CSE 3442/5442: Embedded Systems 1

Lab 7: Building a PIC18F4520 Standalone Alarm System with EUSART Communication

ABET Outcome E:
The ability to identify, formulate, and solve engineering problems

Table of Contents

1 Overview

The Computer Engineering Program at the University of Texas at Arlington is ABET accredited within the EAC Commission. One of the student outcomes (Outcome E) requires students to have: "The ability to design a system, component, or process to meet desired needs." This lab is going to assess student's capabilities towards this outcome. Failure to satisfactorily complete this assignment results in an overall unsatisfactory grade (F) as well as failure of the assessment.

The purpose of this lab assignment is to plan, setup, build, and configure a PIC18F4520 chip as a standalone system (not using the QwikFlash board).

Students will have to design and implement a hardware/firmware "Alarm System" that has bi-directional communication between a PIC18F4520 and a PC. The system must also save previous settings and keep their most recent state even if the PIC loses power or is reset through the $\overline{MCLR}$ pin.

2 General Description

This section provides information about which aspects and features of the PIC should be used and details about the important components/devices used in conjunction with the PIC.

Students will need to have a great understanding of each component in the system. It is good practice to break-down individual parts of the lab to test each device/component individually before combining and using with others. For example, when developing the communication between the PIC and PC, first just use a terminal program on the PC (such as Tera Term) to send simple messages/characters to ensure everything is connected and wired-up correctly.

2.1 Topics and PIC features used in this lab

1. EUSART – Enhanced Universal Synchronous Asynchronous Receiver Transmitter
   a. Receiving/Transmitting on both the PIC and PC (bi-directional) in UART mode
2. Analog-to-Digital Conversion
3. Timers
4. Interrupts – External, Timer, A/D
5. $\overline{MCLR}$ Reset
6. External Crystal Oscillator clock source
7. Digital Input and Output
8. ICSP – In-Circuit Serial Programming using the PICkit 3

Note: Some register or module details for the PIC18F4520 might be slightly different from the PIC18F452 so read the datasheet carefully and look at "Migration from 452 to 4520.pdf" in the *Important Files* folder.

2.2 List of components, parts, and devices used in the system

#	Component	Quantity	Purpose/Function	Link
1	**PIC18F4520-E/P** (Microcontroller)	1	Microcontroller that provides main functionality of the Alarm System	PIC
2	**PICkit 3** (Programmer/Debugger)	1	Used to program the PIC using ICSP and debugging the PIC's code	PICkit 3
3	**USB-to-Serial Chip** (FT232RL)	1	Communication interface adapter between the PIC and PC	USBtoSerial
4	**Keypad** (4x4 Universal Switch)	1	Additional input method to the PIC	Keypad
5	**PIR Motion Sensor**	1	Used for triggering the Motion Alarm	PIR
6	**Temperature Sensor** (TMP36)	1	Used for getting the temperature and triggering the Temperature Threshold Alarm	Temp
7	**20 MHz Crystal**	1	To run the PIC at a higher speed than the internal oscillator's 8MHz frequency	Crystal
8	**Push Button**	1	Used with the $\overline{MCLR}$ pin for resetting the PIC	PB Switch
9	**LEDs**	4 (Blue, Yellow, Red, Green)	For indicating various system statuses	LED
10	**Resistors**	10 (2-8.2kΩ, 4-150Ω, 4-10kΩ)	For getting voltages to the correct level for various components	-
11	**Capacitors**	3 (2-18pF, 1-0.1µF)	Used with the external Crystal Oscillator and Temperature Sensor	-
12	**USB Cable**	2	To connect the USB-to-Serial Chip and the PICkit3 to the PC	-
13	**Headers**	-	For connecting devices with breadboard/PIC	-
14	**Personal Computer**	1	Provides an additional interface to the Alarm System. You may use a lab computer or your own to develop, but your system MUST be able to run and demo on a lab computer.	-
15	**Breadboard**	1	Connections for all components and electronics	-
16	**Wires**	1 box	Connections between devices (you may also buy your own wire or cut/strip more wire from the lab)	-
17	**Pencil Box**	1	Used to hold all parts	-
18	**Virtual COM Port Driver**	1 .exe	Go to the link on the right to install a driver that is required to use the USB-to-Serial chip on your own personal computer	VCP Driver

*The links to certain component's websites are included above

*Other relevant files and datasheets are included in the *"Important Files"* lab folder

*You may take all of these components with you outside of lab but you MUST return all components when the lab/course is completed

2.3 Detailed Information of Each Major Component/Device

2.3.1 PICKit 3

Cable connection type: USB-A to USB-Mini B (red)

Before creating the Alarm System, first you will need to run a simple program to practice programming the PIC using ICSP. Write a simple program that continuously toggles a LED on the IDL-800 every second using *Delay10KTCYx(200)* with the PIC18F4520's internal 8MHz oscillator.

Using the breadboard on the IDL, insert the PIC and make the following connections for ISCP. Make sure the IDL is powered OFF.

