Lab1 Preparation Report

יהונתן ארמא 207938903

יובל סעיד 206921892

PART B:

1) PxDIR – קובע את הכיוונים של הרכיב – Input או Output:
   - 0- Input
   - 1- Output

   PxSEL – קובע את פונקציונליות הרגל של הבקר – IO או חומרה אחרת שאפשר לקנפג לרגל:
   - 0- GPIO(Default)
   - 1- Other Hardware

   PxIN – כאשר אנחנו במצב של I/O במצב של input אז את ערך ה-input אנו מקבלי מרגיסטר זה:
   - 0- Low Value
   - 1- High Value

   PxOUT – כאשר אנחנו במצב של I/O במצב של output אז לרגיסטר זה נכתוב את הערך שאותו נרצה להוציא לרכיב המחובר:
   - 0- Low Value
   - 1- High Value

2) כאשר מבצעים reset:
   a. ראשית ה-PC נטען בכתובת הראשנה של התוכנית.
   b. הרגיסטרים נטענים באופן הבא:
      i. PxSEL -reset with PUC
      ii. PxDIR – reset with PUC
      iii. PxOUT – Untouched (כלומר נשאר מה שהיה הערך לפני ה-reset).
      iv. PxIN – אינו נקבע על ידי הבקר ולכן לא רלוונטי.

      נבחין ש-PUC – Power Up Clear.

      כלומר reset with PUC אומר שבזמן reset הערך מתאפס ל-0.

3) נרצה לכתוב ל- P9SEL את הערך כדי שכל הכניסות יהיו במצב I/O, ובנוסף נרצה ב-P9DIR שהכניסות הזוגיות יהיו ב-'1' (Output) והאי זוגיות ב-'0' (Input), לכן נכתוב:

```
BIC.B #0xFF, &P9SEL
MOV.B #0xAA, &P9DIR
```

4) אם ה-DudyCycle הוא 50% אז נרצה שהערך יהיה '1' למשך 0.5ms, בנוסף תדר ה-MCLK הוא 1MHz ולכן:

$$n = \frac{0.5 \cdot 10^{-3}}{2^{-20}} = \frac{1}{2} \cdot \frac{2^{20}}{10^3} = \frac{2^{19}}{10^3} \approx 524$$

5) פסיקה היא אות חשמלי המתקבל במעבד על מנת להודיע לו על פעולות אסינכרוניות שהתבצעו.
כלומר, ברגע שהמעבד שלנו מקבל – Interrupt מרגיסטר שאנחנו מתייחסים אליו (כלומר מקונפג כמו שצריך לאפשרות לפסיקה ואנחנו לא מתעלמים ממנו), אז המעבד שלנו 'קופץ' לפונקציה ייעודית לפסיקה ומבצע אותה, לאחר ביצוע הפונקציה, המעבד יחזור לשורה ממנה הוא קפץ. אנו צריכים פסיקות מכיוון שיש לנו חומרות או אינפוטים לבקר שלנו שמבצעים פעולות במקביל לבקר, ואנחנו נרצה להודיע לעדכן את המעבד שלנו בצורה אסינכרונית על דברים אלו, וזו הדרך לעשות זאת. בנוסף, לעיתים יש צורך לתעדף פעולות חשובות יותר או פחות, ואנו נעזרים בפסיקות על מנת לבצע את התיעדוף הזה. לדוגמה לחיצה על מקלדת או עכבר, או אפילו לחיצה על כפתור ה-reset, דורשים מאיתנו לבצע -interrupt כי אלו פעולות חשובות שאנו רוצים שהמעבד יטפל בהם עכשיו, ו'יעזוב' את מה שהוא עושה.

6) נזכור שבזמן תשאול, המעבד 'שורף' אנרגיה וזמן על בדיקה אם תנאי מסוים התקיים ובזמן זה אינו יכול לבצע שום דבר במקביל.
אם אנחנו נשתמש בפסיקה ונבקש באופן אקטיבי מהרכיב ש'יודיע' לנו בעזרת פסיקה שהתנאי מתקיים, אז המעבד יוכל לבצע מה שהוא רוצה בזמן הזה (ואולי אפילו לישון ולחסוך אנרגיה), וכאשר התנאי מתקיים אז הוא יבצע את הפעולה שהוא נדרש, מבלי לבזבז זמן ואנרגיה על בדיקת התנאי כל הזמן.
שילוב בין השניים יתקיים לדוגמה כאשר קבלנו כמה פסיקות ואנחנו נבצע תשאול לגבי איזה תנאים התקיימו.

