Middle East Technical University Department of Computer Engineering

CENG 336

Introduction to Embedded Systems Development
Spring 2020
THE3

Due date: 03.06.2021, Thursday, 23:55

This assignment mainly encompasses the usage of the analog-to-digital converter (ADC), LCD, and timers. It also deals with the use of the heater, LEDs, and buttons. You will use the C programming language to create your program for the PIC microcontroller.

1 Overview

IIn this assignment, you will implement the logic behind an electronic safe. Users will be able to set the combination to the safe and unlock it with the selected combination. The safe will also perform various actions when a wrong combination is entered.

Clarifications, revisions, and changes on the scope and the specifications of the assignment will be posted to the discussion forum on the course page at ODTUClass. Make sure you follow the discussions and make sure that your question is not asked before creating new threads.

2 Scenario

A generations-old safe manufacturer, unPICable has been producing high-quality analog safes for more than a century. Now they want to catch up to the technology and create a digital safe with password protection. They also want the safe to stand out between its competitors, so they want to add some protection mechanisms to secure the contents of the safe against thieves. You are hired as their sole embedded systems developer and tasked with creating the software that will run on these safes.

2.1 Safe Operation

When the safe is first powered on, the system will ask the user to set the combination to the safe. The combination is three 2-digit numbers. After the combination is set, the safe begins to wait for the correct combination to be input to unlock the safe. The user has 90 seconds to enter the correct combination after the first movement of the input dial (Potentiometer connected to `AN0`) and has three attempts in total to enter the correct combination. When the correct combination is entered, the safe unlocks. When the safe is unlocked, a button press tells the safe to lock again, and the system starts to wait for the combination again.

When a wrong number is entered as part of the combination, the safe starts to activate some countermeasures depending on the remaining number of attempts. On the first failed attempt, the safe starts to toggle LEDs on the safe on and off. On the second failed attempt, the safe turns on an internal heater, which heats the safe contents to burn the contents in case the thief somehow manages to unlock the safe. If the user runs out of attempts, the contents of the safe are destroyed, and a message is displayed on LCD.

2.2 Specifications

This section explains the behavior of the program in detail in different stages of the execution.

2.2.1 Start-Up State

hen the safe is first powered on, it first displays "SuperSecureSafe!" on the LCD for 3 seconds. You can use busy-wait for this delay. After 3 seconds elapsed, the safe moves on to the **Password Set State**.

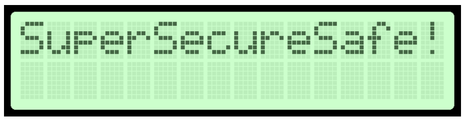

```
SuperSecureSafe!
```

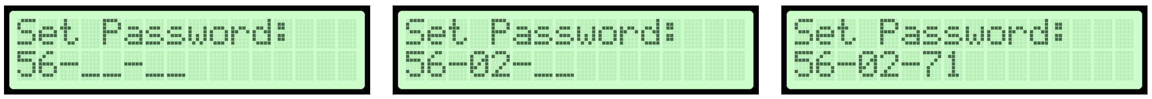
Figure 1: Message displayed in Start-Up State.

2.2.2 Password Set State

In this state, the safe waits for the user to enter a master combination. The user will select three 2-digit numbers using the dial on the safe (Potentiometer on `AN0`) and confirm each number with the `RB0` button. During this state, the LCD will display "Set Password:" on the first line and the three, 2-digit numbers on the second line separated with dashes. Double underscore characters (__) will be used for the numbers that are currently not set. For example, if only the first number is set as 56, the second line of the LCD will display “56-__-__”. The user should be able to see the part of the combination currently being set change while the dial is being turned. When the `RB0` button is pressed, the current value is locked in as the corresponding part of the safe’s combination.

ADC gives you values between 0 and 1023. You will get the corresponding 2-digit number by dividing the ADC value by 10. For example, when the ADC value is 7 the corresponding 2-digit number is ’00’, when the ADC value is 88 corresponding 2-digit number is ’08’, when the ADC value is 649 corresponding 2-digit number is ’64’.