PIC	IDL-800	PIC Kit 3
$\overline{MCLR}$ / V_{PP}	-	PIN 1
V_{DD} (both pins)	+5V	PIN 2
V_{SS} (both pins)	GND	PIN 3
PGD	-	PIN 4
PGC	-	PIN 5
-	-	PIN 6 (not used)
Any digital output pin	Any LED	-

Using the starter code, create a new project and set the following project settings:
Device: PIC18F4520 **Hardware Tool**: PICkit3 **Compiler Toolchain**: C18

Go to *Window > PIC Memory Views > Configuration Bits*

Make the following changes if they are not already…

Field	Option	Setting
OSC	INTIO67	Internal oscillator block, port function on RA6 and RA7
WDT	OFF	WDT disabled (control is placed on the SWDTEN bit)
LVP	OFF	Single-Supply ICSP disabled
MCLRE	OFF	RE3 input pin enabled; MCLR disabled

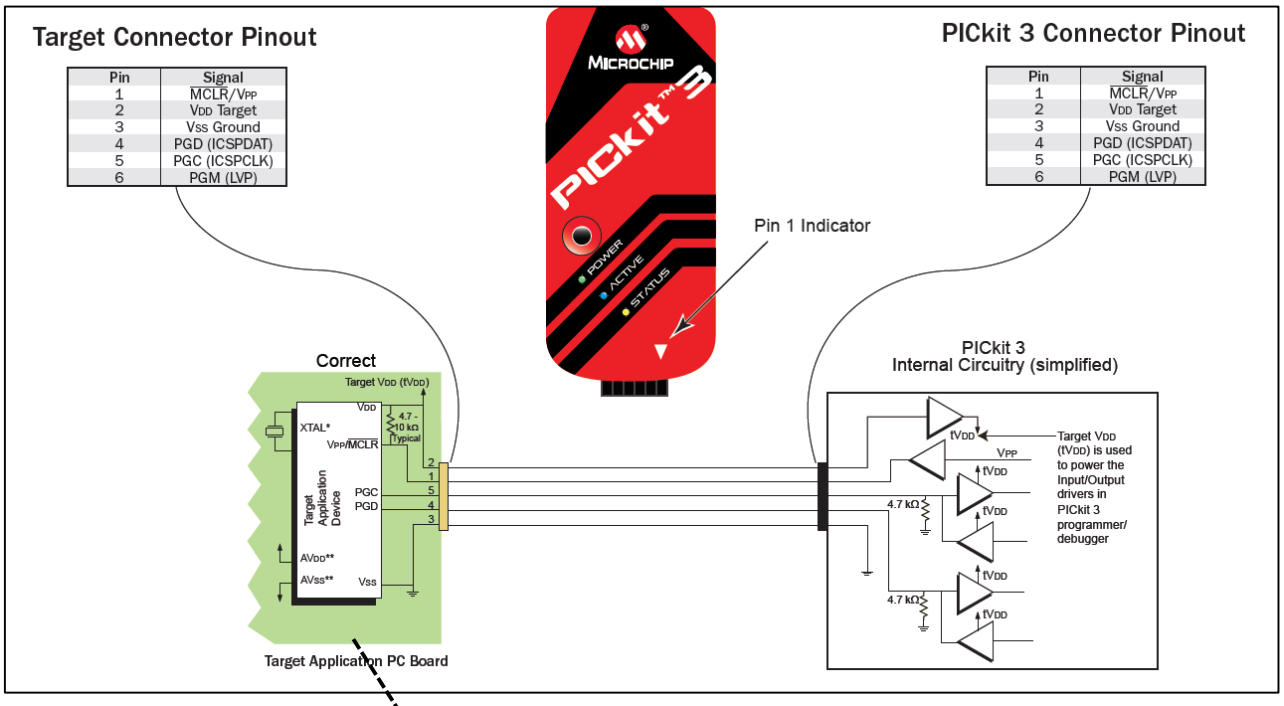
Click *Generate Source Code to Output*

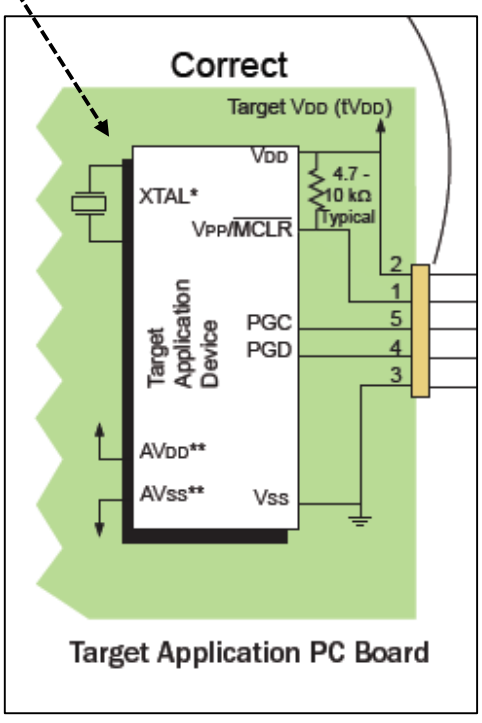
Copy/paste the generated code into your file immediately after your #includes.

You can remove the extra *#include <p18F4520.h>* copied in.

To program click *Make and Program Device Main Project* (next to clean & build)

*Do not forget to include the resistor (8.2kΩ) between the V_{DD} and $\overline{MCLR}$
*When programming the PIC, have the IDL's power ON
*If MPLAB does not successfully program the PIC and gives you a *Target Device Not Detected* error message, if you are using a laptop, make sure it is plugged into the wall and isn't running off a low battery because the PICkit/USB needs a particular voltage level





2.3.2 USB-to-Serial Chip (FT232RL)

Cable connection type: USB-A to USB-Micro B (black)

This chip is essentially an adapter that turns USB into UART (RS232-like serial) and thus allows data being transferred in UART mode between the PIC and PC.

