Introduction to Mechatronics (ME/AE 6705)
Lab Assignment 8

DC Motor Speed Control using MSP432 LaunchPad and PWM Signals

8.1 Objective

The main objective of this lab is to interface and control a brushed DC motor with the MSP432 LaunchPad, using PWM signals and an H-bridge motor driver. You will control both the *speed* and *direction* of the DC motor using the MSP432 LaunchPad.

8.2 Deliverables and Grading

To get credit for this lab assignment you must:

1. Demonstrate proper operation of your code to TAs or instructor during office hours or demo hours. You must write your own code for this lab – no lab groups are allowed. (80 points)
2. Submit the commented final version of your code on Canvas (single .c file containing your main() function, do not submit header files, etc. You should only submit a single file.) (Pass/Fail)

8.3 Setup

This lab requires the *Code Composer Studio*, the *TI MSP432P401R LaunchPad*, and an *H-bridge motor driver circuit* which will be built on a breadboard. The lab uses onboard features of the MSP432 LaunchPad such as *Timer A*, *GPIOs* and *ADC 14*.

All components needed for this lab are included in the Mechatronics kit designed for the course. Four AA batteries (not included) will be needed to power your system. You can use disposable alkaline batteries or rechargeable NiMH 1.5V batteries.

8.4 Problem Statement

Build an H-bridge motor driver circuit to control a brushed DC motor. This motor driver circuit will be used to control the speed and direction of the motor. The direction and speed

control signals are generated by your software program and the MSP432 LaunchPad. Speed control will be based on an analog input from a voltage divider circuit using a potentiometer. The main component used in the driver circuit is an H-bridge IC.

The main hardware component used in the motor driver circuit is the H-bridge IC (SN754410).

Your software control program should operate as follows: Initiate S1 and S2 switches as digital inputs. S1 and S2 should be used as toggle switches. The motor should start spinning in one direction when S1 is pressed, and stop spinning when S1 is pressed again. Same for S2, but in the opposite direction if S2 is pressed.

LEDs can be used to indicate and differentiate the motor in ON condition vs OFF condition for the 2 spin directions. Based on the analog input voltage received from the voltage divider circuit, the duty cycle of the PWM signal should be varied from 0% to 100% to control motor speed while the motor is spinning in either direction. All motor speed control should be accomplished using *PWM signals* generated by `Timer_A0`.

More details about hardware and software requirements will be discussed in the next sections.

8.5 Hardware

8.5.1 Brushed DC Motor

The brushed DC motor is a type of motor that operates on DC voltage and current. It is bidirectional, meaning if the supply terminals are swapped the motor turns in the opposite direction. An H-bridge helps in achieving this functionality without physically having to swap the terminals.

The DC motor assemblies (with gearbox and encoder already installed) used in this lab are manufactured by Pololu. Specifications for this motor can be found at: https://www.pololu.com/product/1520. The motor is capable of high speeds, with a rated output speed of 150 rpm.

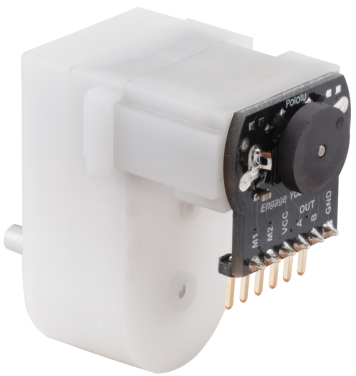


Figure 1: DC motor with encoder assembly (Image courtesy of Pololu).

For this lab assignment, you will connect the two terminals (labeled as M1 and M2) of the

DC motor to the H-bridge motor driver circuit. The other 4 pins of the motor are encoder signals, which will not be used in this lab assignment.

