CPE 325: Embedded Systems Laboratory

Final Project

Cipher Tool

Submitted by: Esther Shore

Date of Experiment: July 24, 2023

Report Deadline: August 1, 2023

Demonstration Deadline: August 2, 2023

**Project Description**

This project for MSP430FG4618 acts as a cryptography tool, receiving messages input through a keyboard and transmitting the corresponding message in code across the connected UART terminal. Multiple ciphers are set up, namely the Caesar cipher, substitution cipher, and XOR cipher, for use and the complexity of the cipher is indicated by the frequency of a blinking LED. The more simple and easy to decode ciphers are indicated by a slower blink rate while more complicated and difficult to decode ciphers are indicated by a faster blink rate. These varying frequencies are controlled by setting the timer count of Timer B. The press of Switch 1 is utilized to control the switching between different ciphers, rotating through them on each press and displaying the selection in the terminal. The switch function is handled through an interrupt service routine for the corresponding port. The encoding itself occurs within subroutines of the software program for each of the corresponding ciphers.

**Theory Topics**

1. Port I/O and interrupts

Interrupts are essentially a break from the main program when certain conditions are met, executing the interrupt service routine, and then immediately transferring control back where the main program left off. They are able to handle events asynchronously, effectively managing events that are reliant on time. Interrupts are often used to detect events such as switch presses. This was utilized for the switches and LED setup in the project.

1. TimerB

TimerB is a peripheral device able to raise interrupt requests at regular intervals that can be specified by the capture and compare blocks as well as the corresponding control register. Each timer consts of a time block that supports different counting modes and source clocks. The MSP430 is able to have multiple independent timers. Timers also allow precise timestamping of events. This was used to handle the LED blink frequency.

1. UART with MobaXterm

UART is a hardware communication protocol that is asynchronous and stands for Universal Asynchronous Receiver/Transmitter. Each byte of data that is sent is framed by a start and stop bit in addition to an optional parity bit. The speed of data transfer is configurable by adjusting the baud rate (bits/sec). This method was used to implement the interactivity of the program, allowing the user to input strings and view the output strings.

1. Switches

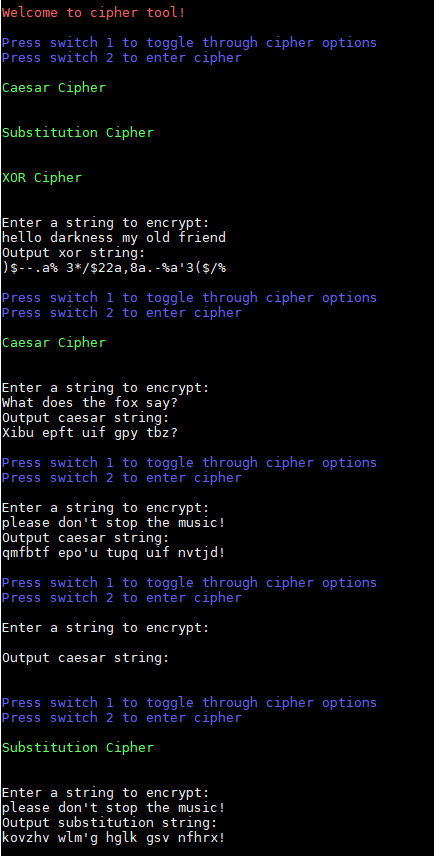
Both Switch 1 and Switch 1 were configured to be handled by an ISR for Port 1. A reasonable delay cycle was used to implement debouncing.

1. LEDs

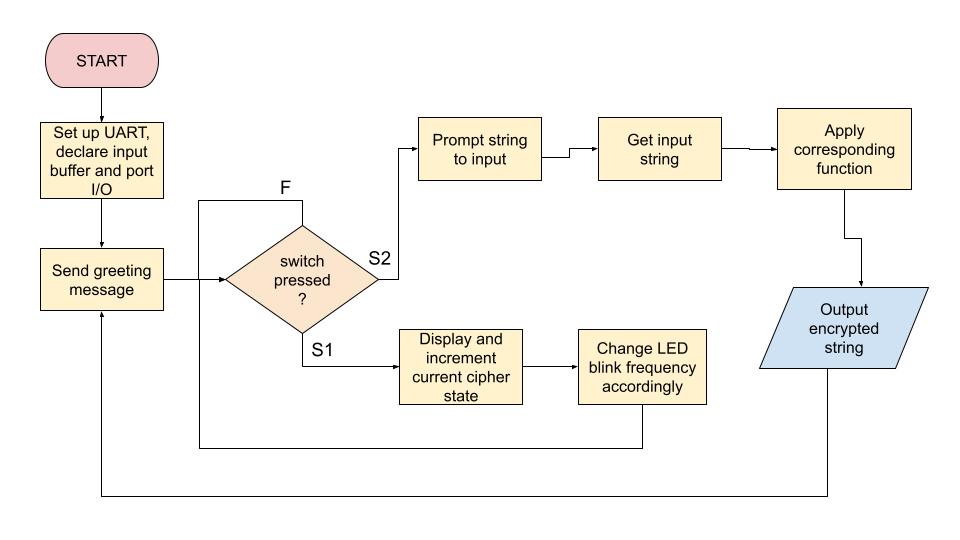
Using TimerB to control the toggling of LED2 at P2.1, the frequency of the blinking was calculated based on the equation f = 32767/CCR0/2 using the ACLK in UP mode and toggle mode.

