

# ONE BUTTON AND TWO BLINKY LIGHTS

## 1.1 Objectives

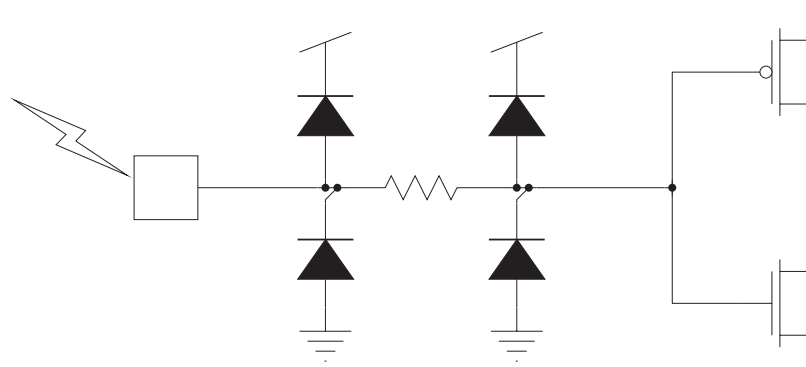
In this lab, you will construct a very basic device from a PIC18F2455 microcontroller. In the process, you should become familiar with the process of programming PICs, with the basic support circuitry required to support their operation, and with the process of interfacing buttons and light emitting diodes (LEDs) to general-purpose digital input/output (I/O) pins on the PIC.

## 1.2 General Facts about CMOS Chips

In this course, you will be working with a lot of *complementary metal-oxide-semiconductor* (CMOS) chips, which are packaged in DIP packages. CMOS chips are generally *very* sensitive to *electrostatic discharge* (ESD), which you have probably experienced first-hand by dragging your feet across a carpet in the winter and getting a shock when you touch a light switch or another person. This section provides some basic guidelines for safe handling of CMOS chips.

### 1.2.1 Handling CMOS Chips: Preventing Electrostatic Discharge

CMOS chips are *extremely* prone to damage by static electricity. The breakdown field of an insulator is the electric field above which the atoms that constitute it ionize, permitting a direct path for electrons to flow through it. For example, when lightning strikes, so much of a potential difference has built up between the clouds and the earth, that the electric field



**Figure 1.1:** Electrostatic discharge (ESD) protection circuitry.

exceeds the break-down field of air, the air molecules ionize, and there is a direct current path between the clouds and the earth (i.e., the lightning bolt). The current flowing through the channel of an MOS transistor is controlled by an insulated gate. The layer of silicon dioxide ( $\text{SiO}_2$ ) between the gate electrode and the silicon substrate is *very* thin—for the chip that you will be using in this lab, the gate oxide is probably between 30 and 50 nm (i.e.,  $30 \times 10^{-9}$  m and  $50 \times 10^{-9}$  m) thick. Even a few tens of volts dropped across such a short distance amounts to a *huge* electric field in the  $\text{SiO}_2$ , approaching its breakdown field of about  $10^9$  V/m, causing the gate oxides to blow out. During a short walk across the lab, you can easily build up a few kilovolts (i.e., a few *thousands* of volts) of static electricity on your body. If you touch the pins of the chip, you can easily blow out the delicate gate oxides.

There are typically some ESD protection structures on CMOS chips, such as that shown in Fig. 1.1, comprising diode clamps to the power-supply rails. Normally, the voltages that we apply to the pins on the chip fall between the two power supplies (i.e., ground and  $V_{DD}$ ). If we try to take the voltage on a pin with ESD protection, one or more of these diodes will forward bias, and take the current that we are supplying to move the voltage on the pin outside of the power rails. If we were to zap the chip with some static electricity, these diodes would eat some of it, but more often than not, these ESD protection structures are often just not enough to save the gate oxides from getting popped. There are some very simple precautions that you can take so your chips will survive from one lab to the next:

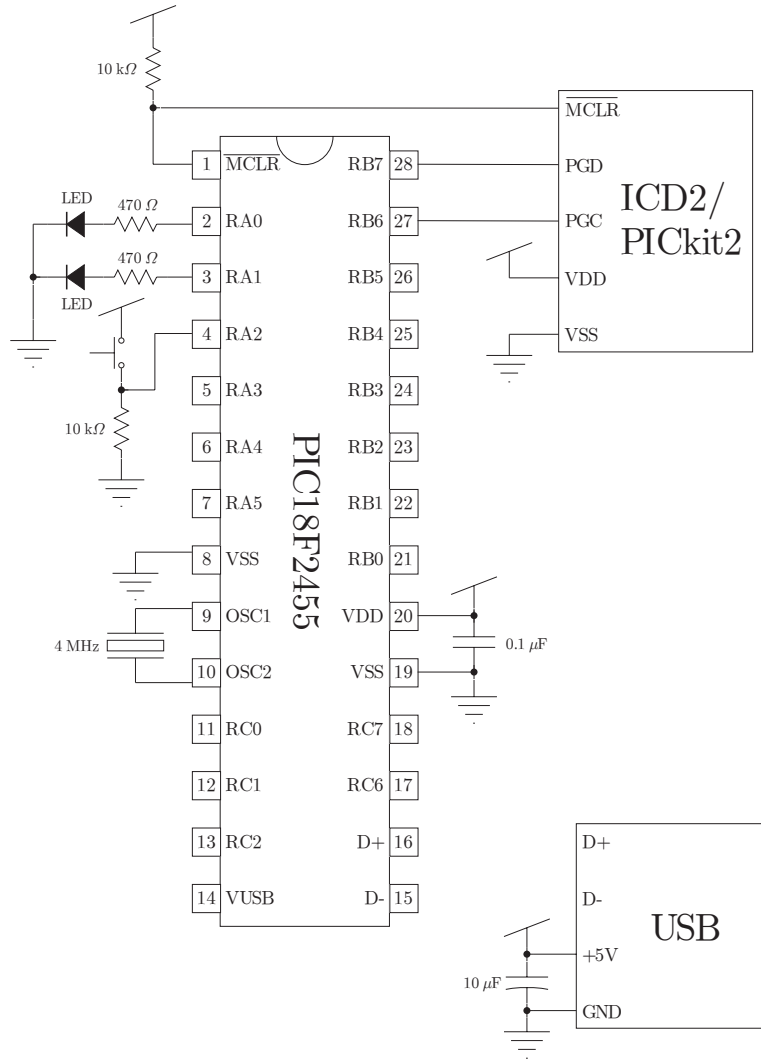
1. When the chip is not powered up in the breadboard, keep it stuck into a black piece of conductive foam. The conductive foam keeps all of the pins nearly shorted together, preventing substantial voltage difference between the pins.
2. Always discharge yourself to chassis ground before picking up or touching a chip. Any charge that has built up on you will then go to ground instead of into your chip. Between discharging yourself and picking up the chip, don't move your feet.
3. Always try to handle chips by their package, not by the pins.

PIC microcontrollers have a reputation for being pretty robust, but please try to be careful with them.

### 1.3 Building the Base Device

Obtain the installers for the MPLAB v7.50 integrated development environment (IDE), for the student version of the MPLAB C18 v3.02 C compiler, and for the PICkit 2 v2.10 Programmer/Debugger application from the Microchip web site ([www.microchip.com](http://www.microchip.com)) and install all three packages on your laptop. Then, download the `lab1.zip` source code archive from the course web site ([ece.olin.edu/poe](http://ece.olin.edu/poe)) and unzip it into a new folder. This archive contains two files: `lab1.c` and `18f2455.lkr`. The `lab1.c` file is the source code for the base firmware, which is written in C, for the device that you will be constructing initially. The `18f2455.lkr` file is a linker script that tells MPLINK how to map the firmware onto the microcontroller's memory.