8.5.2 H-bridge IC

The H-bridge IC that we use for this lab is the TI SN754410. SN754410 is a quadruple half H-bridge driver IC. It has four half H-bridges that can be used either independently for motors with unidirectional applications, or in pairs for motors with bidirectional applications. The datasheet for SN754410 can be found at: http://www.ti.com/lit/ds/symlink/sn754410.pdf

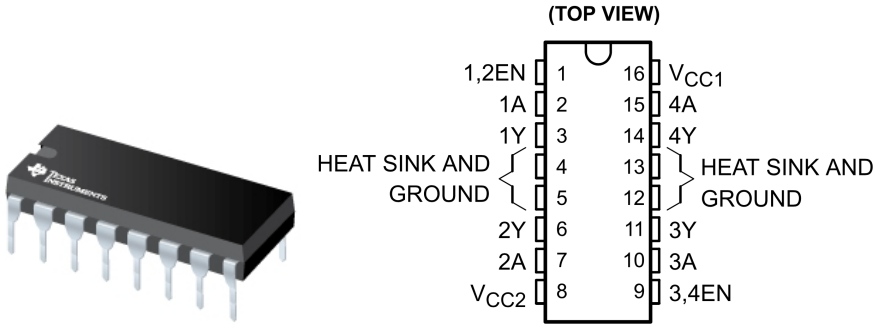


Figure 2: SN754410 pinout (Image courtesy of Texas Instruments).

We will be using channels 1 and 2 of the H-bridge driver IC as a pair to control the motor. This setup forms a full H-bridge which enables bidirectional operation of the DC motor.

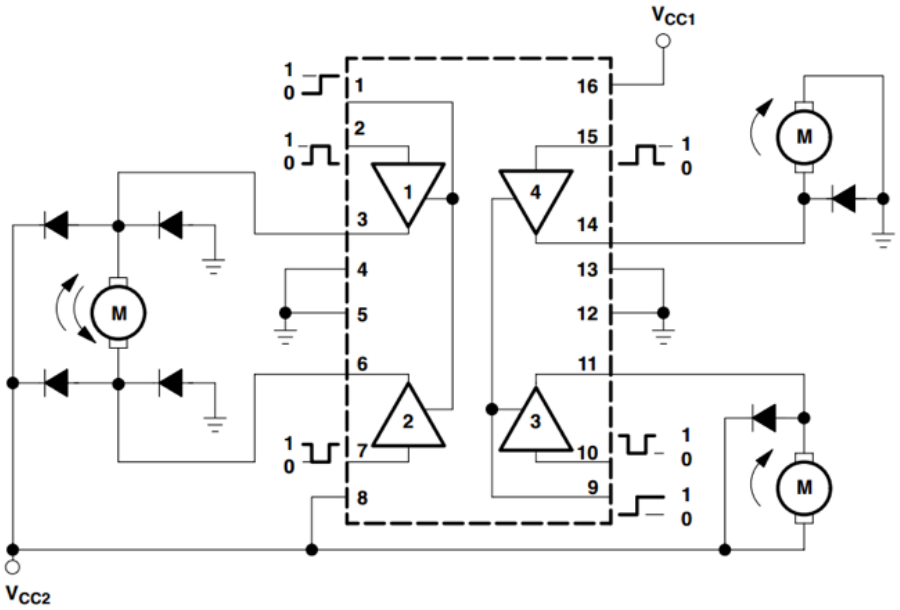


Figure 3: SN754410 functional block diagram (Image courtesy of Texas Instruments).

Figure 3 shows the possible wiring configurations that can be used with the SN754410 H-bridge IC. Note that in the figure, the interfaces at ports 3 and 4 are shown for unidirectional operation of two motors. The left side of the diagram (ports 1 and 2) shows the wiring schematic for bidirectional operation of a motor.

Note: For this lab assignment, we will be using the **bidirectional configuration** in order to have bidirectional control of the DC motor.

Note that the SN754410 IC comes in a PDIP package, and has an identifier to help you locate Pin 1 on the chip: there is a Half Circle or Notch on the end of the chip, Pin 1 is to the left of the notch (Figure 2). IC pins are numbered in a counter clockwise fashion from Pin 1.

The pins of the SN754410 H-bridge IC are described in Table 1.