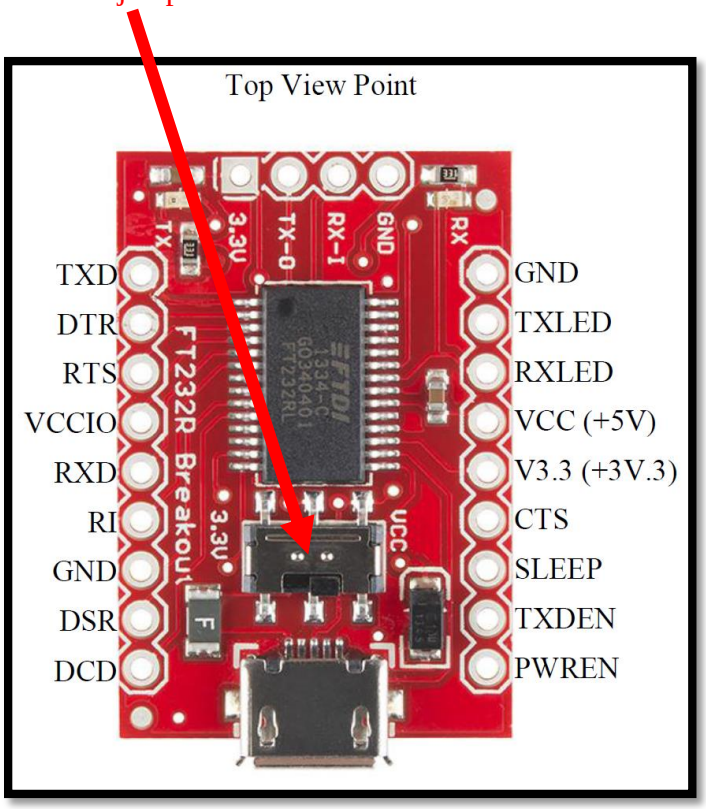
PIC18F4520←→(via EUSART/RX/TX)←→FT232RL←→(via USB)←→PC

There are only 4 pins on this breakout board you need to use:
VCC: +5V (this will supply power for all components on the breadboard, taking power from your computer's USB)
GND: Common ground (common)
RXD: For receiving data from the PIC18F4520
TXD: For transferring data to the PIC18F4520

You must install this Virtual COM Port driver in order for your computer to recognize the chip. Download at VCP Driver (from section 2.2).

Note: *When using the keyboard/terminal on the computer, if you only want to show text on the terminal's screen that comes from the PIC, disable your terminal's echo. In Tera Term *Setup > Terminal > Local Echo (uncheck)*

*Make sure the jumper switch is set to VCC on the board



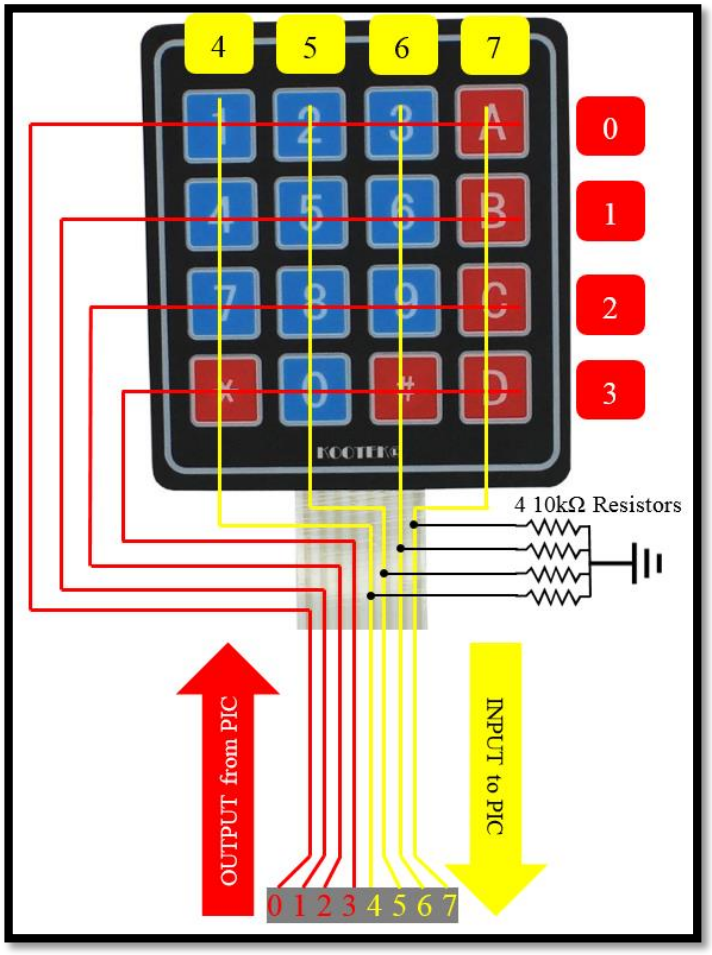
2.3.3 4x4 Matrix Keypad

Each of the 16 keys is a simple push button that when pressed creates a connection between the row pins/wires (0-3) and the column pins/wires (4-7).

The general scheme for determining which button was pressed is to set the row pins (0-3) HIGH one-by-one and read each of the column pins (4-7) one-by-one to see which column pin reads HIGH, but you may use another way if you like.

