תוכן עניינים

LAB1 - System Programming, GPIO, Interrupts, LPM

A. חומר עזר – הכנה למעבדה:

מעבדה זו מבוססת על החומר הנלמד בקורס "מבוא למחשבים" משולב מעבדה צמודה (מעבדת מיקרו-מחשבים).

1. חומר עזר – GPIO:
   - ספר מעבדה MSP430x4xx user guide עמודים: 4**07-414 (ללא עמודים 411,412)**
   - חומר כתוב + סרטוני וידאו (Tutorial3.1 videos).

2. חומר עזר – GPIO interrupt + LPM:
   - בקובץ מעבדה MSP430x4xx user guide עמודים 45, 412 - 411
   - בקובץ MSP430xG461x datasheet עמודים 34-36
   - בקובץ מעבדה MSP430x4xx user guide עמודים 37-40
   - חומר כתוב + סרטוני וידאו (Tutorial 4.1 - Tutorial 4.3 videos).

3. חומר עזר – Code Example MSP430xG46x :

- **msp430xG46x_1.c :**
  Description: Toggle P9 by xor'ing P9 inside of a software loop.
- **msp430xG46x_P1_01.c :**
  Description: Poll P1.4 in a loop, if HI, P5.1 is set, if LOW, P5.1 clear.
- **msp430xG46x_P1_02.c :**
  Description: A high to low transition on P1.4 will trigger P1_ISR which, toggles P5.1.
- **msp430xG46x_P1_05.c :**
  Description: Writes a byte (FFh) to Port 1 and stays in LPM4
- **msp430xG46x_PA_05.c :**
  Description: Writes a Word (FFFFh) to Port A and stays in LPM4
- **Multi_files_project (this project is divided to layers – Bsp, Hal, App {main(), is on top this layer}:**
  Description:
  A high to low transition on P1.0 will trigger P1_ISR, which counts and shows its value onto LEDs.
  A high to low transition on P1.1 will trigger P1_ISR, which exits the CPU from sleep.
  The result:
  If the counting value onto LEDs (stored in a global variable ***BackgroundCnt***) greater than the constant ***Nval***, LEDs array is set and clear (with delay) else nothing happens.

הערה חשובה:

חומר עזר מתאים עבור בקר משפחה **MSP430G2xx3** נמצא במודל תחת הלשונית הבאה:

Personal Evaluation Kit

B. שאלות חלק תיאורטי:

1. רשום את תפקידם של הרגיסטרים $PxDIR,\ PxSEL,\ PxIN,\ PxOUT$
2. לאחר ביצוע RESET לבקר מהו מצב ברירת המחדל של הפורטים ומדוע?
3. רשום את השלבים לצורך קינפוג PORT9 למצב I/O , כאשר מבואות בעלי אנדקס זוגי במצב output ומבואות בעלי אנדקס אי-זוגי במצב input.
4. כדי לייצר במוצא של פורט כלשהו גל ריבועי במחזור של 1ms, כמה מחזורי שעון MCLK נדרשים להשהיה עבור חלק של '1' באות הריבועי ? נמק תשובתך
5. הסבר מהי פסיקה ועל הצורך בה.
6. הסבר את היתרון של שימוש בפסיקה (interrupt) לעומת תשאול (polling), מתי וכיצד נוכל לשלב בין השניים?
7. הסבר את שלוש סוגי הפסיקות ומה הצורך בכל סוג.
8. הסבר את מושג אופני העבודה של הבקר, הסבר כל אופן בנפרד ומתי תבחר להשתמש בו.
9. רשום את השלבים כדי לקנפג את רגל P2.0 כך שבירידת מתח מ-'1' ל- '0' תתבצע בקשת פסיקה.
10. הסבר את כל אחת מפרדיגמות תכנות הבאות ואת ההבדל ביניהן:

    Blocking, Non-blocking, event driven, interrupt driven

C. הקדמה ודרישות מקצועיות מחייבות לקראת ביצוע שלב החלק המעשי:

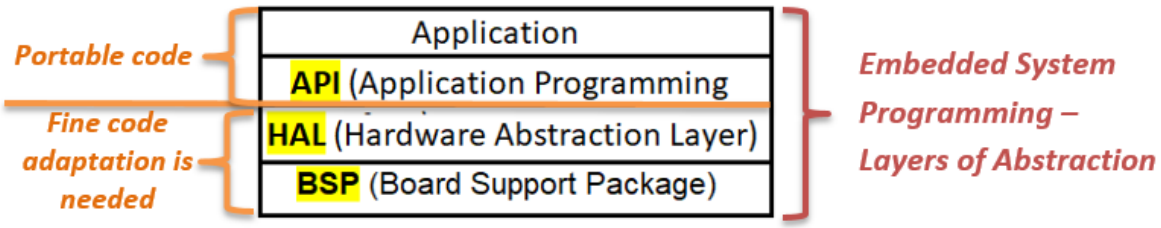
1. הקוד למימוש המערכת נדרש להיות בארכיטקטורת תוכנה FSM ומבוסס Interrupt Driven, כלומר טריגר למעבר בין מצבים במערכת נעשה כתוצאה מבקשות פסיקה (ראו Tutorial 4 pages 11-13) **ולא תחת מעטפת של לולאה אינסופית (הגורמת לבזבוז הספק ובנוסף מוגבלת מבחינת ארכיטקטורת התוכנה בהכללה ותחזוקה של קוד התוכנית) אלא בשימוש מצבי שינה של המעבד.**

הערה: גישה זו במערכת הכתובה על גבי מערכת הפעלה נקראת Event Driven.

2. **נדרש לארגן את הקוד בצורה מסודרת בקבצים נפרדים לצורך חלוקת קוד המערכת לשכבות הבאות:**

(תזכורת: דוגמה טכנית כיצד לבצע חלוקת קוד לקבצים נפרדים מופיעה בקוד לדוגמה)

- ✓ שכבת (Board Support Package) **BSP** מכילה קוד לקנפוג רגיסטרים של רכיבים פריפריאליים של הבקר (בניסוי מעבדה זו מדובר על קינפוג לדים, מתגים ולחצנים). **שם הקבצים בשכבה זו יהיו עם קידומת bsp, למשל bsp_example.c**
- ✓ שכבת ה- (Hardware Abstraction Layer) **HAL** מכילה רוטינות הדרייברים של המערכת המנהלות את הממשק עם הרכיבים הפריפריאליים של המערכת באופן ישיר (בניסוי מעבדה זו מדובר על רוטינה לכתיבת ערך כארגומנט למערך הלדים, רוטינה המחזירה ערך קריאה ממערך המתגים, רוטינת ISR של בקשת פסיקה ממערך הלחצנים). **שם הקבצים בשכבה זו יהיו עם קידומת hal, למשל hal_example.c**
- ✓ שכבת ה- (Application Programming Interface) **API** מכילה רוטינות על בסיסן אנו כותבים את האפליקציה של המערכת ב High Level תוך גישה לרכיבים פריפריאליים דרך API בלבד כאשר המימוש של השכבות מטה "שקוף" לשכבה זו, קוד זה צריך להיות portable כך שהוא יהיה תקף גם במידה וה- MCU של המערכת יתחלף באחר. **שם הקבצים בשכבה זו יהיו עם קידומת api, למשל api_example.c**
- ✓ שכבת ה- Application מכילה רוטינות שירות High level כגון חיפוש איבר במערך, מיון וכו' ומכילה את קוד ה- main היא שכבת קוד הגבוהה ביותר בה מתקיים הממשק עם המשתמש (מכילה את קוד מעטפת ה- FSM של המערכת) . **שם הקבצים של שכבה זו הם main.c, app_func.c**



הערה:

בכתיבת קוד גנרי המחולק לשכבות נוכל לבצע העברה קלה בין מערכת הכתובה עבור משפחה MSP430x4xx למערכת הכתובה עבור משפחה MSP430x2xx ולהיפך, ניהול המעבר מתחלק לשני מקרים:

i. *במקרה של מעבר בין משפחות של אותו שבב הבקר כאשר שתיהן מכילות את המודולים הפריפריאליים בשימוש המערכת אזי נדרש רק לעדכן את קובץ ה- BSP.*

ii. *במקרה שמשפחה אחת חסרה לפחות מודול פריפריאלי אחד הנמצא בשימוש המערכת אזי נדרש לעדכן את קובצי שכבות ה- BSP וה- HAL.*

3. העיקרון המרכזי בארכיטקטורת תוכנה FSM המבוססת Interrupt Driven – בקוד ה ISR של מקור בקשת הפסיקה (במעבדה זו בקשת פסיקה קוראת בלחיצה על אחד מהלחצנים) אנו מקבלים החלטה לעדכון ערך משתנה המצב ***state*** בלבד, ***בצורה זו אנו מעבירים מידע משכבת ה- HAL (המגיע אליה מהשכבה הפיזית = שכבת החומרה) היישר לשכבת ה- Application .***

4. **<u>הארה חשובה:</u>** החל מניסוי 2 ואילך, השהיות בקוד המערכת יהיו בשימוש טיימרים בלבד (חוץ מהשהיות נקודתיות שיוגדרו כיוצאי דופן) ולא כפי שנעשה בניסוי מעבדה 1 בשימוש לולאות for ל"שריפת" מחזורי שעון מעבד (הנקראת תשאול, polling).