Pin Number	Pin Name	Pin Function	MSP432 Connection
1	1,2EN	Enable signal for driver channels 1 and 2 (active high input)	3V3
2	1A	Input PWM signal for driver channel 1	P2.4
3	1Y	Output signal to the motor for driver channel 1	
4	Heat Sink/GND	Connect to Ground	
5	Heat Sink/GND	Connect to Ground	
6	2Y	Output signal to the motor for driver channel 2	
7	2A	Input PWM signal for driver channel 2	P2.5
8	VCC2	Power supply for motors (connect to +6V)	
9	3,4EN	Enable signal for driver channels 3 and 4 (active high input)	NC
10	3A	Input PWM signal for driver channel 3	NC
11	3Y	Output signal for driver channel 3	NC
12	Heat Sink/GND	Connect to Ground	
13	Heat Sink/GND	Connect to Ground	
14	4Y	Output signal for driver channel 4	NC
15	4A	Input PWM signal for driver channel 4	NC
16	VCC1	Power supply for internal logic (connect to 5V)	5V

Table 1: SN754410 Pins Description

To understand exactly what the input control PWM signals do, refer to Figure 3. Setting pin 2 of SN754410 high and simultaneously setting pin 7 of SN754410 low will turn the motor (attached to the output pins 3 and 6) in one direction. To turn the motor in the other direction, pin 2 should be low and pin 7 of SN754410 should be high. Note that these input pins can also accept PWM signals from the MSP432. In order to operate the motor in either direction, a PWM signal should be applied to one of these two pins while maintaining the other pin low.

Setting the enable signals (pin 1 and pin 9) to high will enable the associated driver channels. To make sure that the driver channels are enabled, we manually connect the enable pins to 3.3V, as will be shown in the circuit schematic in Figure 6.

8.5.3 Diodes

Diodes are semiconductor devices used to control the direction of current flow. They are used often in mechatronic devices. In this lab, 1N5819 diodes will be used as flyback diodes for the motor driver. Flyback diodes provide alternative path for current in motor coil and mitigates arcing/excessive current through transistors, to avoid damaging the transistors.

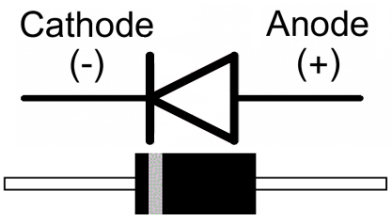


Figure 4: The diode circuit symbol, and the physical shape.

As shown in Figure 4, the silver line on 1N5819 diode indicates the cathode pin, which matches the vertical line in the diode circuit symbol.

8.5.4 400-Point Breadboard

A solderless breadboard is included in the Mechatronics kit for this course. Solderless breadboards are commonly used for prototyping because they allow you to quickly build temporary circuits without soldering.

The breadboard makes it easy to connect the electronic components of a project by inserting component leads and jumper cables into pins on the breadboard. When you are done or want to change your circuit, it is easy to take your circuit apart.

The pins are arranged in groups of 5. The 5 pins in each group are electrically connected to each other at the back of the board. You connect leads and cables together by inserting them into pins within the same group.

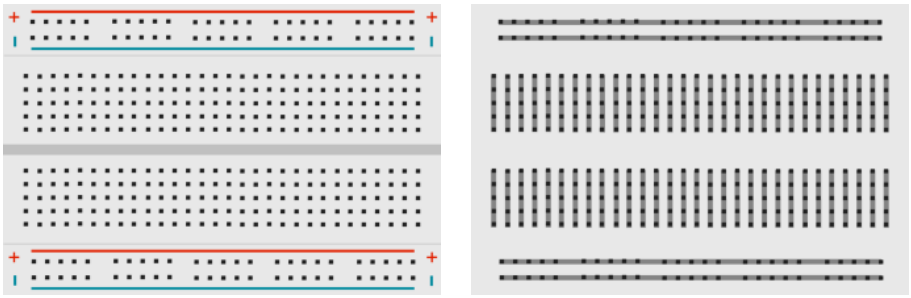
Power rails at the top and bottom are marked with red (+) and blue (−) stripes. The groups in each rail are electrically connected along the entire length of the stripe. The remaining 5-pin groups on the board are labeled with numbers and letters. Each group is electrically isolated from the others.

The gap at the center of the breadboard allows easy connection of electronic components provided as dual-inline packages, such as the SN754410 IC.

For best results, use solid wires when breadboarding; you will find a pre-cut jumper wire kit in the Mechatronics kit.