Since you cannot get a 2-digit number by dividing the ADC value by 10, when the ADC value is above 999, you will need to sanitize the ADC values somehow to get 2-digit numbers correctly. You will do this by not accepting any value that is above 999 (or 99 as a 2-digit number) and displaying double ’X’s (`XX`) when the ADC value goes above 999. If the `RB0` is pressed while `XX` is being displayed instead of a number, the program should do nothing because `XX` is not accepted as a valid 2-digit number.



```
Set Password:
56-__-__
```

(a)

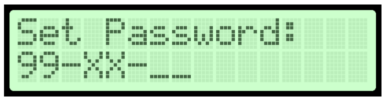
```
Set Password:
56-02-__
```

(b)

```
Set Password:
56-02-71
```

(c)

Figure 2: LCD while setting the (a)first, (b)second, and (c)third part of the combination. (a)→(b)→(c) transition is triggered by `RB0` button



```
Set Password:
99-XX-__
```

Figure 3: ADC value is above 999 when setting the second part of the combination. If `RB0` button is pressed in this situation the program does nothing.

Moreover, the direction of the dial will change with each entered number. When entering the first number, the ADC values will increase as the potentiometer is turned clockwise. ADC values will decrease when entering the second number as the potentiometer is turned clockwise, and again when entering the third number, ADC values should increase with a clockwise turn just like the first number. Figure 4 shows the dial ranges for different parts of the combination input.

After the third number of the combination is selected and the `RB0` button is pressed, the system accepts the combination as the master password and moves onto the **Password Check State**.

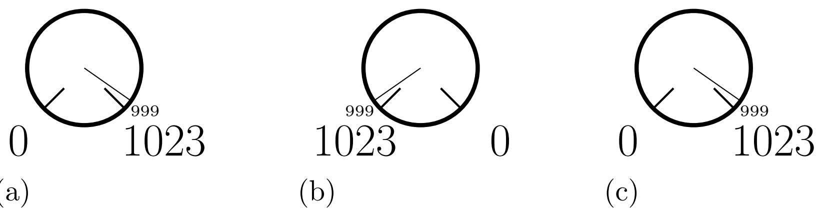
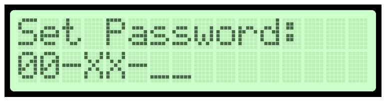


Figure 4: Dial ranges when inputting the (a)first, (b)second and (c)third part of the combination

2.2.3 Password Check State

In this state, the program waits for a combination input to be compared against the master combination set in the *Password Set State*. Combination password input behaves exactly like the *Password Set State*, except this time the LCD displays "Input Password:"" on the first line.



```
Set Password:
00-XX-__
```

Figure 5: Combination input in the Password Check State. Second part of the combination is being entered and the current value of ADC is above 99, so XX is being displayed.

The 7-segment display also lights up in this stage. It displays the number of remaining attempts and the remaining time to enter the correct combination. The remaining attempts are shown as horizontal lines on the third digit from the right of the 7-segment display. The rightmost two digits are used to display the remaining time. Users are given three attempts and 90 seconds. 90 seconds starts to count down when a change in the ADC value is detected or the `RB0` button is pressed (this press still registers the current ADC value). So the timer doesn't immediately begin counting down when the system gets into Password Check State. It starts to count down when the system is in Password Check State, and a change in ADC value is detected or the `RB0` is pressed, and it doesn't stop counting down until either user runs out of attempts or the timer reaches 0. Time is always displayed as two digits, leading with a 0 if necessary.

If the user manages to enter the correct combination, i.e. the entered combination matches the one entered in *Password Set State* exactly, without running out of attempts or time, the system moves on to the **Success State**, and the safe is unlocked.

When a part of the combination is entered incorrectly, we count this as a failed attempt and reduce the remaining attempts by one. As mentioned before, the safe deploys countermeasures depending on the number of attempts remaining.

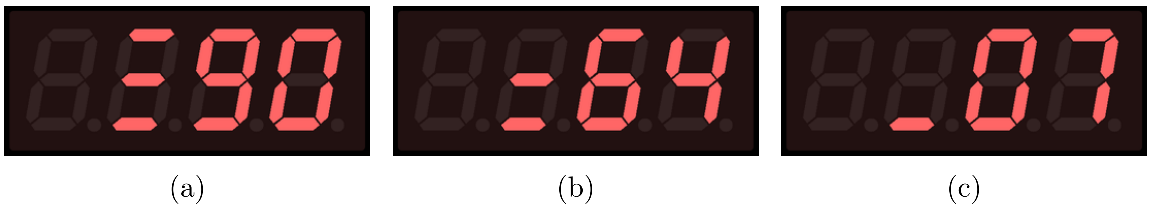


Figure 6: 7-segment display with (a)three, (b)two, (c)one attempts remaining with different remaining times.

3 Attempts Rem. There are no active countermeasures. There is only the 90-second timer counting down on the 7-segment display. To decrease the remaining time, you need to use the timers and their interrupts.

