Digital Computer Structure – Final Report Lab 4

By:

Nachman Mimoun 321730558

Yarden Levi, 206212276

במטלת זמן האמת נתבקשנו להוסיף **STATE 7** ל- FSM הקיים שלנו שבו עלינו לבצע הצגת קוד ASCII של תווים a-z על גבי מסך LCD. המשימה כוללת קליטת תו מהמשתמש דרך תקשורת UART והצגת הערך הבינארי של קוד ה ASCII-המתאים על המסך.

המערכת מבוססת על תקשורת UART בין מחשב PC ובקר MSP430G2553, עם ארכיטקטורת FSM.

הגדרנו בקוד הראשי (main.c) את המשתנים הגלובליים הנדרשים:

```
char ascii_char = 0; // Store received character
int ascii_display_flag = 0; // Flag to indicate new character received
```

2. State 7 Implementation in Main Loop

```
 case state7:                          // NEW: Display ASCII code
       display_ascii();
       break;
```

3. UART RX ISR Enhancement

הרחבנו את פונקציית ה- ISR של UART RX לטיפול ב - state 7:

```
else if (UCA0RXBUF == '7' && !X_flag){       // If 7 is pressed, ASCII display
     state = state7;
     ascii_display_flag = 0;
     IE2 |= UCA0TXIE;
 }

 else if (state == state7 ){ // ASCII character input in state 7
      ascii_char = UCA0RXBUF;
      ascii_display_flag = 1;
      IE2 |= UCA0TXIE;
  }
```

4. UART TX ISR Enhancement

הוספנו טיפול ב state 7-ב:TX ISR-

```
else if (state == state7){       // ack print ascii
        UCA0TXBUF = '7'; //
    }
```

5. ASCII Display Function

מימשנו פונקציה חדשה `display_ascii()` ב- API layer:

```c
//-------------------------------------------------------------
//                      Display ASCII Code (State 7)
//-------------------------------------------------------------
void display_ascii(){
    while(state == state7){
        enable_UARTRX_interrupts();
        enterLPM(mode0);  // Wait for character input

        if(ascii_display_flag && state == state7){
            char ascii_str[4];
            unsigned int ascii_value = (unsigned int)ascii_char;
            char lower, upper;

            // Convert ASCII value to string
            ascii_str[0] = (ascii_value / 100) + '0';        // Hundreds
            ascii_str[1] = ((ascii_value / 10) % 10) + '0';  // Tens
            ascii_str[2] = (ascii_value % 10) + '0';         // Units
            ascii_str[3] = '\0';                             // Null terminator

            // Display on LCD
            lcd_home();

            lcd_puts(ascii_str);


        }

    }
    ascii_display_flag = 0;  // Reset flag
    lcd_clear();
}
```

6. Header Files Updates

עדכנו את קבצי ה- header הרלוונטיים:

api.h:

```c
extern void display_ascii();
```

halGPIO.h:

```c
extern char ascii_char;
extern int ascii_display_flag;
```

app.h:

```c
Enum
FSMstate{state0,state1,state2,state3,state4,state5,state6,state7,state8
,state9};
```

7. PC-Side Application Enhancement

עדכנו את האפליקציה בצד המחשב (main_real.py) לתמיכה בתפריט החדש:

```
menu = "\n                              Menu\n" \
       "1. Counting onto the LCD screen with a delay of X[ms]\n" \
       "2. Circular tone series via Buzzer with delay of X[ms]\n" \
       "3. Get delay time X[ms]:\n" \
       "4. Potentiometer 3-digit value [v] onto LCD\n" \
       "5. On each PB1 pressed, send a Massage \"I love my Negev\"\n" \
       "6. Clear LCD screen\n" \
       "7. Get a char a-z and show its ASCII code onto LCD\n" \
       "8. Show menu\n" \
       "9. Sleep\n"
```

Operation Flow

התהליך כולו עובד כך:

1. **Menu Selection:** המשתמש בוחר אפשרות 7 מהתפריט במחשב
2. **State Transition:** הבקר עובר ל- state7 ומאשר המעבר למחשב
3. **Character Input:** המשתמש מקיש תו במקלדת המחשב (a-z)
4. **UART Reception:** הבקר קולט את התו דרך UART RX ISR
5. **ASCII Conversion:** הפונקציה `display_ascii()` ממירה את הערך הבינארי של ה-ASCII לשלושה ספרות
6. **LCD Display**: הערך מוצג על מסך ה-LCD בפורמט של 3 ספרות
7. **Continuous Operation:** המערכת ממשיכה לקלוט תווים עד לבחירת אפשרות אחרת מהתפריט

Key Features

- **Real-time Response**: תגובה מיידית לקלט מהמשתמש
- **Interrupt-Driven**: שימוש בפסיקות לביצועים אופטימליים
- **Thread-Safe**: שימוש ב flags-למניעת race conditions
- **User-Friendly:** הצגה ברורה של קוד ASCII בפורמט 3 ספרות
- **State Management:** ניהול נכון של מעברי מצבים במערכת

Conclusion

המימוש מספק פתרון יעיל ואמין להצגת קודי ASCII על מסך LCD באמצעות תקשורת UART. השימוש בארכיטקטורת FSM מבוססת פסיקות מאפשר תגובה מהירה וחסכון בחשמל, תוך שמירה על קריאות הקוד וניידותו בין פלטפורמות שונות.

השיטה שפותחה מאפשרת הרחבה קלה למגוון תווים נוסף ויכולה לשמש כבסיס למערכות תקשורת מורכבות יותר.