8.6 Circuit Schematic

Figure 6 shows the circuit we use to interface the brushed DC motor with the MSP432 LaunchPad. The motor is powered via four AA batteries, which provide a voltage in the



(a) (b)

Figure 5: (a) Front of solderless breadboard showing power rails and connection pins. (b) Interconnections at back of board (normally hidden). The 5-pin groups in each power rail are interconnected. All other 5-pin groups are isolated (Image courtesy of Texas Instruments).

range of 4.8V to 6V. We recommend to connect the positive wire of the battery holder to the (+) power rail of the breadboard, and then use the power rail for all the pins connected to +6V. Similarly, the negative wire of the battery holder and the GND pin of the LaunchPad can be connected to the (−) power rail of the breadboard, so that all the pins connected to GND can share that power rail. The +5V required for the VCC1 pin of SN754410 (pin 16) can be directly taken from the 5V pin of the LaunchPad using a Male-Female jumper wire.

The flyback diodes provide paths for current to flow when the drivers are switched off. Pay extra attention to the direction of the diodes when implementing them on the breadboard.

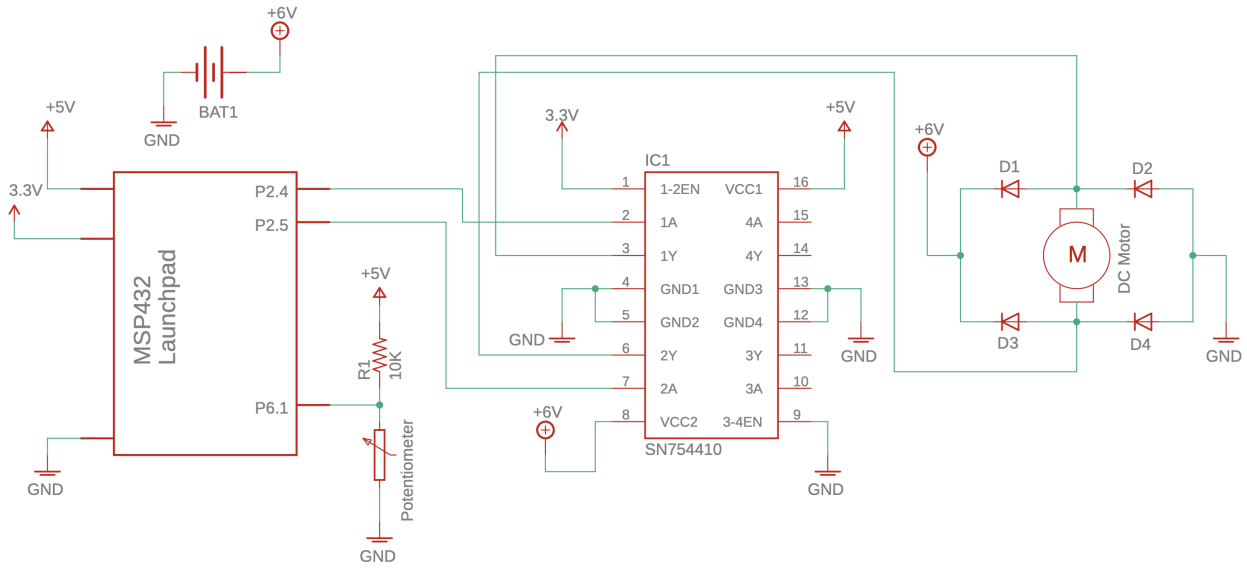


Figure 6: Circuit diagram for the H-bridge interface with motor.

As discussed in the previous section, the DC motors are connected to the SN754410 motor driver via the output pins 3 and 6. The PWM signals should be generated by `Timer_A0` channels of MSP432, specifically P2.4 and P2.5.

The enable signal is active-high, and can be directly connected to the 3V3 pin of the MSP432

LaunchPad using another Male-Female jumper wire.

Note: Remember to connect a common ground between the LaunchPad and the driver circuit.

Warning: While driving the motors using the H-bridge circuit, continue to check the temperature of the MSP432 and the SN754410 ICs. If they are getting hot, disconnect the LaunchPad immediately and debug the circuit.

8.7 Software Architecture

Note: To create a new CCS project for the MSP432P401R target device, and include the DriverLib library in the project, follow the procedure described in Lab Assignment 2. The `driverlib` folder is available on Canvas.