You need to pull-down the 4 input connections to common using resistors so they read as logic LOW when nothing is being pressed (this way you ensure that no pins are floating and thus cannot generate a false positive key press when no keys were actually pressed).

Remember, the PIC does not know what each wire/pin represents (1, #, A, etc.). You must interpret what each connection should be translated to in your code. Interfacing with push buttons can be tricky, as the system should not see repeated key presses when a user has only pushed the button once. Similarly, it should see a single key press no matter how short the user has pushed the button.

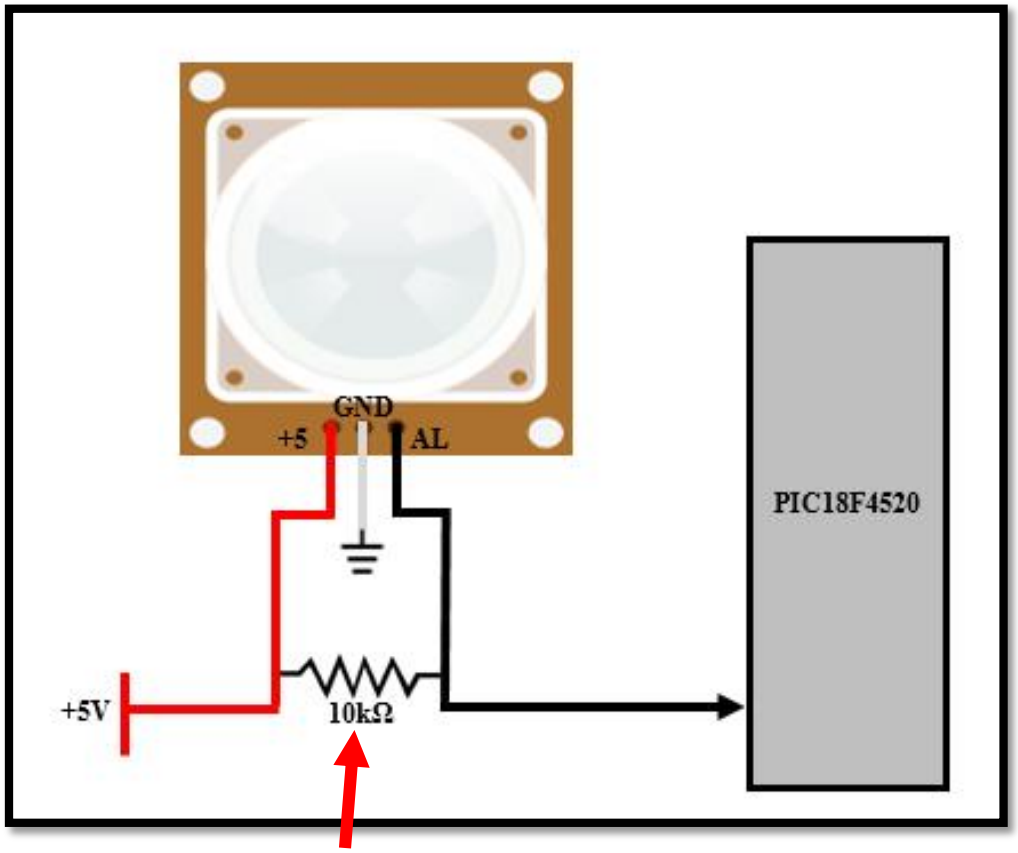


2.3.4 PIR Motion Sensor

This 3-pin passive infrared sensor is used to detect motion by measuring the amount of IR radiation/reflection (heat radiation) from the area/objects it can see. If the IR radiation changes significantly, an IR sensor triggers an alarm. The IR sensor we are using requires about two seconds of startup time before it can be properly used.

The 3 pins of the motion sensor are:
RED (+5V)
WHITE (GND/common)
BLACK (AL) Alarm pin: held HIGH = no motion, pulse LOW = motion detected

The alarm pin is an open-collector output so we must pull-up the pin to +5V with a resistor (e.g., 10kΩ) so it is held HIGH when no motion is detected (otherwise it would be floating and have an unpredictable state). When motion is detected, the pin is pulled to common (pulsed, not held).



Note: Use an 8.2kΩ resistor instead of the 10kΩ shown here.

2.3.5 Temperature Sensor (TMP36)

The TMP36 3-pin temperature sensor is an active component that provides a voltage output that is linearly proportional to the temperature (in Celsius, Kelvin, or Fahrenheit). It does not require any calibration.

The output can be directly fed into the PIC; an ADC is then performed to determine the output voltage of the sensor, and thus code can be written to provide the temperature reading (in any temperature unit). For the Alarm System, use and display in the unit of Fahrenheit.

A 10 mV change on the output corresponds to a change of 1°C. Look on page 8 of the temperature sensor's datasheet for information regarding a constant offset that you must account for in your code.

It is a good practice to use buffer capacitors on the supply voltage of all active components. Use one 0.1 µF capacitor between the +5V and GND for optimal operation. The capacitor should be placed (physically) as close to the temperature sensor as possible.

Read the temperature sensor's datasheet for the pinout and more information.

