microcontrollers because they are constantly switching into low power modes and to ensure that the microcontroller does not get stuck in an infinite loop. The WDTCTL is the control register for the WDT, control registers are what need to be filled with certain bits to enable special functions. We are sending the WDTPW & WDTOLD, this tells the timer to stop counting, and this is done to ensure the flag will never be set for the timer so we are basically turning it off. The next two lines BCSCTL1 and DCOCTL are setting the master clock frequency to 1 MHz using the Digital Clock Oscillator. The next three lines that have P2 as the first two letters is setting Port 2 bit 6 (0x40 in hex) to be an output and PWM feature enabled. If P2SEL was not turned on also then Port 2 bit 6 would just be a General Port Input Output (GPIO). TACTL is the timer register, and what we are doing in this line is setting the time to use the 1MHz DCO and count upwards. CCR0 sets the pulse frequency, so a frequency of 8000 means that we will have a frequency of $1/(8000 \times 1\mu) = 125 \text{ Hz}$. Next we set CCTL1 to an output

mode that will allow us to adjust the duty cycle. CCR1 controls the width of the inside first high signal, we set the CCR1 to 4000 which is exactly half of the pulse so we will have a duty cycle of 50%.

```
#include <msp430.h>
void main(void)
{

    WDTCTL = WDTPW + WDTHOLD;           //Stop WDT
    BCSCTL1 = CALBC1_1MHZ;              //Set DCO to 1MHz
    DCOCTL = CALDCO_1MHZ;

    P2SEL |= 0x40;                      //p2.6 in TA mode pwm
    P2DIR |= 0x40;                      //p2.6 output
    P2OUT |= 0x40;

    TACTL = TASSEL_2 + MC_1 + ID_0;      // SMCLK , UP mode

    //outmodes: 0-output, 1-set, 2-toggle/reset,
    //           3-set/reset, 4-toggle, 5-reset,
    //           6-toggle/set, 7-reset/set

    CCR0 = 8000;                        // PWM Period, 16 bits
    CCTL1 = OUTMOD_7;                   // was 6CCR1 reset/set
    CCR1 = 4000;                        // CCR1 PWM duty cycle

    __bis_SR_register(CPUOFF);          // Enter LPM0

}
```

Figure 4. MSP430F2013 Code

The `__bis_SR_register(CPUOFF)` line is used to put the MSP430 in one of its 4 power modes. Normally microcontroller code requires an infinite loop so it will never reach the end of a program, but with the low power mode we are turning off the processor and just using the timer.

I will now talk briefly about how to extend this idea to gain greater control of your motor. In the schematic below you can see that I have wired much more pins to the MSP430, let's start from the top right and work around clockwise. So first we have the PWM signal from P2.6 as before connecting to the STEP pin on the DRV, this controls the speed of the motor. For nHOME we would tie an interrupt to this pin, then whenever the interrupt was tripped we would get the time from the last interrupt and then scale this to get a RPM value. DIR would simply be a General Output, and based on what direction we wanted we would send it a 1 or a 0, we can also tie this to a toggle switch to get a physical way to change directions. Next comes

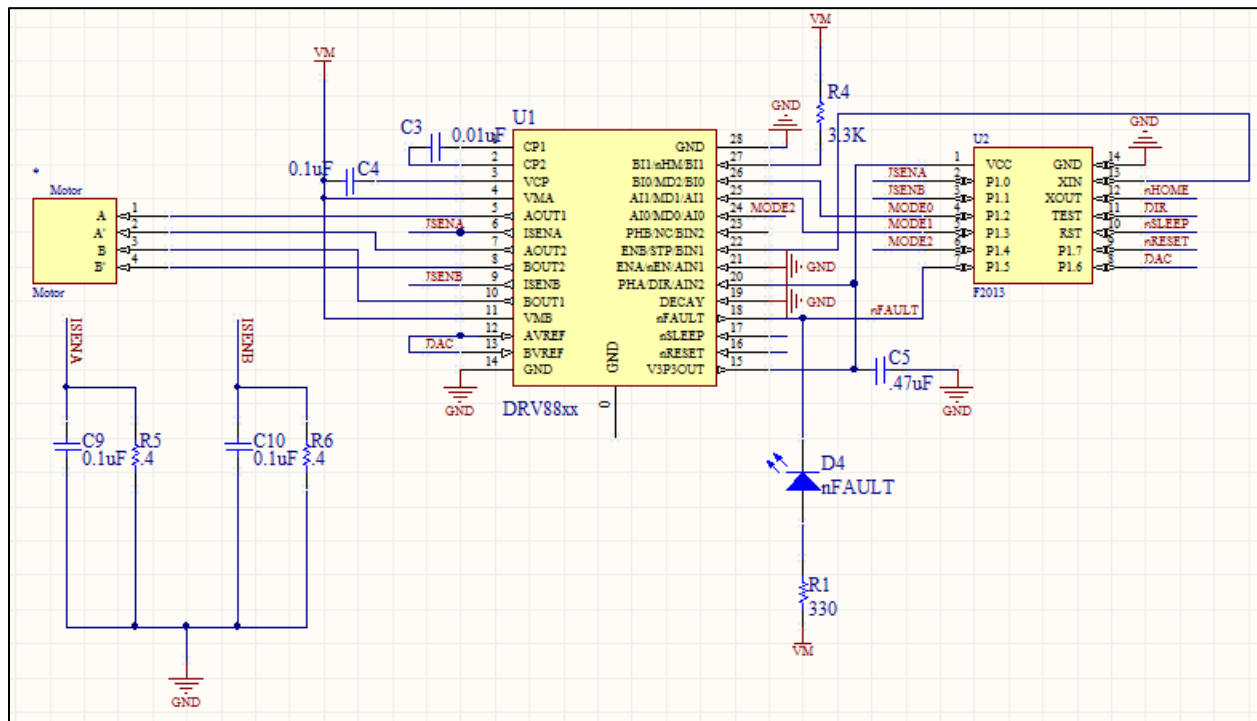


Figure 5 Application Schematic

nRESET and nSLEEP, these are used for resetting position and entering low power modes. The DAC pin is used to control AVREF and BVREF on the DRV, these pins on the DRV control that amount of current allowed through the motor, so power can be saved by adding this output.

nFAULT uses internal features to detect when something has gone wrong with the motor controller, so we typically tie an LED to this to visually see something is wrong, and also tie it to the MSP430 so it can realize this has happened and stop providing power. MODE 0,1,2 are used to control the resolution of steps, so these are all General Inputs into the DRV and this is something easily controlled by the MSP430.

References

“MSP430F2013 Datasheet.” Texas Instruments.

<http://focus.ti.com/lit/ds/symlink/msp430f2013.pdf>

“DRV8824 Datasheet.” Texas Instruments.

<http://focus.ti.com/lit/ds/symlink/msp430f2013.pdf>