מבנה מחשבים ספרתיים
361.1.4191

Final report LAB2

בר קופפרשמיד 318912193

רותם ארביב 208065052

תאריך הגשה : 23/04/2025

המשימה:

הוספת מצב חדש לקוד המערכת המבוסס מודל שכבות אבסטרקציה וגרעין הפעלה FSM מבלי פגיעה בתפריט המערכת הקיימת ותקינות המערכת לא תפגע.

המצב החדש יהווה את state4. המצב מוגדר כך שבלחיצה על לחצן PB3 המערכת נדרשת להפיק מרגל הבקר P2.2 אות PWM בתדר 1kHz עם ערך Duty Cycle המתעדכן כל שנייה לפי סדר הערכים הבא: 75%,50%,25% בצורה מחזורית.

המצב מוגדר להסתיים בלחיצה על לחצן של מצב אחר. הפקת האות נדרשת להתבצע בשימוש באופן עבודה Output compare במשפחה MSP430x2xx של Timer_A1.

דרך הפתרון:

הוספנו למערכת המצבים מצב חדש state4 למימוש המשימה. כניסה למצב זה מתרחשת בעת לחיצה על לחצן PB3 אותו חיברנו לרגל P2.1 וקינפגנו בהתאם בשכבת הbsp.

רגל P2.2 כבר מקונפגת להוצאת אות PWM מטיימר A1 אז יש לקנפג את רגל P2.1 ללחצן PB3:

יש לקבוע את ערכי SEL,DIR,IES,IE,IFG עבור רגל P2.1 להיות 0,0,0,1,0 בהתאמה על מנת לקבוע את הרגל להיות I/O במצב input ומצב עבודה pull-down ולאפשר פסיקות ולנקות פסיקות קיימות.

בנוסף יש לעדכן את הISR של PORT2 על מנת לאפשר טיפול נכון בעת לחיצה על כפתור PB3.

<u>קינפוג בשכבת הbsp:</u>

bsp.h:

```
// PushButtons BP3 abstraction
#define PBsArrPort3          P2IN
#define PBsArrIntPend3       P2IFG
#define PBsArrIntEn3         P2IE
#define PBsArrIntEdgeSel3    P2IES
#define PBsArrPortSel3       P2SEL
#define PBsArrPortDir3       P2DIR
#define PB3                  0x02      // Connects to P2.1

```

bsp.c:

```
                                        // PB3
  PBsArrPortSel3 &= ~0x02;              // I/O
  PBsArrPortDir3 &= ~0x02;              // Input
  PBsArrIntEdgeSel3 &= ~0x02;           // pull-down mode
  PBsArrIntEn3 |= 0x02;
  PBsArrIntPend3 &= ~0x02;               // clear pending interrupts
```

עדכון הISR של PORT2 בשכבת הHAL:

```
//*********************************************************************
//            Port2 Interrupt Service Routine - switch & PB3
//*********************************************************************
  #pragma vector=PORT2_VECTOR
    __interrupt void SWs_handler(void){
        delay(debounceVal);
        if (SWsArrIntPend & SW0){       // SW0
            SWsArrIntPend &= ~SW0;      // Turn off IFG
            __no_operation();
        }
        if(PBsArrIntPend3 & PB3){
            state = state4;
            PBsArrIntPend3 &= ~PB3;
            LPM0_EXIT;
        }
    }
```

הגדרת המצב בmain ובapi:

```
        case state4:    // Real time state
            state4Logic();
            break;
```

בעת כניסה למצב מופעלת פונקציית הstate4Logic שנמצאת בשכבת הapi:

```
//-------------------------------------------------------------------
//              State 4 Logic
//-------------------------------------------------------------------
void state4Logic(){
    unsigned int n = 1111;

    TIMERS4start(); // Start timer A1
    TIMERS2start(); // Start timer A0 in Up-Down mode

    while (state == state4) {
        TimerPWMUpdateDuty(n>>2);
        enterLPM(lpm_mode);     // Go to sleep
        if (state != state4){   // Check if state had changed
            break;
        }
        TimerPWMUpdateDuty(n>>1);
        enterLPM(lpm_mode);     // Go to sleep
        if (state != state4){   // Check if state had changed
            break;
        }
        TimerPWMUpdateDuty(n - (n>>2));
        enterLPM(lpm_mode);     // Go to sleep
    }
    TIMERS2stop();  // Stop timer A0
    TIMERS4stop();  // Stop timer A1
}
```

ראשית נסביר את אופן פעולת הטיימרים והפונקציות שהגדרנו עבורם:

פונקציות הטיימרים שבשימוש:

במימוש state3 בעבודת המכין קינפגנו את טיימר A1 כך שיוצא מרגל P2.2 אות PWM ומצאנו כי יש להכניס ערך של 1111 במעטפת TA1CCR0 על מנת שתדר הPWM יהיה 1kHz. בstate3 קבענו את ערך הDuty Cycle של האות באמצעות הכנסת ערך למעטפת TA1CCR1. לכן, בדומה לפונקציית הפעלת הטיימר עבור state3 יצרנו בשכבת הHAL פונקציית הפעלה, פונקציית כיבוי ופונקציית שינוי הDuty Cycle של טיימר A1 עבור state4:

```
//-----------------------------------------------------------------------
//                Timer State 4
//-----------------------------------------------------------------------
void TIMERS4start(void){
    TA1CTL |= MC_1 + TACLR;
    TA1CCTL1 = OUTMOD_7;
    TA1CCR0 = 1111;
}
void TIMERS4stop(void){
    TA1CTL &= ~ MC_3;
    TA1CCTL1 = OUTMOD_0;
}
void TimerPWMUpdateDuty(unsigned int d){
    TA1CCR1 = d;
}
```

פונקציית TIMERS4start מפעילה את TimerA1 במצב Up mode ו-OUTPUT במצב Reset/Set. ובנוסף, קובעת את ערך המעטפת TA1CCR0 להיות 1111. פונקציית TIMERS4stop מכבה את הטיימר ומחזירה את OUTPUT להיות במצב OUT bit.

בנוסף, פונקציית TimerPWMUpdateDuty מעדכנת את ערך TA1CCR1 לערך d שאותו היא מקבלת – באמצעות הפונקציה הזו אנחנו שולטות על הDuty Cycle כיוון שמתקיים:

$$Duty\ Cycle\% = \frac{TA1CCR1}{TA1CCR0} \cdot 100$$

זכרנו כי את state2 מימשנו באמצעות שימוש בטיימר A0 שקונפג לבצע פסיקה כל שנייה ולכן השתמשנו בו ופונקציית ההפעלה שלו עבור מצב state2 גם במצב זה עם אותו הקינפוג בדיוק:

```
//-----------------------------------------------------------------------
//                Timer State 2
//-----------------------------------------------------------------------
void TIMERS2start(void){
    TA0CCTL0 |= CCIE;
    TA0CTL |= MC_3 + TACLR; // Clear the register and start in Up/Down mode
}
void TIMERS2stop(void){
    TA0CTL &= ~MC_3;        // Stop timer to save power
    TA0CCTL0 &= ~CCIE;      // Disable envelope interrupts
}
```

הפעלה במצב Up/Down mode, כאשר בשכבת הbsp קבענו את ערך המעטפת להיות:

```
// State 2
TA0CTL = TASSEL_2 + ID_3 + TACLR; // SMCLK input divided by 8, cleared current value
TA0CCR0 = 0x8FFF;
```

שעבורו נקבל פסיקה כל שנייה.

הסבר הלוגיקה בשכבת הapi:

אז בכניסה למצב נפעיל את הטיימרים (A1 שמייצר את אות הPWM וA0 שמייצר פסיקה כל שנייה). כל עוד המצב לא התחלף נבצע את רצף הפעולות הבא:

1. נעדכן את ערך המעטפת TA1CCR1 להיות 1111/4 וכך נקבל Duty Cycle של 25%.
2. נכנס למצב שינה.
3. ברגע שתתקיים פסיקה (מטיימר A0 כי עברה שנייה או מלחיצה על כפתור) נתעורר.
4. נבדוק כי המצב לא השתנה – במידה וכן נצא מהלולאה, נכבה את הטיימרים ונחליף את המצב.
5. במידה ולא השתנה המצב נעדכן את ערך המעטפת TA1CCR1 להיות 1111/2 כך נקבל Duty Cycle של 50%.
6. נכנס למצב שינה.
7. ברגע שתתקיים פסיקה (מטיימר A0 כי עברה שנייה או מלחיצה על כפתור) נתעורר.
8. נבדוק כי המצב לא השתנה – במידה וכן נצא מהלולאה, נכבה את הטיימרים ונחליף את המצב.
9. במידה ולא השתנה המצב נעדכן את ערך המעטפת TA1CCR1 להיות 1111-1111/4 כך נקבל Duty Cycle של 75% (1-0.25=0.75).
10. נכנס למצב שינה.
11. בפעם הבאה שנתעורר - תתקיים פסיקה (מטיימר A0 כי עברה שנייה או מלחיצה על כפתור), נחזור לבדיקת תנאי הלולאה (המצב נשאר state4) ואם מתקיים התנאי נחזור חלילה.
12. אם התנאי לא מתקיים (המצב לא state4) נכבה את הטיימרים ונחליף את המצב.