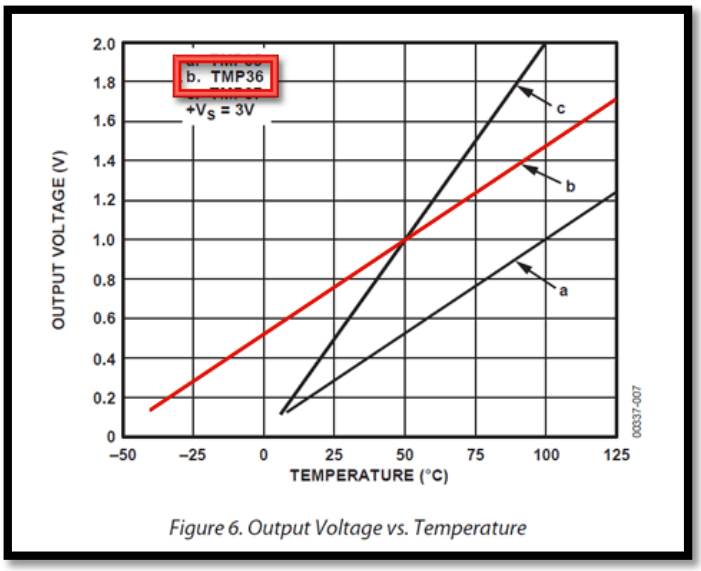
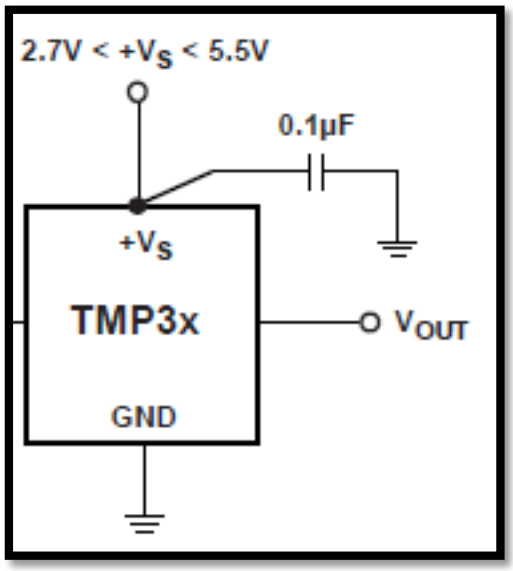


Figure 6. Output Voltage vs. Temperature



2.3.6 Pushbutton for MCLR Reset

The $\overline{MCLR}$ pin of the PIC has to be kept HIGH for normal operation but can be brought down LOW to trigger a $\overline{MCLR}$ reset. Section 4.2 of the PIC's datasheet contains more details. Use a two-pin pushbutton and a resistor (8.2kΩ) to complete a reset circuit.

2.3.7 External Crystal Oscillator

You will be using a 20MHz crystal oscillator instead of the PIC18F4520's internal 8MHz. External higher-speed crystals are preferred when using the communications peripherals of microcontrollers.

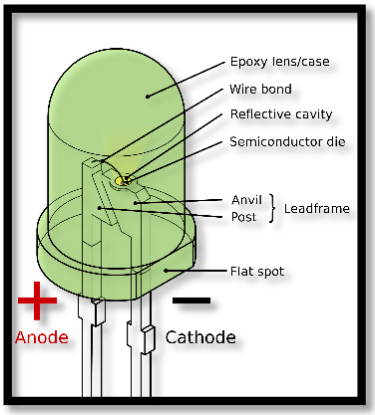
Using the two 18pF capacitors and the diagram in section 2.2 of the PIC's datasheet hook up the 20 MHz crystal to the appropriate pins on the PIC18F4520.

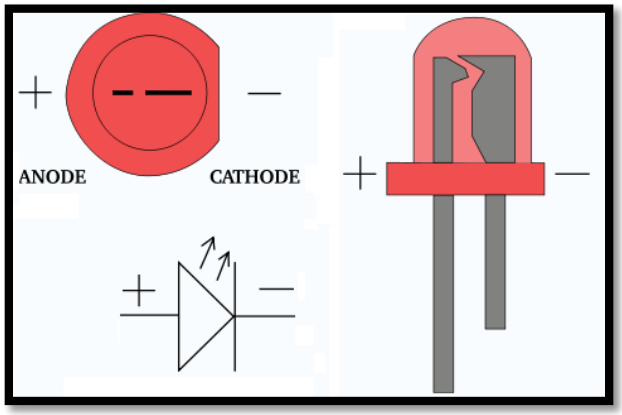
It is crucial that the crystal and capacitors be as close to the PIC pins as possible and that their leads are as short and snug to the breadboard as can be (do not cut the pins too short though to prevent connection problems with the bread board).

Make sure you set the right configuration bits for the Oscillator Type (OSC).

2.3.8 LEDs (Light-Emitting Diodes)

You will need to use four LEDs in your circuit. Their purpose is explained in the following sections. Orient them correctly and use one of the 150Ω resistors with each of them so they do not consume too much current and burn out.