D. סיווג ארכיטקטורת תכנות FSM לשני סוגים:

ארכיטקטורת תכנות FSM מחולקת לשני סוגים הבאים:

1. **מערכת Simple FSM :**

2. ***מעבר ממצב נוכחי (current_state) למצב הבא (next_state) אפשרי רק לאחר סיום קטע הקוד (המשימה) של המצב הנוכחי.*** כדי לתמוך בכך יש צורך להגדיר את קטע הקוד הנדרש להיות אטומי במצב הנוכחי כ critical section כדי שאף פסיקה לא תוכל "לחתוך" את קטע הקוד הזה, כלומר בתחילת קטע הקוד של המצב למסך גלובאלית את הפסיקות (GIE=0) ובסיום לאפשר אותן (GIE=1).

3. **מערכת Advanced FSM :**

מעבר ממצב נוכחי (current_state) למצב הבא (next_state) אפשרי גם במהלך ביצוע המצב הנוכחי במידה והמצב הבא הוא ברמת עדיפות גבוהה יותר.

כדי לתמוך בכך יש צורך בהגדרת מבני הנתונים הבאים:

i. משיקולי ביצועים תחת משטר Real Time את רמת העדיפות של המצבים נגדיר ע"י קידוד המצבים ברמת עדיפות עולה (מצב idle=0) ככל שערך קידוד עולה כך רמת העדיפות גבוהה יותר.
המשמעות, בכניסה ל ISR עקב בקשת פסיקה, מעבר למצב הבא יתבצע רק אם ערך המצב הבא גדול מערך המצב הנוכחי.

ii. במקרה שבו המצב הבא "חותך" את המצב הנוכחי במהלך ביצוע המצב הנוכחי עלינו לנהל זאת כך שבסיום, ביצוע המצב הנוכחי ימשיך מהמקום (ערך PC) וערכי ה- context (ערך data החיוני) בו "נחתך".

לצורך כך נגדיר שני מערכי נתונים:

- ✓ למצב i נגדיר את מערך $\boldsymbol{context_i}$ בגודל מוגדר מראש המוקצים עבור שמירת ה- context של הרגיסטרים הפריפריאליים הנדרש בכל אחד מהמצבים. למשל, בעבודה הכוללת הדפסה למערך הלדים נצטרך לשמור את ה context במבנה המכיל את $LEDs\ value, loop\ delay\ value$.
במצב זה תוכן מבנה $\boldsymbol{context_i}$ מכיל שלוש שדות ונכתוב לתוכו **<u>או</u>** נקרא מתוכו את הערכים בכל *context switch* .
- ✓ נגדיר מערך למימוש מבנה נתונים של מחסנית למימוש תור של ביצוע מצבים **exeQue .** מצב "שנחתך" יכנס לתור בשיטת LIFO לצורך המשך ביצוע.

להלן סיכום סדר הפעולות לביצוע:

i. בכניסה ל ISR עקב בקשת פסיקה, מעבר למצב הבא j יתבצע רק אם ערך המצב הבא j גדול מערך המצב הנוכחי i. במקרה זה, נבצע שמירת ה context של המצב הנוכחי (כתיבה למערך $\boldsymbol{context_i}$), ביצוע push (לא להתבלבל עם פקודת אסמבלי push של ה stack) של ערך המצב הנוכחי לתור **exeQue** ולבסוף עדכון משתנה המצב ***state*** לערך המצב הבא j.

ii. בסיום ביצוע מצב j ביצוע מצב i צריך להמשיך מהיכן "שנחתך" לכן נבצע את השלבים בצורה הפוכה, ביצוע pop (לא להתבלבל עם פקודת אסמבלי pop של ה stack) מהתור **exeQue** לתוך משתנה המצב ***state*** וטעינת תוכן $\boldsymbol{context_i}$ ל Register File .

iii. פעולת שלב *ii* תימשך עד לריקון התור **exeQue** .