**Project Documentation**

***Program Sample Output:***



***Program Flowchart:***



This program is an infinite loop, where the switch presses trigger an interrupt and handle the corresponding actions based on which interrupt flags were set.

**Conclusion**

This project was ultimately successful although there were some difficulties in the implementation and time management. When attempting to add the timer and interrupt functionality to the program with UART working, some of the port setup seemed to be interfering with the proper use of UART. It also posed a challenge trying to put all the logic in the main loop, so the bulk of the program handling ended up within the interrupt service routine, which is far from ideal, but is functional. A lack of sufficient time management also resulted in stress and some messier code. Overall, it was a beneficial experience to combine many of the concepts learned from the previous labs.

***Appendix:***

**Table 1:** Program Source Code

| /\*------------------------------------------------------------------------------  \* File: project.c  \* Function: Cipher tool  \* Description: Toggles between different ciphers via switch press, relative  \* security indicated by frequency of LED blinking  \* Input: Characters entered via MobaXterm received via UART  \* Output: Encrypted text  \* Author: Esther Shore  \* Date: August 2023  \*----------------------------------------------------------------------------\*/  #include <msp430xG46x.h>  #include <stdio.h>  #include <string.h>  #include <ctype.h>  // ANSI escape codes for text color  #define COLOR\_RED "\x1b[31m"  #define COLOR\_GREEN "\x1b[32m"  #define COLOR\_BLUE "\x1b[34m"  #define COLOR\_RESET "\x1b[0m"  // function prototypes  void UART\_setup();  void UART\_send\_character(char);  void UART\_send\_string(char\* string);  void UART\_get\_word(char\* buffer, int limit);  void caesar\_cipher(char\* string);  void substitution\_cipher(char\* string);  void xor\_cipher(char\* string);  void switch\_setup();  void timerB\_setup();  unsigned int current\_cipher = 0; // 1: caesar, 2: substitution, 3: xor  unsigned int S1pressed = 0;  unsigned int S2pressed = 0; // booleans for switch press handling  char input[50]; // buffer for input strings  void main(void) {  WDTCTL = WDTPW + WDTHOLD; // Stop WDT  \_EINT(); // enable interrupts  switch\_setup();  timerB\_setup();  P2DIR |= BIT1; // LED2 output  P2SEL |= BIT1 + BIT4 + BIT5; // set special reg for timer b and tx/rx  UART\_setup();  // greeting message  UART\_send\_string(COLOR\_RED);  UART\_send\_string("\r\nWelcome to cipher tool!");  UART\_send\_string(COLOR\_BLUE);  UART\_send\_string("\r\n\r\nPress switch 1 to toggle through cipher options\r\nPress switch 2 to enter cipher\r\n");  UART\_send\_string(COLOR\_RESET);  while (1) {  \_BIS\_SR(LPM0\_bits + GIE);  }  }  void UART\_setup() {  UCA0CTL1 |= UCSWRST; // \*\*Put state machine in reset\*\*  UCA0CTL1 |= UCSSEL\_2; // SMCLK  UCA0BR0 = 18; // 1MHz / 57600 = 18.2(see User's Guide)  UCA0BR1 = 0; // 1MHz 57600  UCA0MCTL |= UCBRS\_1 + UCBRF\_0; // Modulation UCBRSx=1, UCBRFx=0  UCA0CTL1 &= ~UCSWRST; // \*\*Initialize USCI state machine\*\*  IE2 |= UCA0RXIE; // Enable USCI\_A0 RX interrupt  }  void switch\_setup() {  P1IE |= BIT0;  P1IES |= BIT0;  P1IFG &= ~BIT0; // switch 1 setup with interrupt  P1IE |= BIT1;  P1IES |= BIT1;  P1IFG &= ~BIT1; // switch 2 setup with interrupt  }  void timerB\_setup() {  TB0CCTL0 = OUTMOD\_4; // toggle mode  TB0CTL = TBSSEL\_1 + MC\_1; // ACLK, UP mode  TB0CCR0 = 32767; // default 1 sec interval, 0.5 Hz frequency blink  }  void UART\_send\_character(char my\_char) {  while (!(IFG2&UCA0TXIFG)); // USCI\_A0 TX buffer ready?  