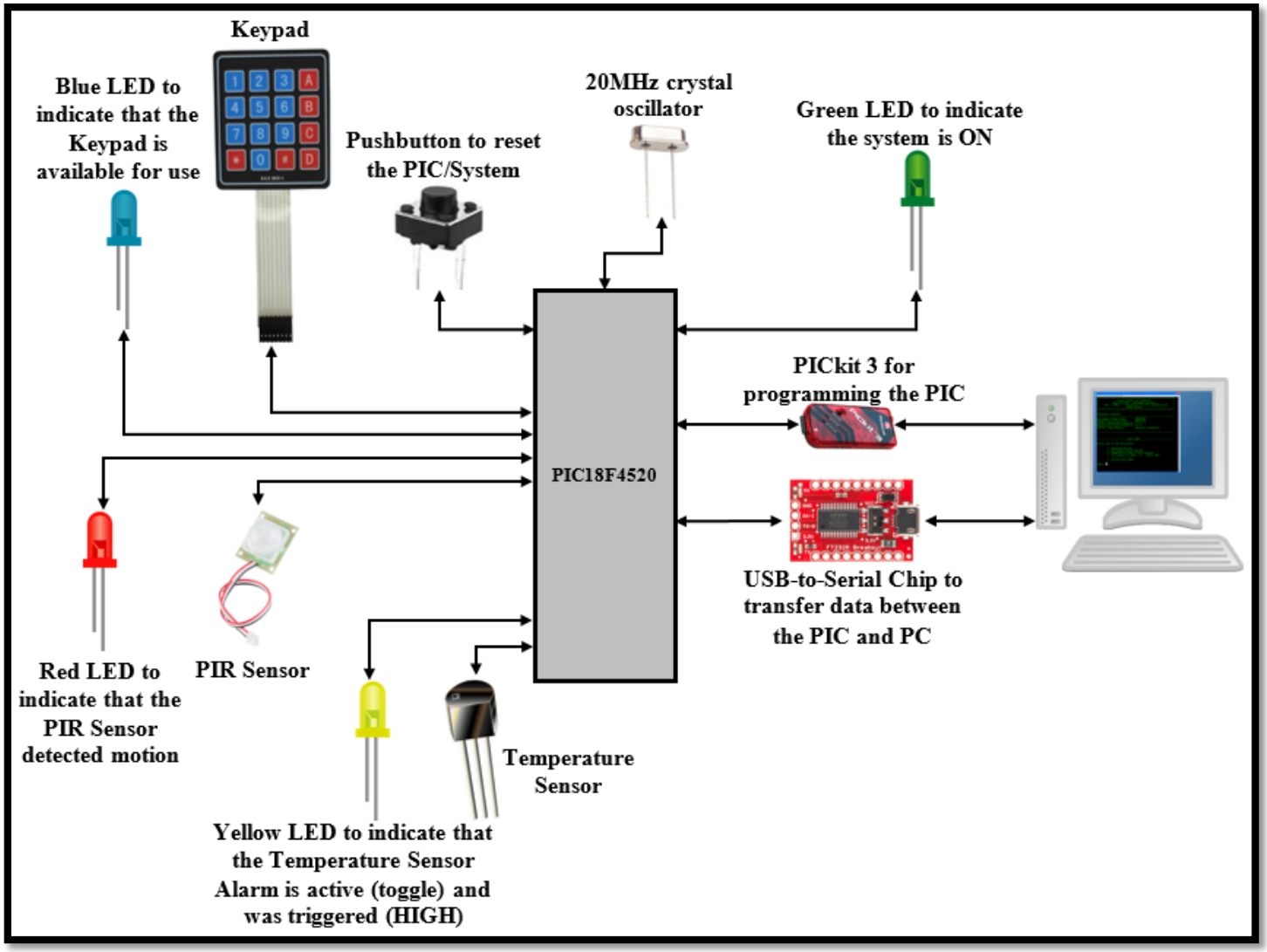




3 Detailed Description

This section provides the actual details of how to create the Alarm System and what output/functionality is expected. When planning, creating, wiring, and programming the Alarm System you need to strategize and break down everything into small more manageable parts and integrate them piece-by-piece to eventually construct the whole Alarm System.

The following graphic is not a detailed and honest circuit diagram so it does not show "actual" pin and wire connections, just general high-level layout connections of the system. The details of how to hook-up each component are addressed in the previous section.



Note: *The entire Alarm System circuit will be powered through the power provided by the USB-to-Serial Chip (FT232RL) coming from the computer's USB connection (no battery or direct wall power adapter is needed).
*Make sure to use the 4 150Ω resistors with the 4 LEDs

3.1 Alarm System Requirements

You do not have to design your Alarm System and its interfaces exactly as the example provided in lab; you are given design freedom to arrange the Alarm System code structure how you like as long as the following requirements and features listed below are fulfilled.

1) The PIC is powered ON and the current method of input is used (keyboard or keypad)
   1.1) If this is the System's first time to be powered ON (first time use after a new PICKit programming aka initial startup), the default method of input is the terminal/keyboard and the user is shown a message telling them to enter and set the passcode for the system since there isn't one already in the system.
   1.2) The user can then enter a 4-digit passcode on the PC and presses enter.

2) The user is shown a message on the PC telling them to enter the system's passcode to enter the Main Menu. If incorrect passcode is entered the embedded system prompts to, try again. (This should work in a general terminal emulator program, so the messages should be coming from the PIC system.)

3) The green LED is turned ON to indicate that the system is up and running.

4) If the passcode entered is correct, the user is shown a Main Menu with current device/component states and statuses along with a command list for them to choose.
   A. Passcode Options
      - User must be able to change the passcode. They first give the current passcode and then give the new one. Only one passcode is required.
      - BONUS: Implement a Username/Passcode scheme to allow multiple users with different system privileges for 5 extra points.

