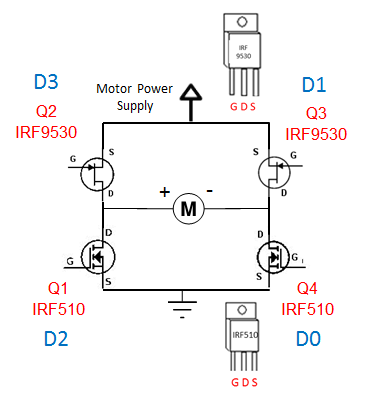
**Project**

**H-Bridge and Keypad**

An H-Bridge is used to control a motor’s rotational direction. An H-bridge can be constructed with 2 N-channel and 2 P-channel transistors. Each transistor can be turned on/off with a digital line. As such, the four digital output lines from the PCF8574 will be used to control the motor’s state.

The Lego NXT Brick only has 4 keys. A 12-key keypad can be easily interfaced to a PCF8574. Such keypads use a row/column matrix membrane. The keys 0 to 9, \* and # have a specific row and column position. Thus 4 digital lines can be used to detect the row, and 3 digital lines can be used to detect the column.

**H-Bridge Circuit and the PCF8574**



**Fig. 1A:** Classic H-bridge using two N-channel (IRF510) and two P-channel (IRF9530) MOSFETs

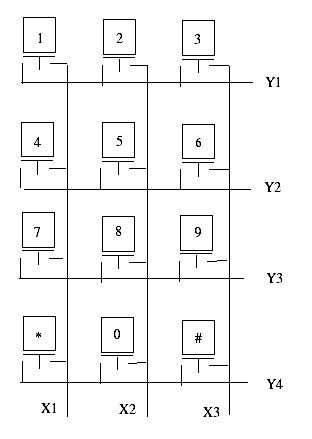
From a recent lab, one knows that HI or LO voltages on a MOSFET’s gate will switch the transistor on or off. The PCF8574 can provide four digital lines D0, D1, D2, and D4. Connecting these lines to the gates of the four transistors Q1, Q2, Q3, and Q4 will turn the motor clockwise or counter-clockwise, stop the motor, or allow the motor to freely rotate.

**12-key Keypad and the PCF8574**

Surplus keypads rarely come with pinout specifications. However, they are easy to decode with an ohmmeter **Fig. 2A** (right). The first step is to clip the ohmmeter probes to two different pads[[1]](#footnote-1) on the keypad. The second step is to observe the ohmmeter and press keys until you find the key that returns an almost 0 Ohm resistance. The resistance is almost zero because there is a row and column wire associated with each key (see **Fig. 2A** left). Write down this key's row and column position and continue doing this two-step procedure for all pads on the keypad. You should start seeing a pattern. Once all keys on the keypad have been assigned their row and column position, you can then decipher which pad corresponds to which column, and which pad corresponds to which row. Once you have determined X1-X4 and Y1-Y3 (for a 12-key touch-tone phone like keypad), you can then wire them to the PCF8574 digital lines D0-D3 for rows Y1-Y4 and digital lines D4-D6 for columns X1-X3.

**Fig. 2A:** Standard 12-key keypad requires an ohmmeter (right) to determine which row and column are shorted when the specific key is closed (left).





**Sample code:**

// FILE: keypad1\_2.nxc - Works!

// AUTH: P.Oh

// DESC: Detect keys 1, 4, 7, and \*

#define I2Cport S1 // Port number

#define I2CAddr8574 0x70 // I2C address x040 8574 or 0x70 for 8574A

// Global variables

// array variables (since NXC's I2C functions take array variables

byte WriteBuf[] = {I2CAddr8574, 0x00}; // sets up PCF8574A for writing

byte ReadBuf[]; // data received from PCF8574A. We won't be reading any data but we need this for I2CBytes

int RdCnt = 1; // number of bytes to read

long GetKey() {

WriteBuf[1] = 0xEF; // Col 1: 0xEF = 239 decimal = 1110 1111. Sets Col 1 HI

I2CBytes(I2Cport, WriteBuf, RdCnt, ReadBuf);

if(ReadBuf[0]==0xEE) return(1); // Row 1; "1" key. 0xEE = 238 decimal = 1110 1110. If HI, then "1" key

if(ReadBuf[0]==0xED) return(4); // Row 2: "4" key

if(ReadBuf[0]==0xEB) return(7); // Row 3; "7" key

if(ReadBuf[0]==0xE7) return(14); // Row 4; "\*" key

return(-1); // return -1 if no key pressed

} // end of GetKey

task main() {

long key;

long value = 0;

