**Preparation report LAB 1**

**GPIO**

Dvir Zaguri – 315602284

Elad Hubashi 313612038

B. שאלות חלק תאורטי:

1. PxDIR – תפקידו להגדיר עבור פורט x (בין 1 ל-2) את כיווניות הפינים, זתומרת כל ביט ברגיסטר הזה מגדיר איזה מהפינים ברגיסטר הזה הוא אינפוט או אווטפוט כאשר ביט הוא '1' אזי הפין התואם ב-OUTPUT וכאשר הביט המתאים '0' אז הפין התואם יהיה INPUT. הרגיסטר מאפשר להגדיר על פין בבקר אם הוא INPUT או OUTPUT. מאחר וכמות הפינים הוא 8 אז נעבוד עם רגיסטר זה בפקודות של בייט בלבד.

PxSEL – רגיסטר שבריסט הוא מתאפס.ואכן צריך להיות מאופס עבור כל הפורטים x למיניהם שאיתם נרצה לעבוד במצב של O/I .

PxIN – במידה והגדרנו פינים בפורט x להיות אינפוט אזי ברגיסטר זה, נראה מה ערך המתח שכל אחד מהפינים שהוגדרו להיות אינפוט קוראים, זתומרת במידה והגדרנו את פין y להיות INPUT אזי אם ברגע נתון אנו רואים שערך הביט Y הוא '1' אזי מבינים שנופל עליו מתח של 3.3 וולט, ואם ערך הביט הוא '0' אזי נבין שנופל על הפין הנ"ל מתח קטן מכדי להיות '1' לוגי, כנראה קרוב ל-0 וולט.

PxOUT – במידה והגדרנו את הפין המסויים בפורט x להיות '1' לוגי אזי נוכל להגדיר באמצעות הביטים ברגיסטר זה את המתח שיהיה בפינים שהוגדרו כ-OUTPUT כאשר עבור כל פין שהוגדר כ-OUTPUT ערך הביט המתאים '1' אומר שיהיה בו מתח של '1' לוגי זתומרת 3.3 וולט, ו-'0' לוגי יגרור למתר של 0 וולט.

2. לאחר ביצוע RESET הרגיסטרים PxSEL מתאפסים כי זהו מצב ברירת המחדל של הפורטים, לשמש O/I עבור הבקר. הרגיסטרים PxOUT גם כן מתאפסים כדי שלא תהיה פגיעה לא רצויה ברכיבים פריפריאליים ולא יתקבל מתח איפה שלא רצוי. PxIN מושפע מרכיבים חיצוניים שמחוברים אליו כמובן לכן במידה ולא מחובר דבר ערכו יהיה 0 וכן PxDIR גם כן 0 מאותה הסיבה שPxOUT הוא 0.

3. השלבים הם:

א. בחירת הרגל שתשמש ל-O/I והזנת הרגיסטר P9SEL = 11111111

ב. בחירת איזה מהרגליים יהיו אינפוט ואיזה אווטפוט P9DIR = 01010101

ג. עכשיו אמנם ביצענו את הנדרש אך גם נרצה כמובן להגדיר מה יהיה הערך המתח ברגליים שהם OUTPUT אז כדאי להמשיך ולהגדיר גם את P9OUT.

4. תדר השעון MCLK של הבקר שלנו הוא 1MHz זתומרת שעבור השהייה של מחזור שעון אחד נקבל השהייה של 1/1,000,000 שניות, זתומרת 1 מיקרו שניה. לכן עבור 1 מילי שניה נרצה השהייה של 1,000 מחזורי שעון.

5. פסיקה היא אות חשמלי המתקבל בבקר ומורה לCPU שיש רכיב חומרתי שמצריך תשומת לב מיידית ולהיות המשימה העיקרית שלו כרגע. פסיקה היא איפשור שינוי סדר ביצוע הפקודות שכתבנו בבקר שלא ע"י בקרה מותנית. בעת פסיקה המחשב משהה את הביצוע הסדרתי של התוכנית כדי לצאת לרוטינת הפסיקה ובסיומה המחשב חוזר לאותה הנקודה בה עצר בדיוק. הצורך בפסיקות נובע מצורך של יוזר/מפתח לבצע פעולות מסויימות פעם בכמה זמן שאינן בתדירות קבועה.

6. polling היא שיטה שבה כל כמה זמן המערכת תשאל את המשתמש בה האם ירצה לבצע פעולה מסויימת/להריץ איזושהי רוטינת קוד שאינה רצה בלי בקשת המפעיל. לעומת זאת interrupt אף פעם לא שואל האם להתחיל פעולה אלא דומה למצב בו 'המערכת מוכנה לקריאה'. המצב הזה דומה למצב בוא ב-'polling' המערכת כל כמה זמן תשאל את המשתמש "אתה מוכן" ? ויכולה גם לשאול אותו המון פעמים ותלויה באינפוט של המשתמש. לעומת זאת interrupt זה מצב בו המערכת "תאמר" למשתמש : 'תגיד לי כשאתה מוכן'. חוסך את הצורך בתשומת הלב של המשתמש אך כופה עליו להיות יותר מדוייק בתזמון לעומת polling שבלעדי אישור של היוזר לא תוכל להמשיך. אלו היתרונות והחסרונות של שיטה אחת לעומת השניה בהקשר זה.

