



Creating Force Feedback using C++ programming language and Actuators

Behdad Rashidian

Design Team 5-Smart Voting Joystick for Accessible Voting Machines

April 4, 2013

Facilitator: Dr. John Deller

Abstract

The purpose of this application note is to aid the user create the desired force feedback to different axis of a joystick. This document contains material about programming the force feedback and how the actuators will work based on the code that is provided and the H-Bridge based circuit designed for the motors. Actuator motors will be discussed in detail to give a better perspective of the design of the force feedback. This note also includes relevant schematics and figures to provide the user with visual references to the mentioned components to minimize misinterpretation.

Keywords

Actuators, C# programming, Motors, Force Feedback, Axis, Joystick, DirectX, DirectInput, H-Bridge circuit

Introduction

Many varieties of input devices and control interfaces have been developed for powered joystick to satisfy diverse needs for disabled people. However, for some people with severe motor disabilities, it is impossible to use these powered joysticks since they do not provide the desired force feedback. The force feedback is needed in order to control the range of joystick movements. Moving through different selections in a webpage needs Constant type of force feedback so it can help the user with motor disabilities to move easier among selections.

In this application note we will discuss developing a program which allows the user to have this constant force with the DirectX library that will be discussed.

H-Bridge circuit introduction

H bridge circuit enables a voltage to be applied across a load in either direction. We need to use such circuit for our force feedback motor to allow DC motors to run forwards and backwards. An H-bridge circuit is typically constructed using opposite polarity devices, such as PNP transistors or P-channel MOSFETs connected to the high voltage bus and NPN BJTs or N-channel MOSFETs connected to the low voltage bus.

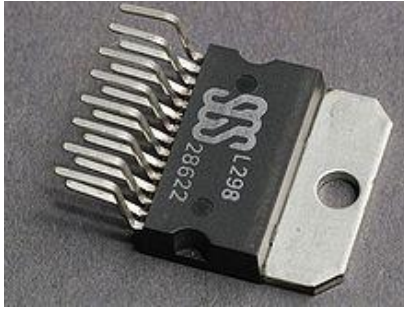


Figure 1 - Dual H-bridge motor Driver

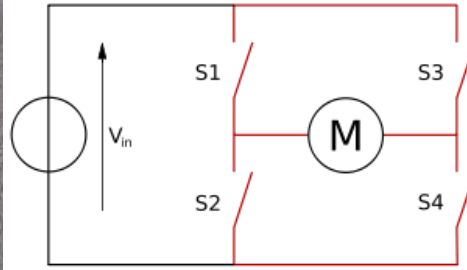


Figure 2 - Structure of H-Bridge

There are different conditions that we have to consider for the H-Bridge circuit. When the switches S1 and S4 (shown in figure 2) are closed and S2 and S3 are open, we will have a positive voltage applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed. This will let us have reverse operation of the motor. The switches S1 and S2 should never be closed at the same time. It will cause a short circuit on the input voltage source. The same applies to the switches S3 and S4.

The following diagram summarizes the operation of the H-Bridge:

S1	S2	S3	S4	Result
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor free runs
0	1	0	1	Motor brakes
1	0	1	0	Motor brakes
1	1	0	0	Shoot-through
0	0	1	1	Shoot-through
1	1	1	1	Shoot-through

Figure 3 - H-Bridge Switches

Actuator Motor Power Supply

In order to choose the correct power supply, the amount of current and power required must be known. The force that raises the pin in the motor is directly related to the current flowing through the actuator motor. The AC/DC supply delivers 24Vdc to the main printed circuit board. The CPU receives the USB signals, checks the position of the X, Y, rudder and Throttle potentiometers, 12-switches, and controls the motor current.

The motor current control and power stages are all separate. Motor current is around $\pm 0.7A$ max, with 8.5 ohm motors. We could increase the current up to 1.4A. The circuits really control the current through the motors instead of just adjusting the PWM duty-cycle. Therefore, the motor current is precisely controlled, and not dependent on the motor internal resistance.