UCA0TXBUF = my\_char; // TX -> RXed character  }  void UART\_send\_string(char\* string) {  int i;  for (i = 0; i < stringlen(string); i++) {  UART\_send\_character(string[i]);  }  }  void UART\_get\_word(char\* buffer, int limit)  {  int i = 0;  char received\_char;  while (i < limit - 1) {  while (!(IFG2 & UCA0RXIFG)); // USCI\_A0 RX buffer ready?  received\_char = UCA0RXBUF; // RX -> Get received character  UART\_send\_character(received\_char);  if (received\_char == '\r') {  break;  }  buffer[i] = received\_char;  i++;  }  buffer[i] = '\0'; // Terminate the string with null character  }  void caesarCipher(char\* string) {  int i;  int shift = 1;  for (i = 0; string[i] != '\0'; i++) {  char ch = string[i];  // for both upper and lower case characters  if (ch >= 'A' && ch <= 'Z') {  ch = 'A' + (ch - 'A' + shift) % 26; // determine new character based on shift  } else if (ch >= 'a' && ch <= 'z') {  ch = 'a' + (ch - 'a' + shift) % 26;  }  string[i] = ch;  }  }  void substitutionCipher(char\* string) {  const char key[] = "ZYXWVUTSRQPONMLKJIHGFEDCBA"; // example key, could be any order  int i;  for (i = 0; string[i] != '\0'; i++) {  char ch = string[i];  if (isalpha(ch)) {  ch = toupper(ch); //convert char if letter, temp convert to uppercase to handle  int index = ch - 'A'; // find index of char in the alphabet key  // find corresponding char in alphabet key  if (index >= 0 && index < 26) {  string[i] = isupper(string[i]) ? key[index] : tolower(key[index]); // back to original case  } else {  string[i] = ch; // not changed if not found  }  }  }  }  void xorCipher(char\* string) {  char xor\_key = 'A'; // could be any number or character  int i;  for (i = 0; string[i] != '\0'; i++) {  string[i] = string[i] ^ xor\_key; // xor each character  }  }  #pragma vector = PORT1\_VECTOR  \_\_interrupt void Port1\_ISR(void) {  if (P1IFG&BIT0) {  if (S1pressed == 0) {  \_\_delay\_cycles(20000); // debounce  if (S1pressed == 0) {  S1pressed = 1;  current\_cipher = (current\_cipher + 1) % 4; // get current cipher selection (1-3)  UART\_send\_string(COLOR\_GREEN);  if (current\_cipher == 1) {  UART\_send\_string("\r\nCaesar Cipher\r\n\r\n");  TB0CCR0 = 32767; // 0.5 Hz -> f = ACLK (32kHz)/CCRO/2  } else if (current\_cipher == 2) {  UART\_send\_string("\r\nSubstitution Cipher\r\n\r\n");  TB0CCR0 = 16384; // 1 Hz  } else if (current\_cipher == 3) {  UART\_send\_string("\r\nXOR Cipher\r\n\r\n");  TB0CCR0 = 8192; // 2 Hz  }  UART\_send\_string(COLOR\_RESET);  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  P1IES &= ~BIT0; // P1IES.BIT0 low/high edge  }  } else if (S1pressed == 1) {  \_\_delay\_cycles(20000);  if (S1pressed == 1) {  S1pressed = 0;  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  P1IES |= BIT0; // P1IES.BIT0 hi/low edge  }  }  }  else if (P1IFG&BIT1) {  if (S2pressed == 0) {  \_\_delay\_cycles(20000);  if (S2pressed == 0) {  S2pressed = 1;  UART\_send\_string("\r\nEnter a string to encrypt:\r\n");  UART\_get\_word(input, sizeof(input)); // get input from user  if (current\_cipher == 1) {  UART\_send\_string("\r\nOutput caesar string:\r\n");  caesarCipher(input);  } else if (current\_cipher == 2) {  UART\_send\_string("\r\nOutput substitution string:\r\n");  substitutionCipher(input);  } else if (current\_cipher == 3) {  UART\_send\_string("\r\nOutput xor string:\r\n");  xorCipher(input);  }  UART\_send\_string(input);  UART\_send\_string(COLOR\_BLUE);  UART\_send\_string("\r\n\r\nPress switch 1 to toggle through cipher options\r\nPress switch 2 to enter cipher\r\n");  UART\_send\_string(COLOR\_RESET);  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  P1IES &= ~BIT1; // P1IES.BIT1 low/high edge  }  } else if (S2pressed == 1) {  \_\_delay\_cycles(20000);  if (S2pressed == 1) {  S2pressed = 0;  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  P1IES |= BIT1; // P1IES.BIT1 hi/low edge  }  }  }  } |
| --- |

The only external connection required other than the JTAG was the UART cable. The TX goes to P2.4 and RX to P2.5.