7. ישנן שתי סוגי פסיקות, פסיקות שאינן ניתנות למיסוך ושניתנות למיסוך. פסיקות הניתנות למיסוך דורשות טיפול מיידי ולא ניתן לדחות אותן לסיום הרוטינה הנוכחית. לרוב נועד לניהול תהליכים קריטיים כמו שגיאות בחומרה/מצבי קיצון אחרים. דוגמה לפסיקה זו היא אתחול הבקר עם כפתור ה-RESET.

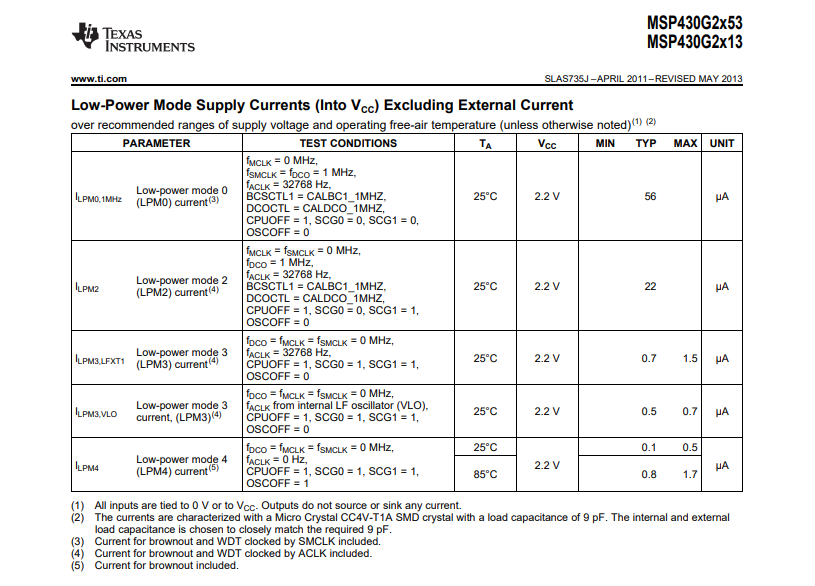
פסיקות הניתנות למיסוך הינן פסיקות הנגרמות ע"י רכיבים פריפריאליים. ישנן שלושה סוגים של פסיקות הניתנות למיסוך:

פסיקה חיצונית – נגרמת ע"י רכיב חומרה שאינו תלוי בריצת התוכנית הנוכחית בבקר.

פסיקה פנימית – נגרמת ע"י החומרה או תוכנה במועד ידוע מראש עקב ביצוע פקודה מסוימת בתוכנית. פסיקה זו נקראת "סינכרונית" משום שהיא צפויה.

פסיקת תוכנה – פסיקה המתעוררת בעקבות הדלקת דגל כלשהו בתוכנה.

8. אופני העבודה של הבקר הם :



אלו הם מצבי השינה של הבקר אותם ניתן להגדיר בשימוש לפי עבודתו בהמתנה לפסיקה.

9. כדי לקנפג את רגל P2.0 לביצוע פסיקה בירידת מתח נתחיל מקינפוג הרגיסטר P2IES להיות על 1 כדי להגדיר שהפסיקה תהיה בfalling edge (נזכיר שפעולות על רגיסטרים בסעיף זה הן פקודות על ביט (.b). לאחר מכן נרצה לאפשר את הפסיקה ע"י אפשור מקומי ואתחול הרגיסטר P2IE = 1 ובמידה ונרצה לראות את דגל הפסיקה (אם הפסיקה מחכה להתחיל ונדלקה אך בעדיפות נמוכה יותר מקטע הקוד שרץ כרגע) ברגיסטר P2IFG.

10.

Blocking – פעולה או קריאת פונקציה חוסמת את ביצוע התוכנית עד להשלמת הפעולה או לתנאי מסוים. במהלך זמן זה, ה-CPU בדרך כלל לא פעיל ומחכה לסיום הפעולה לפני שהוא ממשיך להוראה הבאה. בפרדיגמה זו, ביצוע התוכנית נעצר עד למילוי תנאי מסוים או לסיום הפעולה. זה בדרך כלל כרוך בתכנות סינכרוני, שבו משימות מבוצעות ברצף בסדר קבוע מראש.

Non-Blocking – בניגוד ל-Blocking, פעולות שאינן חוסמות מאפשרות לתוכנית להמשיך בביצוע גם בזמן שהפעולה מתבצעת. במקום להמתין, התוכנית בודקת את מצב הפעולה מעת לעת וממשיכה במשימות אחרות אם הפעולה לא הושלמה.

Event Driven – בתכנות מונחה אירועים, זרימת התוכנית נקבעת על ידי אירועים חיצוניים ולא סדרה רציפה של הוראות. אירועים אלה יכולים לכלול קלט משתמש, קריאות חיישנים או תפוגת טיימר. התוכנית מגיבה לאירועים על ידי ביצוע מטפלי אירועים מוגדרים מראש או התקשרויות חוזרות.

Interrupt Driven – תכנות מונע פסיקה מסתמך על היכולת של החומרה להפריע לזרימה הרגילה של ביצוע תוכנית כדי לטפל באירועים או אותות אסינכרוניים. כאשר מתרחשת פסיקה (כגון הצפת טיימר או קלט חיצוני), ה-CPU משעה זמנית את המשימה הנוכחית שלו וקופץ לרוטינת שירות פסיקה (ISR) כדי לטפל בהפרעה.