Actuator motor design

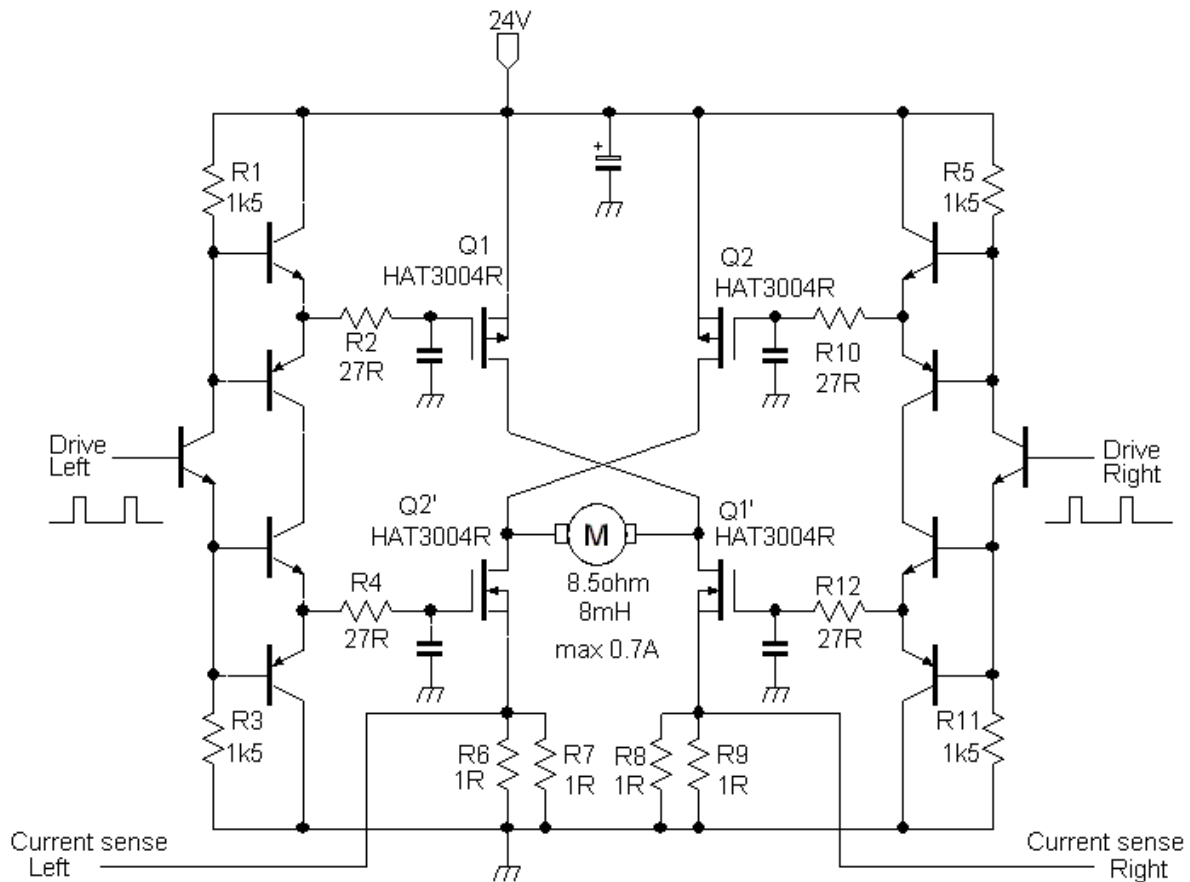


Figure 4 - Actuator motor circuit

The above schematic is an H-Bridge based circuit which is introduced earlier. This design will help us reach our goal of making force feedback for the joystick.

We need to have 6 NPN BJT transistors that can act as a switch that will cut off current when the output is 0 V and switch on the external power supply when the output is 5 V. For this application the transistor will operate in the active region when the microcontroller output is 5 V, and it will operate in the cutoff region when the microcontroller output is 0 V. The transistor model being used is the 2N2222A NPN BJT.

We also need to use 4 PNP BJT transistors that are sourcing the current. The PNP transistors have their emitter connected to the resistor and collector to the ground. Base is connected to the 1K Ω resistor and the emitter of the NPN BJT with the right-drive base.

In the schematic with MOSFETs switching that has conduction from GND(0V) to 24V via 2 MOSFETs, division of current can easily occur if there is a small delay in one of the MOSFET drives. We need to only have one side of the bridge active at the time for one motor current direction, the other side will be fully switched off, and the inductive current will flow through the other side MOSFET's body diodes. Disadvantage of this method is that the power dissipation in the MOSFET that is not active is actually quite high. This happens due to the higher voltage drop across the body diodes.

The voltages across the current sense resistors are processed with OPAMPs and work as the feedback of the force current loop. There are current sense resistors in each side of the bridge. By reducing the current sense resistors R7 and R9, the system will increase the voltage across the motors to keep the same feedback voltage, thus the current through the motors will increase.

Software Design of the Force Feedback

The programming of force feedback is done in C++ language. We will be using DirectX library in order use different functions necessary for force feedback.

DirectX's DirectInput component provides access to joysticks and other human interface devices.

DirectInput focuses on force feedback functionality. It is composed of three objects: DirectInput, DirectInputDevice, and DirectInputEffect.