2 Attempts Rem. The LEDs connected to the `PORTB` start to blink every 500ms. You need to keep track of this duration using timers and interrupts.

1 Attempts Rem. The system turns on the heater and starts reading the temperature as well. The heater is connected to the `RC5` and setting this pin `HIGH` will turn on the heater. The thermometer is connected to the second channel of the ADC (`AN2`). Since the ADC can sample one channel at a time, you need to keep track of the channel currently being sampled very closely so that you don't read the dial input as the temperature or the value from the thermometer as the dial input. The system will move to the **Fail State** when the thermometer reads $40°C$ or above. You need to detect that the temperature has reached $40°C$ within 1 second. It is up to you how you handle sampling `AN0` and `AN2` as long as you detect going above $40°C$ in 1 second and the user doesn't feel sluggishness when using the dial.

Note that the countermeasures apply on top of each other. 90-second timer doesn't stop once it starts counting down, LEDs keep blinking once they start blinking and the heater stays on until we move on to the **Success State** or the **Fail State**. From the *Password Check State* we can move to the *Fail State* in three ways:

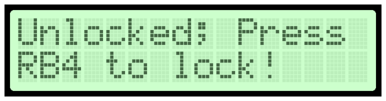
1. Running out of attempts.
2. 90-Second timer reaching zero.
3. Thermometer reading a temperature above $40°C$.

And to the *Success State* in a single way:

1. Enter the correct combination.

2.2.4 Success State

In this state the, safe is unlocked. All the LEDs (except the LED corresponding to RB4), 7-segment display, and the heater are turned off in this state. LCD displays "Unlocked; Press" on the first line and "RB4 to lock!" on the second line. It waits in this state until a button press is detected on `RB4`. With the button press system moves to the *Password Check State*, attempts and 90-second timer are also reset.

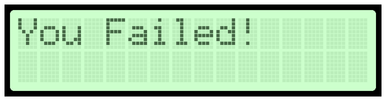


```
Unlocked: Press
RB4 to lock!
```

Figure 7: LCD in Success State

2.2.5 Fail State

This state means that the user ran out of attempts, the 90-second timer hit zero, or the temperature is above $40°C$. Upon reaching this state, the system turns off the heater and the 7-segment display. The LEDs connected to `PORTB` are fixed in a turned on state. LCD displays "You Failed!" on its first line and nothing on the second line. The program is stuck in this state until a power cycle is performed.



```
You Failed!
```

Figure 8: LCD in Fail State

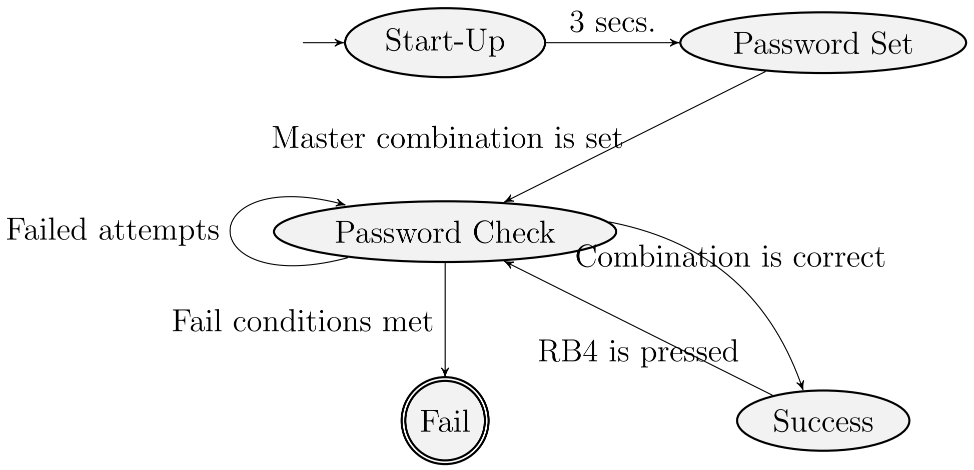


Figure 9: State diagram of the whole program

3 ADC Usage

ADC module allows conversion of an analog input signal to a corresponding 10-bit digital number. ADC module is controlled and configured by five registers:

1. A/D Result High Register (ADRESH)
2. A/D Result Low Register (ADRESL)
3. A/D Control Register 0 (ADCON0)
4. A/D Control Register 1 (ADCON1)
5. A/D Control Register 2 (ADCON2)

ADCON0 controls the operation of the module, ADCON1 configures the pin functions, ADCON2 configures the clock source, acquisition time and justification. ADRESH and ADRESL registers contain the result of the conversion operation.