E. כתיבת קוד מערכת גנרי בשפת C המחולק לשכבות אבסטרקציה ומבוסס גרעין הפעלה FSM:

הבהרות מקדימות:

- בחלק ההכנה היישומי נממש כתיבת קוד מערכת גנרי בשפת C המחולק לשכבות אבסטרקציה ומבוסס גרעין הפעלה FSM בין שתי משפחות מעבדים שונות.
- ההבדל בסעיף זה בין חלק 1 לחלק 2 הוא הגדרה שונה של שכבת ה- BSP בלבד.
- בניסויים הבאים נעבוד רק על משפחת MCU אחת (בניסויים 2,4 והפרויקט גמר תעבדו על ערכת פיתוח אישית מבוססת משפחה MSP430x2xx בניסוי 3 תעבדו על ערכת פיתוח הנמצאת במעבדה מבוססת משפחה MSP430x4xx מאחר ומודול DMA אינו מצוי במשפחה 2).

1. חלק 1 יישומי לביצוע – כתיבת קוד מערכת בשפת C (דרישה המתאימה לערכת הפיתוח האישית מבוססת שבב MCU של משפחה MSP430x2xx):

ארכיטקטורת התוכנה של המערכת נדרשת להיות מבוססת ***Simple FSM*** (ראה הסבר בסעיף E) המבצעת אחת מתוך ארבע פעולות בהינתן בקשת פסיקה חיצונית של לחיצת לחצן מתוך ארבעת הלחצנים PB3, PB2, PB1, PB0 המחוברים לארבעת רגלי הבקר **P2.0 – P2.3 , את מערך הלדים LEDs נחבר ל- PORT1 .**

בתחילת התוכנית, הבקר נמצא במצב שינה. קוד התוכנית נדרש להיות מחולק לשכבות (כמתואר בסעיף D). טרם שלב כתיבת הקוד נדרש לשרטט גרף דיאגרמת FSM מפורטת של ארכיטקטורת התוכנה של המערכת ולצרפה לדו"ח מכין. המצבים אלו הצמתים והקשתות אלו המעברים ממצב למצב בגין בקשות פסיקה.

- בלחיצה על לחצן PB0 (state=1):

נדרש בלחיצה ראשונה להדליק על גבי 8 הלדים ספירה בינארית כלפי **מעלה** ובלחיצה שנייה ספירה בינארית כלפי **מטה** וחוזר חלילה (בכל פעם המנייה תתבצע מהיכן שהפסיקה בפעם האחרונה, על כן נדרש לשמור *את ערך הכתיבה ללדים*). הספירה תהיה מחזורית עם השהיה בין ערכי הספירה של 0.5sec.

משך זמן הפעולה יהיה 10 שניות.

הערה: מצב אחר אינו רשאי "לחתוך" מצב זה טרם השלמת הביצוע המוגדר של המצב

- בלחיצה על לחצן PB1 (state=2):

נדרש להדליק לד בודד בדילוגים מימין לשמאל עם השהיה בין ערכי הספירה של 0.5sec.

משך זמן הפעולה יהיה 7 שניות (תוך שמירת ערך הכתיבה ללדים בחלוף הזמן, כך שבביצוע הבא של המצב הלד ימשיך לדלג מהיכן שהפסיק).

הערה: מצב אחר אינו רשאי "לחתוך" מצב זה טרם השלמת הביצוע המוגדר של המצב

- בלחיצה על לחצן PB2 (state=3):

התוכנית מפיקה אות PWM במוצא רגל P2.7 בתדר 4kHz עם DutyCycle=75% (ברזולוציה מקסימאלית – ודאו זאת בעזרת שימוש ב- scope).

הערה: מצב אחר רשאי "לחתוך" מצב זה (מאחר ופעולתו היא אינסופית) ובך לסיים אותו

- (state=idle=0):

הבקר מכבה את הלדים וחוזר למצב שינה (Sleep Mode).

2. חלק 2 יישומי לביצוע – כתיבת קוד מערכת בשפת C (דרישה המתאימה לערכת הפיתוח הנמצאת במעבדה מבוססת שבב MCU של משפחה MSP430x4xx):

ארכיטקטורת התוכנה של המערכת נדרשת להיות מבוססת ***Simple FSM*** (ראה הסבר בסעיף E) המבצעת אחת מתוך ארבע פעולות בהינתן בקשת פסיקה חיצונית של לחיצת לחצן מתוך ארבעת הלחצנים PB3, PB2, PB1, PB0 המחוברים לארבעת רגלי הבקר **P2.0 – P2.3 , את LEDs_A נחבר ל- PORT9.**

בתחילת התוכנית, הבקר נמצא במצב שינה.

קוד התוכנית נדרש להיות מחולק לשכבות (כמתואר בסעיף D).