DirectInput: Captures high-level DirectInput functionality, and allows you to count devices and create DirectInputDevice objects.

DirectInputDevice: It will be used as Interface to a input device, such as a joystick. It allows us to gather state information about the device, set information for the device, and create DirectInputEffect(force feedback) objects.

DirectInputEffect: Captures a single effect that will be used on a force feedback device. Provides functions to start, stop, and setup force feedback effects such as gain, inertia, friction, magnitude and etc.

DirectX provides a helper function called DirectInputCreate, that initialize a DirectInput object. It's defined in the DINPUT.H header file along with all of DirectInput's function. The actual function is located in the DINPUT.LIB file. DirectInputCreate is defined as follow:

```
HRESULT WINAPI DirectInputCreate(  
    HINSTANCE hinst,  
    DWORD dwVersion,  
    LPDIRECTINPUT * lplpDirectInput,  
    LPUNKNOWN punkOuter  
);
```

After we setup the DirectInputCreate, we need to create member variables to setup the parameters for the effect that we want for our joystick. In our particular joystick we need force feedback for 2 axis to be Constant.

There are two methods to relay physical force to an input device. The first method is to assign a specific effect that is supported by the device. These effects have parameters that can be configured and assigned to axes on the device. The other method is to download a force feedback effect file to the device. This file contains a set of effects that can be run at any time once they are downloaded into the input device. (The first method is used and is demonstrated below)

There are different parameters that should be assigned while we try to create the constant force feedback. The following code will describe how we will assign effects to both X and Y axis of the joystick. First we create a member variable of type HRESULT that we initialized above. Next we receive pointer to create the effect and we create member variable of the class called DirectInputEffect called diEffect so we can set the attributes (variables) of the effect class to the joystick. We need to define the axis to be an array.

```

HRESULT hr;
LPDIRECTINPUTEFFECT lpdiEffect; // receives pointer to created effect
DIEFFECT diEffect; // parameters for created effect
DWORD dwAxes[2] = { DIJOFS_X, DIJOFS_Y };
LONG lDirection[2] = { 180 * DI_DEGREES, 0 };
DICONSTANTFORCE diConstantForce;

diConstantForce.lMagnitude = DI_FFNOMINALMAX; // Full force
diEffect.dwSize = sizeof(DIEFFECT);
diEffect.dwFlags = DIEFF_POLAR | DIEFF_OBJECTOFFSETS;
diEffect.dwDuration = (int)DI.Infinite; // Effect duration
diEffect.dwSamplePeriod = 0; // = default
diEffect.dwGain = DI_FFNOMINALMAX; // No scaling
diEffect.dwTriggerButton = DIEB_NOTRIGGER; // Not a button response
diEffect.dwTriggerRepeatInterval = 0; // Not applicable
diEffect.cAxes = 2;
diEffect.rgdwAxes = &dwAxes[0];
diEffect.rglDirection = &lDirection[0];
diEffect.lpEnvelope = NULL;
diEffect.cbTypeSpecificParams = sizeof(DICONSTANTFORCE);
diEffect.lpvTypeSpecificParams = &diConstantForce;

hr = g_lpdi->CreateEffect(GUID_ConstantForce,
    &diEffect,
    &lpdiEffect,
    NULL);

```

The direction of the effect is set to be 180 degrees. We need to have full force magnitude and duration of infinite since we do not want any interrupt in the force feedback effect. Gain is set to be maximum possible. The DI_FFNOMINALMAX is the same 10000. The gain of 10000 indicates that magnitudes are not to be scaled. For example 6100 means that forces are to be scaled to 61% of their nominal magnitude. Moreover we need to set the effectType to ConstantForce which is declared in the DirectInput library. Finally “hr” which is a member variable of HRESULT needs to be set with all the parameters that have been established by diEffect.

Throwing an exception is also useful to see if the object has been filled with the required parameters.

This could be done by checking if the member variable is empty or not.

```

if (hr == null)
{
    throw new Exception("ConstantForce is not supported.");
}

```

Conclusion

In this application note, circuit design and software design of the force feedback of the joystick was discussed in detail. We first discussed the H-Bridge circuit which has been used as the base circuit for the design of the constant force feedback. Moreover switches of the H-Bridge circuit were explained and how they make the motor to operate in reverse order. Furthermore the actual design of the force feedback was demonstrated and details about the current flowing, voltage and components used in the design were

explained. At last software design of the force feedback was discussed in depth and how we set different attributes to the required effect of the joystick.

References

- 1- http://en.wikipedia.org/wiki/H_bridge
- 2- [http://msdn.microsoft.com/en-us/library/windows/desktop/ee418273\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ee418273(v=vs.85).aspx)