When the GO/$\overline{\text{DONE}}$ bit is set the ADC module starts the conversion. When the conversion is complete the result is loaded into the ADRESH:ADRESL registers and the GO/$\overline{\text{DONE}}$ bit is cleared and the ADIF interrupt bit is set. Which 10-bits of the ADRESH:ADRESL is used is defined by the ADFM bit of the ADCON2 register.

As described in the datasheet, the following steps should be followed to perform an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, mark corresponding TRIS bits as input. Configure voltage reference and digital I/O (ADCON1)
   - Select the input channel for the conversion (ADCON0)
   - Select acquisition time (ADCON2)
   - Select conversion clock (ADCON2)
   - Turn on the module (ADCON0)
2. Configure the A/D interrupt (optional):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. What for the required acquisition time (if required)
4. Start conversion by setting GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for conversion to finish by either polling the GO/$\overline{\text{DONE}}$ bit to be cleared or by waiting for the A/D interrupt.
6. Read ADRESH:ADRESL, clear ADIF bit (if required)
7. Go to Step 1 or Step 2 for the next conversion.

See the datasheet for acquisition time and clock calculations.

4 LCD Usage

Functions to manipulate the LCD will be provided with the homework files. This section only exists to explain how the LCD and the provided functions operate.

PICGenios board of the PICSimLab has the HD44780 (16×2) LCD connected to it. It has 16 pins that we can use to communicate with it to display the glyphs we want. 5 of these 16 pins are used for ground(`VSS`), power(`VDD`), contrast of display(`V0`) and backlight(`LED+/LED-`) and aren't directly related to the operation of the LCD. These are already connected on the PICSimLab properly and we don't need to look into these 5 pins. Moreover, the `R/W` pin on PICGenios board is hardwired as 0 (connected to ground), so we don't need to consider it as well. When `R/W` pin is set `HIGH`, it is used to read values from the LCD's memory. So the pins we need to learn how to operate are Register Select(`RS`), Enable(`E`) and data bus(`D0:D7`). You can see in Figure 10 which pins of the PIC the LCD pins are connected to.

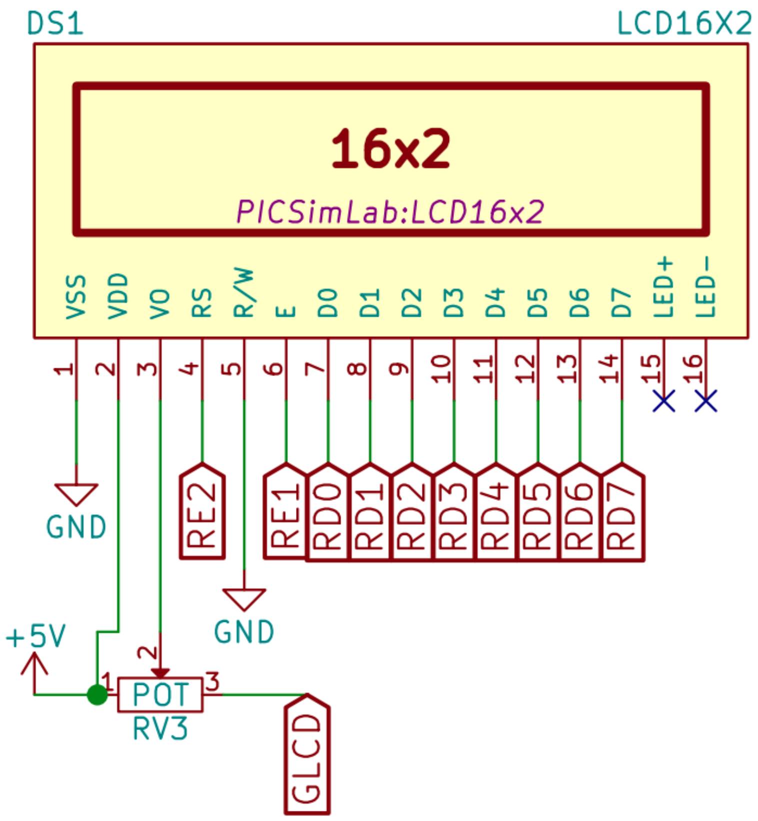


Figure 10: LCD Pin Diagram

LCD has two operation modes: command mode and data mode. We specify which mode we want to operate in by setting the `RS` pin. `LOW RS` means command mode and `HIGH` `RS` pin means data mode. The values on the data bus `D0:D7` are interpreted as commands in command mode and as data in data mode. Enable pin `E` is used to notify the LCD that the contents of the data bus and `RS` pin are set correctly and are ready to be read. When we pulse (i.e. setting `HIGH` and then `LOW` again) `E`, LCD reads the information and performs the operation we specified.