After disabling the watchdog timer by calling the `WDT_A_holdTimer()` function to avoid unnecessary watchdog timer time-out interrupts, initialize the P1.1 and P1.4 GPIOs (connected to the S1 and S2 switches of the LaunchPad) as inputs with pull-up resistors.

Initialize two channels of `Timer_A0` module to generate the two PWM signals required for the DC motor, one for each direction. That is, since `Timer_A0` is used for PWM generation, the channels `A0.1` (P2.4) and `A0.2` (P2.5) can be used to generate two PWM signals with different duty cycles. Make sure that `Timer_A0` channels are initialized to 0% duty cycle at the beginning.

As you learned in the lectures, after initializing the `Timer_A0` to generate the PWM signals, the duty cycle of a PWM signal can be changed by only changing the value of the `TAxCCRn` register associated to that timer channel. For example, to change the duty cycle of channel `A0.1`, we only need to change the value of `TA0CCR1` register (i.e. `x=0, n=1`).

Note: You can use a voltmeter, an oscilloscope or a logic analyzer to verify PWM signals and H-bridge driver circuit outputs.

Configure the ADC14 module `Channel 14` in the *10-bit* single channel single conversion mode. Set the clock source to *SMCLK*, and allocate the register `ADC_MEM0` to hold the results from the channel `A14`, with *non-differential* functionality. Select the `AVCC` source as the voltage reference. Make sure to set the sample timer to *manual iteration*. Do not forget to enable the ADC14 conversion at the end of the initialization.

Configure the pin P6.1 (`A14`) to its tertiary function (which is ADC), by calling the DriverLib function:

```
MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P6, GPIO_PIN1,
                                               GPIO_TERTIARY_MODULE_FUNCTION);
```

Design your code inside the main `while(1)` loop to update the ADC reading, and to detect whether the S1 and S2 switches are pressed or not via polling. Based on the ADC voltage reading from the voltage divider, adjust the duty cycle of the `Timer_A0` channels to change the motors' speed and direction, as explained in the Problem Statement in Section 8.4.

To turn the motor in one direction, set the duty cycle of one of its input channels to a level higher than 0%, and set the other duty cycle to 0%. Make sure that the SN754410 enable pin is set to high. To change the motor direction, set the duty cycle of its first input channel to 0% and the other channel to a level higher than 0%. To stop the motor, set both duty cyles to 0%.

Never set both PWM channels connected to a motor to duty cycles higher than 0% at the same time, as putting two simultaneous PWM signals through the motor in opposite directions may damage it.

Warning: A DC motor should never receive PWM signals of opposite spin directions at the same time. This could potentially cause the motor to burn out. Therefore, before you connect the motor terminals to the circuit, verify the functionality of using S1 and S2 as ON/OFF switches to ensure that the motor receives only one PWM signal at a time. Only when you are confident of the functionality of your code should you connect the terminals of the DC motor.

8.8 Expected Performance

Attach a wheel (included in the Mechatronic kit) to the motor to be able to verify the direction it is spinning at prior to doing the demos.

Press the switch S1 of the LaunchPad to start spinning the motor in one direction. Turning the knob of the potentiometer should change the speed of the DC motor (zero to maximum speed) in real time. Then press S1 to stop spinning the motor. Now, press S2 to start spinning the motor in the opposite direction. Once again, turning the potentiometer knob should allow you to vary the motor speed throughout its speed range from 0 to maximum. Then press S2 to stop spinning the motor.

8.9 Troubleshooting

The MSP432 LaunchPad or the SN754410 IC get hot:

- Double check the power (+6V and 5V) and GND wires connected to the breadboard.
- Verify the direction of flyback diodes.

Motors not do spin or get hot:

- Remove power and double check the connections.

- Recharge the batteries.
- Verify the four signals from the LaunchPad to the motor driver breadboard using a voltmeter, an oscilloscope or a logic analyzer.

A pushbutton switch does not work:

- Verify that you have initialized the GPIOs and their internal pull-up resistors correctly.
- Set breakpoints using the CCS debugger and verify that the application goes into the polling `if()` statements when pressing a button.

8.10 Requirements

1. Successfully demonstrate the outcome of the program and all the required functionalities to TAs or instructor. Make sure that you have attached a wheel to the motor to verify the direction it is spinning at prior to doing the demos.
2. Submit a commented final version of the code on Canvas.