Figure 1.2 shows the circuitry required to implement the firmware contained in the `lab1.c` source file. Construct this circuit neatly in a solderless breadboard. Note that, in this lab,



**Figure 1.2:** PIC18F2455 core circuitry.

you will be using the +5V power supply on a USB port on your laptop, but the device will not enumerate as a USB peripheral. The 4.000 MHz crystal oscillator provide a stable frequency reference from which the PIC derives its 6-MHz clock internally. The 470-Ω resistor in series with each LED serves to set the current flowing through the LED when the PIC's I/O pin is set high. The resistor and LED are connected in series, which means that the same current will flow through each of them. Depending on the type of material that the diode is made of, there will be a characteristic voltage drop across it for a typical current of a few milliamperes. For signal diodes made of silicon, this voltage is typically about 0.7 V. For LEDs, which are made from a compound semiconductor, this voltage drop is approximately 2 V. Like all diodes, LEDs have a *very* steep current–voltage characteristic—the current typically increases by a factor of ten for every 60-mV change in the voltage across the diode. Consequently, even if the current differs markedly from the nominal milliamperere current level, the voltage drop will remain roughly the same. So, to set the current at some desired

level, we would compute the voltage that should appear across the resistor as  $V_{DD}$  (5 V) minus the drop across the LED (about 2 V) and divide by the desired current, and select a resistor value close to the one that we computed. The  $470\ \Omega$  resistor sets the current to a little more than 6 mA. The  $10\ \text{k}\Omega$  pull-down resistor connected between the pushbutton switch and ground serves to restore the nominal low voltage on the RA2 when the pushbutton is not being pressed. Likewise, the  $10\ \text{k}\Omega$  pull-up resistor on the  $\overline{\text{MCLR}}$  (i.e., master clear), serves to restore the nominal high voltage when the PICkit 2 is not connected or is not holding the PIC in a reset state.

Launch MPLAB and create a new project (Project > New...) and select the Microchip C18 Toolsuite (Project > Select Language Toolsuite...). For each of the components (i.e., `mpasmwin.exe`, `mplink.exe`, `mplib.exe`, and `mcc18`) in the toolsuite, you should make sure that MPLAB is using the executables that came with the C18 compiler. Add `lab1.c` under the “Source Files” heading and `18f2455.lkr` under the “Linker Scripts” heading in the project window. Next, under the “General” tab of the “Build Options” dialog (Project > Build Options... > Project) for your project, set the Include Path to point to `C:\MCC18\h\` and set the Library Path to point to `C:\MCC18\lib\`. Next using the Configure menu, select PIC18F2455 as the device that you will be using. Build the project (Project > Build All), obtain a PICkit 2 programmer/debugger, and program your PIC18F2455 with the firmware that you just built. When you first plug the PICkit 2 into a USB port, it will enumerate as an HID class device (the operating system should find appropriate drivers automatically). When you launch the PICkit 2 application, the status bar will report the version of the operating system firmware version loaded in the nonvolatile memory of the PICkit 2. The status bar will probably tell you to “Upgrade the OS firmware to X.XX.XX or greater.” To do so, select Tools > Download PICkit 2 OS Firmware. The application will open a file browser dialog. Browse to `C:\Program Files\Microchip\PICkit 2 v2\` and select the file `PK2V020100.hex`. The application will then perform the OS firmware upgrade. To program your device, select Device Family > PIC18F, connect the RJ-11 plug into the ICD2/PICkit2 connector in your protoboard, select Programmer > Read Device, select File > Import File, select the `lab1.hex` file that you should find in your working project directory, and select Programmer > Write Device. Once the device is programmed, you should see one LED on steadily and one blinking about once per second. When you push the button, the LED that is on steadily should turn off while the other one keeps on blinking.

## 1.4 Modifying the Device

Now, you will need to modify the firmware to change the functionality of the device. Modify the firmware so that the two LEDs both blink out of phase with each other (i.e., when one is on the other is off and vice versa) and so that, when you push the button, the LEDs maintain their present state until the button is released.