   B. PIR Sensor Alarm Options and Features
      - MUST BE INTERRUPT DRIVEN
        - An external interrupt is triggered and handled immediately.
      - The user must be able to enable and disable the state of the alarm. The alarm is disabled upon initial startup.
      - Whatever state the user sets the alarm to should be the state when the PIC restarts after losing power or after a $\overline{MCLR}$ reset.
      - The PIR Sensor Alarm has the highest priority over everything else in the system. No matter what state/screen the system is currently in (except the initial *Enter Passcode* screen), if motion is detected the user is immediately notified and they are instructed to enter the system's passcode to reset it and move forward.
      - The red LED is turned ON when motion is detected. It is turned OFF after the user clears/resets the alarm.
      - BONUS: Allow the user to set a timer for when the PIR Sensor Alarm should turn on in the future for 5 extra points (motion auto arm).

C. Temperature Sensor Alarm Options and Features
- MUST BE INTERRUPT DRIVEN
  - A timer interrupt and an ADC interrupt handle the periodic sampling of the Temperate Sensor ADC reading.
- The user must be able to enable and disable whether temperature can trigger alarm. The alarm is disabled upon initial startup.
- The user must be able to change the temperature threshold setting that triggers the alarm when the actual temperature reading reaches or exceeds that amount. When displaying, show all degrees in Fahrenheit
- Whatever state the user sets the alarm to should be the state when the PIC restarts after losing power or after a $\overline{MCLR}$ reset.
- The Temperature Sensor Alarm has the 2nd highest priority over everything else in the system (PIR has the greatest). No matter what state/screen the system is currently in (except the initial *Enter Passcode* screen), if the temperature reading equals or exceeds the threshold, the user is immediately notified and they are instructed to enter the system's passcode to reset it and move forward. The user is also given the option during an alarm triggering to adjust the temperature threshold in case it is too low and constantly setting off the alarm. Even at this screen/state the PIR Alarm can still trigger.
- The yellow LED should be toggled at the acquisition of a new ADC sample (timer). The LED should remain constantly ON when the temperature alarm is triggered. It should be turned OFF or back to toggle action after the user clears/resets the alarm.
- BONUS: Allow the user to set a timer for when the Temperature Alarm should turn on in the future for 5 extra points (temp auto arm).

D. Switch Input Method
- The user must be able to switch freely between having the PC's Keyboard and the 4x4 Keypad as input.
- The Keyboard is the default method of input upon initial startup.
- Whatever input method the user picks should be the method when the PIC restarts after losing power or after a $\overline{MCLR}$ reset.
- All system input should be doable both from the PC and the keypad. You need to designate one of the Keypad's buttons as an *Enter* key.
- When the Keypad is a selected input, turn the blue LED ON. This LED indicates to the user when it is safe to press a new key on the Keypad, so it needs to alternate ON to OFF to ON when a user presses any key. This is to ensure no input is missed by the program. Turn OFF when the Keyboard is the only selected input method.
- BONUS: Allow the user to use BOTH the keyboard and keypad simultaneously for 5 extra points. This does not replace the ability to switch between single input methods, it's only a 3rd option for the user.

5) Additional Features and Requirements

A. Invalid Input Handling

- All inputs at any state/screen need to have error and incorrect input handling (e.g., the user should not be able the enter option 8 if there is no such option, etc.)

B. Reset Feature

- At any point in time, the PIC can be reset using the push button attached to the $\overline{MCLR}$ pin to restart the Alarm System.
- This is a "soft" reset and important variables/settings should not be lost. Explained in detail below.

C. Permanent Storage for Important Settings/Variables

- It is required that important variables (passcode, sensor settings, etc.) are non-volatile so they will not be lost or changed if the PIC loses power or is reset using the $\overline{MCLR}$ pin.
- You must use the PIC's EEPROM capabilities to achieve this. You may use the header file *eep.h* so you don't have to set the EEPROM registers "manually" (there is an abundance of examples online of how to write to and read a PIC's EEPROM).
- If you ever need to verify and look directly at what values are stored in the PIC's EEPROM, in MPLAB use…
  Window > PIC Memory Views > EE Data Memory.
- Reprogramming the PIC through MPLAB will erase EEPROM.

D. BONUS: Menu Timeout (5 extra points)

- If the user has not input anything or an alarm has not been triggered after a certain amount of time, the system goes to an *Idle State* where the passcode must be entered to proceed and reengage the system.
- The user should be able to set and adjust this time limit in the same fashion as changing the temperature threshold.

E. BONUS: Writing a PC-Side Program (10 extra points)

- You may write a simple C/C++ program on the PC to handle communication with the PIC. The C++ program would replace the use of the terminal's command-line style I/O with a simple GUI with buttons, text boxes, drop-down lists, etc. You can use any OS (Linux, Windows, MAC OS, etc.).

F. Adding Your Own Features (possible bonus points but not guaranteed)

- You may of course expand upon or add more features to the system as long as it does not take away from the requirements specified above. Well thought out and nice-looking interfaces are a plus as well.

4 Deliverables

4.1 Circuit Schematic

You must create a circuit schematic of the entire Alarm System. This can be hand-drawn and scanned (but only if it is neat, clean, and professionally done) or using any other software. We recommend the use of PCB Artist, but this will require some learning.