7) ראשית נבחין בשני סוגים של פסיקות:
   a. לא ניתנות למיסוך – פסיקות בעדיפות גבוהה שלא ניתן לקנפג אותן להיות ממוסכות:

Table 5. Interrupt Sources, Flags, and Vectors

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	WORD ADDRESS	PRIORITY
Power-Up External Reset Watchdog Timer+ Flash key violation PC out-of-range(1)	PORIFG RSTIFG WDTIFG KEYV(2)	Reset	0FFFEh	31, highest
NMI Oscillator fault Flash memory access violation	NMIIFG OFIFG ACCVIFG(2)(3)	(non)-maskable (non)-maskable (non)-maskable	0FFFCh	30

פסיקות אלו הן הפסיקות בעדיפות הכי גבוהה ולכן המעבד עוזב הכל ומטפל בהן קודם. דוגמה – Reset.

b. ניתנות למיסוך – פסיקות שנגרמות מגורמים חיצוניים, וניתן למסך אותן באופן לוקלי או גלובלי.

Timer1_A3	TA1CCR0 CCIFG(4)	maskable	0FFFAh	29
Timer1_A3	TA1CCR2 TA1CCR1 CCIFG, TAIFG(2)(4)	maskable	0FFF8h	28
Comparator_A+	CAIFG(4)	maskable	0FFF6h	27
Watchdog Timer+	WDTIFG	maskable	0FFF4h	26
Timer0_A3	TA0CCR0 CCIFG(4)	maskable	0FFF2h	25
Timer0_A3	TA0CCR2 TA0CCR1 CCIFG, TAIFG (5)(4)	maskable	0FFF0h	24
USCI_A0/USCI_B0 receive USCI_B0 I2C status	UCA0RXIFG, UCB0RXIFG(2)(5)	maskable	0FFEEh	23
USCI_A0/USCI_B0 transmit USCI_B0 I2C receive/transmit	UCA0TXIFG, UCB0TXIFG(2)(6)	maskable	0FFECh	22
ADC10 (MSP430G2x53 only)	ADC10IFG(4)	maskable	0FFEAh	21
			0FFE8h	20
I/O Port P2 (up to eight flags)	P2IFG.0 to P2IFG.7(2)(4)	maskable	0FFE6h	19
I/O Port P1 (up to eight flags)	P1IFG.0 to P1IFG.7(2)(4)	maskable	0FFE4h	18
			0FFE2h	17
			0FFE0h	16
See (7)			0FFDEh	15
See (8)			0FFDEh to 0FFC0h	14 to 0, lowest

כאלו יש שלושה סוגים:

i. פסיקה חיצונית - פסיקה הנגרמת על ידי רכיב חומרה חיצוני כמו מקלדת או עכבר.
ii. פסיקה פנימית – פסיקה הנוצרת על ידי החומרה או התוכנה בעקבות אירוע פנימי של הבקר.
iii. פסיקת תוכנה – פסיקה המתעוררת בעקבות הדלקת דגל מהתוכנה.

8) נבחין שבמצב הרגיל המעבד שלנו מבצע את הפקודות ב-flash באופן סריאלי מצב זה נקרא I_{AM}, המצב האקטיבי של הבקר. לעיתים התוכנית שלנו מאופיינת בכך שיש בה חלקים שבה נרצה שהבקר 'יחכה' או יחסוך באנרגיה. לטובת זה יש דגלים תחת השמות – CPUOFF, OSCOFF, SCG0, SCG1, שבעזרתם נוכל לקבוע באיזה מצב חיסכון באנרגיה נהיה:

$$I_{LM0}, I_{LM2}, I_{LM3}, I_{LM4} \dots$$

תיאור:

PARAMETER		TEST CONDITIONS	T_A	V_{CC}	MIN	TYP	MAX	UNIT
$I_{LPM0,1MHz}$	Low-power mode 0 (LPM0) current(3)	f_{MCLK} = 0 MHz, f_{SMCLK} = f_{DCO} = 1 MHz, f_{ACLK} = 32768 Hz, BCSCTL1 = CALBC1_1MHZ, DCOCTL = CALDCO_1MHZ, CPUOFF = 1, SCG0 = 0, SCG1 = 0, OSCOFF = 0	25°C	2.2 V		56		µA
I_{LPM2}	Low-power mode 2 (LPM2) current(4)	f_{MCLK} = f_{SMCLK} = 0 MHz, f_{DCO} = 1 MHz, f_{ACLK} = 32768 Hz, BCSCTL1 = CALBC1_1MHZ, DCOCTL = CALDCO_1MHZ, CPUOFF = 1, SCG0 = 0, SCG1 = 1, OSCOFF = 0	25°C	2.2 V		22		µA
$I_{LPM3,LFXT1}$	Low-power mode 3 (LPM3) current(4)	f_{DCO} = f_{MCLK} = f_{SMCLK} = 0 MHz, f_{ACLK} = 32768 Hz, CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 0	25°C	2.2 V		0.7	1.5	µA
$I_{LPM3,VLO}$	Low-power mode 3 current, (LPM3)(4)	f_{DCO} = f_{MCLK} = f_{SMCLK} = 0 MHz, f_{ACLK} from internal LF oscillator (VLO), CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 0	25°C	2.2 V		0.5	0.7	µA
I_{LPM4}	Low-power mode 4 (LPM4) current(5)	f_{DCO} = f_{MCLK} = f_{SMCLK} = 0 MHz, f_{ACLK} = 0 Hz, CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 1	25°C	2.2 V		0.1	0.5	µA
			85°C			0.8	1.7	

9) נרצה להגדיר את הדברים הבאים:

a. נרצה שהרגל תהיה מסוג I/O ולכן נקבע את רגל 0 של P2SEL ל-0.
b. נרצה שהרגל תיתן לנו פסיקה בירידת מתח ולכן נגדיר אותה כ- Input, ולכן נקבע את רגל 0 של P2DIR ל-0.
c. נרצה שהפסיקה תתקבל מ-PullUp, ולכן נקבע את רגל 0 של P2IES ל-1.
d. נרצה לאפשר פסיקה מרגל זו, ולכן נקבע את רגל 0 של P2IE ל-1.
e. נוריד את דגל הפסיקה (נאפס את הפסיקה), ולכן נאפס את רגל 0 של P2IFG.

ולכן אלו הפקודות שנבצע:

```
BIC.B #0x01, &P2SEL
BIC.B #0x01, &P2DIR
BIS.B #0x01, &P2IES
BIS.B #0x01, &P2IE
BIC.B #0x01, &P2IFG
```

10) נעמוד על ההבדלים:

a. Event Driven – תכנות המבוסס פרדיגמה מסוג זה הוא תכנות שמבסס את התקשורת בין רכיבים שונים, בינם או מול המעבד מתבצעת ברמת התוכנה שהיא רמה גבוהה. כלומר, הטיפול בפעולות אסינכרוניות מתבצע בעיקר בעזרת Events, שנשמרים בבאפרים או בתורים.
b. Interrupt Driven – תכנות המבוסס פרדיכמה מסוג זה הוא תכנות המבסס את התקשורת בין רכיבים שונים, בינם או מול המעבד בעיקר בעזרת Interrupts, שזה כפי שאנחנו יודעים מתבצע ברמת החומרה. תכנות זה לדוגמה מתאים כאשר אנחנו מתכנתים מערכת שמתוארת בעזרת מכונה מצבים וה-Interrupts מגיעים ו-'משנים' את מצב התוכנית.
c. Blocking – פרדיגמה זו היא פרדיגמת תכנות בה הקוד מתבצע בצורה טורית ולא ניתן למקבל את התוכנית כלומר שורת הקוד הקודמת 'חוסמת' את שורת הקוד הבאה, ואם צריך לעמוד בתנאי מסויים בשביל להמשיך בתוכנית את המצב הזה 'חוסם' את המשך התוכנית.
d. Non-Blocking – פרדיגמה זו היא פרדיגמת תכנות שבה הקוד כתוב בצורה כזו שהוא לא 'חוסם' את המשך התוכנית, כלומר אם הפעולה לא מתבצעת עד הסוף, זה לא מונע מהמשך התוכנית להתבצע, בצורה זו ניתן 'למקבל' את התוכנית.

FSM