SetSensorLowspeed (I2Cport); // PCF8574A connect to NXT on S1

NumOut(0,0,value,true); // show initial value

while(true) { // endless loop

while(GetKey() < 0); // do nothing

key = GetKey(); // get key value

PlayTone(1000,5); // make key pressed sound

switch(key) { // case based on key value

case -1: // no key, do nothing

break;

case 15: // #: so negate value

value = -value;

break;

case 14: // \*: quit

Stop(true);

break;

default: // 0-9: just build up number

if(value >= 0) value = value\*10 + key;

else value = value\*10 - key;

break;

} // end switch

NumOut(0,0,value, true); // show current value

while(GetKey() >= 0); // wait for key to be unpressed

} // end of while

} // end main

**Project:** This project is worth 15% of your final grade. Partner up as pairs. One student shall construct and program the H-bridge, while the other constructs and programs the 12-key keypad.

**Objective:** Using a single Brick (with perhaps 2 NXT breadboard adapters) have the motor perform the following:

|  |  |
| --- | --- |
| Key | Motor Function |
| 4 | Rotate CW |
| 6 | Rotate CCW |
| 8 | Brake |
| 2 | Free spinning |

Grading rubric will be presented in another document

Code Explanation: Much like dioOutput2\_0.nxc from a previous lab, dioRelay1\_0.nxc configures the PCF8574’s eight digital lines for output. A do-while loop continuous reads button presses. Pushing the right arrow button (rightArrowButtonPushed) closes the switch and turns the motor on and vice-versa with the left arrow button. The program aborts when the grey button is pressed.

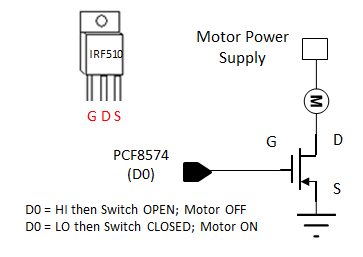
**Exercise 1:** In NxC create programs for the following:

* 1. Replace the DC motor with a buzzer so that closing the relay’s switch results in a sound
  2. Modify your code and wiring to turn the motor on using digital line D1.

**Concept 2 – Concept 1 – NXT and Transistors**

Like a relay, a transistor is a switch. Unlike a relay, transistors do not have any mechanical parts. As such, they are capable of switching much quicker than relays, and have longer mean-time-between-failures (MTBF). Popular NPN transistors include the TIP31 and IRF510 which both come in TO-220 packages. The TIP31 is a current-driven transistor, whereas the IRF510 is a voltage-driven one. Since the PCF8574’s source current is low, turning on or off a current-hungry device like a motor is best done using the IRF510. The IRF510 is called a MOSFET and allows current to flow when its gate pin is above a certain voltage (about 5 Volts). One could employ a TIP31, but then a current-to-voltage converter would be needed.

Step 1: Breadboard the schematic in **Fig. 2A**, noting the IRF510’s pin labels (**Fig. 2B**)



**Fig. 2A:** Interfacing a IRF510 MOSFET to NXT using one of the PCF8574’s digital output lines.



**Fig. 2B:** The IRF510 MOSFET can simply insert into the solderless breadboard.

**Fig. 2A** does not use any diodes (and capacitors) in contrast to **Fig. 1A**. While the circuit in **Fig. 2A** will work, the back-EMF from the motor could kick-back enough current to damage the digital line. The analogy is back-wash in hydraulics. A capacitor across the motor leads and a diode will prevent such back-wash to occur.

**Step 2:** Write and execute an NxC program called dioMosfet1\_0.nxc

Much like dioOutput2\_0.nxc and dioRelay1\_0.nxc, one can set digital line D0 HI (+5V) or LO (GND) by setting decimalNumber to 1 or 0 respectively.

When the right arrow button is pressed, have the motor turn on and vice-versa with the left arrow button. Use the grey button to abort the program.

**Exercise 2:**

2-1: Refer to dioInput2\_0.nxc and dioDipLed2\_0.nxc from a previous lab. Write a program that reads DIP switch positions, and when the value is one, turn on the motor. For any other DIP switch value, the motor turns off. You can use either a reed relay or IRF510 circuit.

1. Some keypads have pads (for soldering wires to) or come with header pins [↑](#footnote-ref-1)