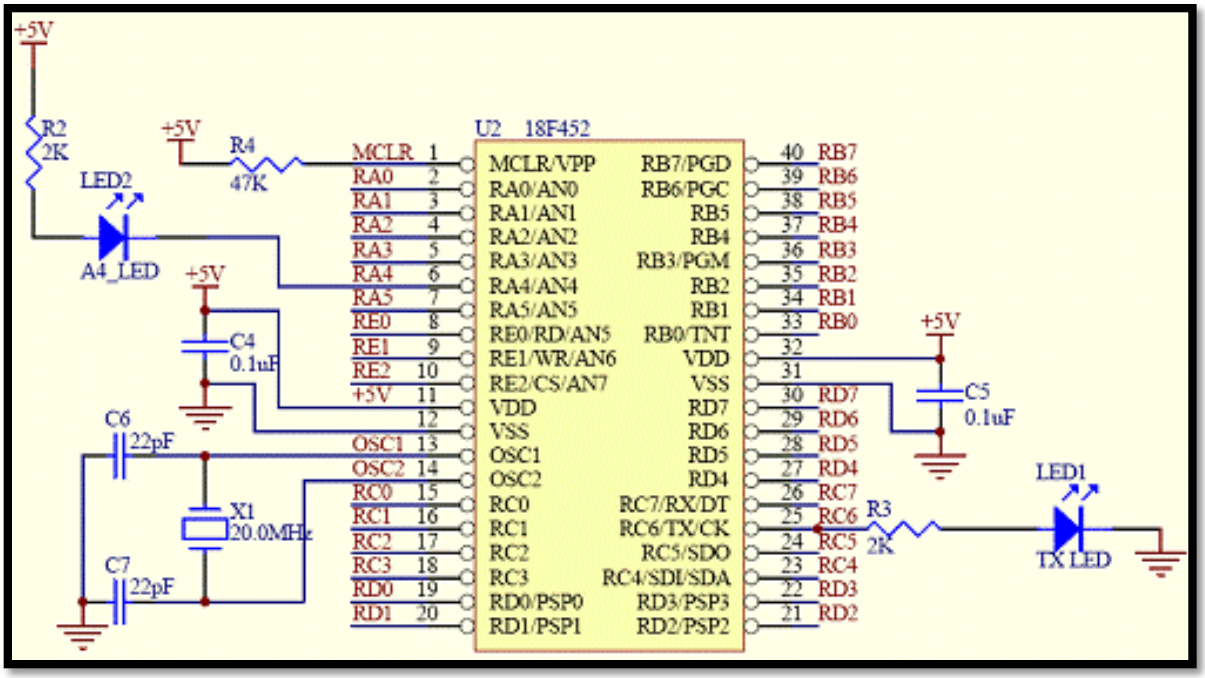
PCB Artist is free Printed Circuit Board and Schematic Design software.

Download for free here: http://www.4pcb.com/free-pcb-layout-software/
That link also contains video tutorials about how to use the software.

Here are some other useful tutorial links:
http://www.4pcb.com/downloads/pcbartist/PCBArtistTutorial.pdf
http://www.4pcb.com/media/PCBArtistTutorial.pdf
http://www.4pcb.com/media/tips-for-pcb-artist.pdf

Pay attention about how to add your own custom components to the PCB Artist libraries since PICs are usually not standard.

The below schematic drawing was not created using PCB Artist but it may provide a general idea of what is required:



4.2 Deliverables

Put all of the following neatly organized into a zip file and upload to Blackboard.

1. A single schematic file as an image (*.tif, *.jpg, *.png, *.bmp, etc.) of your circuit
2. A single source file of your schematic if done in a CAD tool (e.g., *.sch of PCB Artist, *.ppt if using PowerPoint, etc.)
3. Your .c file(s) and .h file(s) (if you made any headers) with **detailed** comments
4. A single well-organized lab report (.pdf) containing…
    1) Alarm System description
    2) General textual overview of your process, planning, and solution
    3) Detailed description of your use of PIC Interrupts
    4) Detailed description of your use of the timer modules
    5) Your timer and temperature (ADC) calculations in detail
    6) Detailed description of your process for reading the Keypad
    7) A block diagram (flow chart) of your program
    8) Any and all problems encountered
5. Suggestions for how to make this ABET lab better in the future (complete honesty is encouraged)

Note: Include screen shots and/or diagrams from the PIC's datasheet, other documents, or websites and feel free to expand by adding more sections.

5 Grading Breakdown

5.1 Grading

50% - Quality of your circuit/program/code/user interface
(How efficient is it? Are there any bugs? Is there any missing functionality? Are there any unexplainable delays?)
20% - Quality of lab report and circuit schematic (content and style/formatting as well)
20% - Quality of comments in source code
10% - Quality and neatness of your hardware implementation on the bread board.

5.2 Possible Bonus Points

35 Possible Bonus Points (specified in *3.1 Alarm System Requirements*)

- 5 points – Implement a Username/Passcode scheme to allow multiple users
- 5 points – Have a user-set timer for the PIR Sensor Alarm to turn on in the future
- 5 points – Have a user-set timer for the Temperature Alarm to turn on in the future
- 5 points – Allow simultaneous input from keyboard AND keypad as a 3rd input option
- 5 points – Menu Timeout
- 10 points – Writing a PC-Side C/C++ Program

ABET: A failing or incomplete grade (for the course) is given if you don't implement, complete, and pass the lab or fail to submit the deliverables and demo. Passing grade for this lab is 70%.