The University of Jordan

School of Engineering

Department of Computer Engineering

Embedded Systems Course Project

Simulating Secure Communication using PICRYPT Algorithm

Project Description

In this project, you are required to implement a simple end-to-end communication system using two PIC16F877a series controllers. The system will use USART to transmit and receive data, port interrupts, timers, and will also encrypt and decrypt data using a very simple software algorithm.

The first microcontroller will have a set of four switches connected to any port of your choice. Each two switches are responsible for choosing a letter/shape/number. For example, this is a sample table:

SW1	SW2	Data	SW3	SW4	Data
0	0	'A'	0	0	'D'
0	1	'F'	0	1	2
1	0	0	1	0	4
1	1	5	1	1	'L'

You can choose any data you want for your project (different letters, shapes, numbers) than the sample table above as long as they can be represented using a 7-segment display. A push button is connected as an external interrupt source to the first PIC and only when this push button is pressed that the switches' values are read into the PIC, and the corresponding letter/shapes/numbers retrieved. Eventually, your data will be comprised from two 8-bit registers each storing a value from your table. The push button must use interrupts.

The two bytes will then be encrypted (details below) and sent using the USART module to the second PIC, which will decrypt the letters and display the sent data on two common cathode 7-segment displays. You must use interrupts for both the receiver and transmitter.

Also, the second PIC will have a flashing LED that flashes ON and OFF every 0.25 seconds. Use Timer 0 module with interrupts to achieve this delay. Assume 4MHz clock. Note that the HW will not be able to achieve this rate. **Hint**: Think of adding a counter inside the interrupt to extend the hardware timer delay. Each time an interrupt occurs this counter is decremented and is thus used to extend TMR0 range. Most of the codes that we learnt in class can be reused in the project.

PICRYPT Algorithm

In this project, you are going to use 16-bit symmetric key encryption/decryption algorithm that we invented solely for the use with this project. In symmetric key cryptography, the algorithm uses the SAME keys for encrypting and decrypting the message. In real-life, cryptography standards use 128-bit, 256-bit, 512-bit and more (*e.g.* 4K-bits) as s key size. The larger the key size, the more complex it is to crack the encryption using brute force search techniques. For example, a 16-bit key means that it only takes 65,536 attempts to guess what the key is, while a 4K bit key will require 2^{4096} attempts. In this project, the PICRYPT algorithm uses 16-bit keys due to memory limitations and only to serve as an example for cryptographic operations. Kindly note, our PICRYPT algorithm is just an algorithm for educational purposes and is not to be deployed in real systems. Our algorithm is based on similar operations that are used in many current encryption algorithms. These include xor-ciphering, bit-rotations, the use of prime numbers, and multiple rounds of encryption.

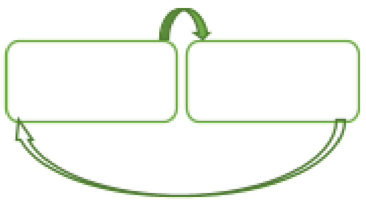
The PICRYPT algorithm uses **three** **16-bit encryption keys: Key1H** and **Key1L, Key2H** and **Key2L, Key3H** and **Key3L** that should be prime numbers of your choice in the range of 16,384 and 65,536. Remember that these 16-bit numbers will not fit in one PIC register or memory location. You must reserve two bytes of memory for each and store the keys in hexadecimal format.

For the purposes of this project, the number of rounds the algorithm goes through is two. The $\oplus$ symbol means an XOR-operation. Msg is the 16-bit data to be encrypted/decrypted.

PICRYPT Algorithm	
Encryption	**Decryption**
R = Msg	While (No_Rounds != 0)
While (No_Rounds != 0)	{
{	R = Rotate R to the right by 1
R = R $\oplus$ Key_1	R = R $\oplus$ Key_3
R = Rotate R to the left by 1	R = Rotate R to the right by 1
R = R $\oplus$ Key_2	R = R $\oplus$ Key_2
R = Rotate R to the left by 1	R = Rotate R to the right by 1
R = R $\oplus$ Key_3	R = R $\oplus$ Key_1
R = Rotate R to the left by 1	No_Rounds = No_Rounds – 1
No_Rounds = No_Rounds – 1	}
}	Msg = R
*Msg and R are simulated 16-bit numbers on PIC16Fxxx	

Notice that the Decryption algorithm is in the reverse order of operations of the encryption algorithm.

Hint: Remember that we are dealing with the keys and operations on simulated 16-bit numbers, so you have to make sure that when you use the rotate operations, that bits from one byte are rotated into the other byte as follows (Rotate right example):



Competition Bonus

It is of utmost importance to write codes which are minimum in size and execute quickly. I ask you to use subroutines, modular design and functional reuse whenever possible. In some cases, the use of indirect addressing (FSR and INDF) can be helpful in reducing code size and increasing speed. It is important to not forget to use functional comments.

During the discussion, I will ask you to show me on MPLAB the total program size on one of the PICs or how long it takes to do a certain subroutine or code. I will collect this data for all groups and assign bonuses as follows:

Shortest code or quickest code	2 Marks Bonus
2nd shortest code or 2nd quickest code	1.5 Marks Bonus
3rd shortest code or 3rd quickest code	1 Mark Bonus

If any two groups have suspiciously similar codes, timing, or size, both groups will get a zero for the bonus.

Signup Sheet

The project must be done by **three** students within the allotted time frame. Kindly divide yourself into groups and signup your names in the following Google sheet Embedded Course Project Groups before December 25th, 2022

Grading

The project is assigned the following grades

- Working Proteus Model (2 Marks) *(Working 2 / 2, partially working 1 / 2, not working 0 / 2)*
- Correct setup/configuration of hardware (TMR, USART, PORTs) (3 Marks, 1 Mark each)
- PICRYPT (2 Marks, 1 for working encryption, one for working decryption)
- Project Discussion and Understanding (3 Marks)

Important Notes

➢ Start as early as possible on your project, though the project description sounds simple, there is inherent complexity in both hardware and software aspects, so do not underestimate the time it needs, you might have some problems along the way which you will have to resolve!

➢ Never think of buying a model or commissioning someone to do it for you, not only will you get a zero in the project, but also your act will be considered as a direct violation to JU laws and your actions shall be reported as cheating in the final exam!

➢ **Code sharing between groups is NOT allowed and leads to 0 points.**

➢ If you acquire a ***part*** of your software from a book, website, etc … kindly reference it properly, else it will be considered as plagiarism.

➢ You are only allowed to base your project on PIC16F877a.

➢ All programming must be done in **PIC ASSEMBLY** language only; using high level languages in the project will get you a Zero.

- Your submitted work must be professional:
  - Software: your work should be fully documented, all inputs/outputs should be listed, and each subroutine/macro should be fully documented! Use functional comments!
- Students are not allowed to move between groups once they are formed, so choose your group carefully from the beginning! ***We are not responsible if your colleagues in the group chose to drop the class, we will not allow you to join another group!***
- Divide the work such that each student is responsible for a specific task, **YET EVERY** student is required to answer for **ANY QUESTIONS** in relation to any submitted work of the project.

Software

MPLAB LINK
https://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_IDE_8_92.zip

Proteus: You can get the software from the Embedded Systems Lab or you can find some versions online

Good Luck and Have Fun Building the Project