טרם שלב כתיבת הקוד נדרש לשרטט גרף דיאגרמת FSM מפורטת של ארכיטקטורת התוכנה של המערכת ולצרפה לדו"ח מכין. המצבים אלו הצמתים והקשתות אלו המעברים ממצב למצב בגין בקשות פסיקה.

- בלחיצה על לחצן PB0 (state=1):

נדרש בלחיצה ראשונה להדליק על גבי 8 הלדים ספירה בינארית כלפי **מעלה** ובלחיצה שנייה ספירה בינארית כלפי **מטה** וחוזר חלילה (בכל פעם המנייה תתבצע מהיכן שהפסיקה בפעם האחרונה, על כן נדרש לשמור *את ערך הכתיבה ללדים*). הספירה תהיה מחזורית עם השהיה בין ערכי הספירה של 0.5sec.

- משך זמן הפעולה יהיה 10 שניות.

הערה: מצב אחר אינו רשאי "לחתוך" מצב זה טרם השלמת הביצוע המוגדר של המצב

- בלחיצה על לחצן PB1 (state=2):

נדרש להדליק לד בודד בדילוגים מימין לשמאל עם השהיה בין ערכי הספירה של 0.5sec.

משך זמן הפעולה יהיה 7 שניות (תוך שמירת ערך הכתיבה ללדים בחלוף הזמן, כך שבביצוע הבא של המצב הלד ימשיך לדלג מהיכן שהפסיק).

הערה: מצב אחר אינו רשאי "לחתוך" מצב זה טרם השלמת הביצוע המוגדר של המצב

- בלחיצה על לחצן PB2 (state=3):

התוכנית מפיקה אות PWM במוצא רגל P2.7 בתדר 4kHz עם DutyCycle=75% (ברזולוציה מקסימאלית – ודאו זאת בעזרת שימוש ב- scope).

הערה: מצב אחר רשאי "לחתוך" מצב זה (מאחר ופעולתו היא אינסופית) ובך לסיים אותו

- (state=idle=0):

הבקר מכבה את הלדים וחוזר למצב שינה (Sleep Mode).

הבהרות:

- **נדרש לארגן את הקוד בצורה מסודרת בקבצים נפרדים (קובצי source וקובצי header).**
- ערך תדר ברירת המחדל של שעון MCLK הוא:

$$f_{MCLK} = 32 \cdot 32768 = 2^{20} = 1,048,576\ Hz \rightarrow T_{MCLK} = \frac{1}{2^{20}} \approx 0.954\ \mu sec$$

E. צורת הגשה דוח מכין:

- הגשת מטלת דוח מכין תיעשה ע"י העלאה למודל של תיקיית zip מהצורה **id1_id2.zip** (**כאשר id1 < id2**),
רק הסטודנט עם הת"ז id1 מעלה את הקבצים למודל.
- התיקייה תכיל את שני הפרטים הבאים בלבד:
  - ✓ **קובץ Preface_lab$_1$.pdf** – מכיל תשובות לחלק ההקדמה של ניסוי LAB1
  - ✓ **קובץ Preparation_lab$_1$.pdf** – מכיל תשובות לחלק תיאורטי דו"ח מכין LAB1
  - ✓ תיקייה בשם **IAR** - מכילה שתי תיקיות, אחת של קובצי source (קבצים עם סיומת **.c***) והשנייה של קובצי header (קבצים עם סיומת **.h***).

הערה: תיקיית header צריכה להכיל שני קבצים שונים עבור שכבת bsp, קובץ bsp_msp430x2xx.h וקובץ bsp_msp430x4xx.h (בשאר השכבות השתמשו בהידור מותנה לצורך התאמת הגישה).

F. צורת הגשה דוח מסכם:

- הגשת מטלת דוח מכין תיעשה ע"י העלאה למודל של תיקיית zip מהצורה **id1_id2.zip** (**כאשר id1 < id2**),
רק הסטודנט עם הת"ז id1 מעלה את הקבצים למודל.
- התיקייה תכיל את שני הפרטים הבאים בלבד:
  - ✓ **קובץ final_lab$_x$.pdf** – מכיל תיאור והסבר לדרך הפתרון של מטלת זמן אמת.
  - ✓ תיקייה בשם **IAR** - מכילה שתי תיקיות, אחת של קובצי source (קבצים עם סיומת **.c***) והשנייה של קובצי header (קבצים עם סיומת **.h***).

הערה: תיקיית header צריכה להכיל שני קבצים שונים עבור שכבת bsp, קובץ bsp_msp430x2xx.h וקובץ bsp_msp430x4xx.h (בשאר השכבות השתמשו בהידור מותנה לצורך התאמת הגישה).

בהצלחה.