With commands, we can perform actions such as clearing the display, setting cursor position, turning cursor display on/off, shifting the display etc. For example, to clear the LCD and to put the cursor on the beginning of the second line we perform following operations (assuming that the LCD is properly initialized beforehand):

1. Set `RS` `LOW`
2. Put Clear Display command on `D0:D7` (Clear Display command is `0x01`)
3. Pulse `E`

- *Now the display is cleared and the cursor is at the beginning of the first line.*

4. Set `RS` `LOW`
5. Put Set DDRAM Address command on `D0:D7` with the address corresponding to the second line of the LCD. (`D0:D7` is set to `0xC0`)
6. Pulse `E`

- *Now we write character 'A' to the LCD.*

7. Set `RS` `HIGH`
8. Put the value of 'A' on data bus. (`D0:D7` is set to 0×41)
9. Pulse `E`.

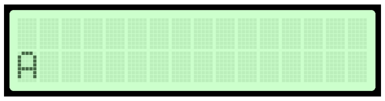


Figure 11: Result of the example LCD operation

Datasheet for the LCD can be found here. Also, Ben Eater has a great video on operating the HD44780. However, the functions to manipulate the LCD will be provided, so you don't have to learn the individual commands and implement them in your code.

5 Hints

1. Here are the ports and pins the components use:
   - **LCD:** `PORTD` for data bus, `RE1` for enable, `RE2` for register select.
   - **7-Segment Display:** `PORTD`, `RA2-RA3-RA4-RA5`. Note that `RA2` is also used by thermometer but also note that we don't need that part of the 7-Segment display.
   - **ADC:** `RA0(AN0)` for the potentiometer, `RA2(AN2)` for the thermometer.
   - **Buttons:** `RB0` will cause `INT0` external interrupt. `RB4` will cause `PORTB` port change interrupt. Read the datasheet carefully for these interrupts.
   - **Heater:** `RC5` Heater will turn on when this pin is `HIGH` and turn off when `LOW`.
2. When setting `PCFG3:PCFG0` values in `ADCON1`, both `AN0` and `AN2` should be analog. Choose a value accordingly.
3. Conversion formula from ADC to Celsius is: $\frac{ADC\_VALUE \times 5}{1024.0} \times 100.0$
4. You might have to disable and then re-enable some/all interrupts while handling some other things depending on your design.
5. You might have to store and restore port values (just like callee-saved registers from Computer Organization course), especially `PORTD` since it is used extensively by the LCD and 7-Segment display, to have a stable program.
6. You can change `TRIS` values throughout your program. Don't think that a port or a pin is always fixed as input or output.
7. In *Password Check* state LED corresponding to `RB0` won't light up due to DIP switch configuration. Also, because we need to use `RB0` button and `INT0` interrupt. That single LED not lighting up is OK in *Password Check* state.
8. You can use either `TIMER0` or `TIMER1`, or both. However, you need to clearly state which timer(s) are used for what purpose in your comments. You should also state the modes (8/16-bit) and prescalers if any.
9. It is beneficial to avoid function calls in ISRs. You may use some counters or flags and do the function calls or calculations outside the ISR.

6 Regulations

1. You will code your program using PIC C language. You should use XC8 C compiler for MPLAB X.

2. Your program should be written for PIC18F4620 working at 40MHz.

3. **Your code MUST be properly commented and you must provide a complete description of the flow of your program just before the `main` function.** 5% of your grade will be given by the quality of your comments.

4. All the timing related tasks should be performed by timers except the 3-second wait in the beginning.

5. All button related operations are handled by interrupts, polling is not allowed.

6. You must get the ADC result using its interrupt, polling is not allowed

7. RTFM! Use your lecture notes and datasheets extensively. Consult these resources first before asking any questions to the assistants or on the forums.

8. There is already a discussion forum for THE3, please ask your questions there. Unless you have *really specific* question about **your code** use the forum! If you think that your question is too specific to ask on the forum or private you can ask your questions through mail to Hakan or Saim.
   [`hbostan@ceng.metu.edu.tr` & `saimsunel@ceng.metu.edu.tr`]

9. You will submit your codes as a `.tar.gz` archive through ODTUClass.