

PIC Tutorial

1	הקדמה
2	MPLAB
2	פתיחת פרויקט חדש
4	קימפול
5	ICD2
7	Bootloader
7	קימפול התוכנית
7	צריבת התוכנית
8	צריבת ה-Bootloader
9	I/O
12	Interrupts
14	Timer0
16	PWM
18	סיכום
18	טיפים

הקדמה

ה-PIC הוא מיקרובקר הניתן לתכנות ומבוסס על פסיקות (interrupts, הסבר בהמשך). קיימים דגמים שונים רבים הנבדלים ביכולות החישוביות שלהם, וכן בפונקציות ספציפיות המותאמות לצרכי בקרה שונים, כגון שליטה על מנועי סרוו, קריאה מ-encoders לצורך שיערוך מהירות ומיקום, מסי כניסות A/D וכו'. תכנות הבקר מתבצע, במקרה שלנו, בשפת C (אך עשוי להתבצע גם באמצעות BASIC, Assembler ותלוי רק ב-compiler הנתון) בסביבת MPLAB שפותחה לשם כך.

הדגם עימו נעבוד הוא PIC 18F4431, המותאם במיוחד לצרכי רובוטיקה ומאפשר שליטה על 4 מנועים במקביל וכן קריאה ממספר רב של גלאים שונים, דיגיטליים ואנלוגיים (9 ערוצי המרת A/D). זיכרון ה-flash שיעמוד לרשותנו לצורך התכנות גודלו 16384 bytes, וזכרון ה-RAM למידע גודלו 4096 bytes והוא מחולק ל-16 בנקי זכרון בני 256 bytes. מהירות השעון של המעבד נקבעת ע"י גביש אוסצילטור חיצוני וחסומה ע"י 40 MHz. לסיכום, בכדי להפיג את הזלזול, נציין כי זהו בקר חזק יחסית, המאפשר להגיע לתוצאות מרשימות ביותר ביחס לגודלו ולעלותו (כ-6 דולר).

תדריך זה נועד להכרות ראשונית עם ה-PIC וסביבת העבודה שלו. במהלכו יודגשו ההבדלים שבין תכנות ב-PC לבין תכנות ב-PIC, בצירוף מספר קטעי קוד לדוגמא. זאת מתוך הנחה כי הקוראים בעלי רקע בסיסי בשפת C.

MPLAB

סביבת MPLAB נועדה לניהול הפרויקט בכל הקשור לעריכת הקוד, קימפולו, דיבוגו וצריבתו על הבקר, ובאופן טבעי מזכירה סביבות אחרות כגון Visual Studio, כך שבמהרה תרגישו כמו בבית.

פתיחת פרויקט חדש

תפריט Project Wizard ← Project Wizard ← Next

בחרו את הרכיב עימו תעבדו PIC18F4431 ← Next

ודאו כי נבחרה האפשרות HI-TECH PICC-18 Toolsuite, וכי מיקומי הקבצים של הקומפילר,

האסמבלר והלינקר מצביעים על C:\HTSOFT\PIC18\bin\pic18.exe ← Next

בחרו את שם הפרויקט והמיקום לקבציו ← Next

תוכלו להוסיף קבצים קיימים לפרויקט שלכם באמצעות בחירתם בתיבה משמאל ← Next

עתה נותרה רק סידרה של הגדרות פשוטות והכרחיות – Configuration Bits – הקובעות את מצב

העבודה של הרכיב. לצורך כך נכנס לתפריט Configure ← Configuration Bits

להלן המלצה למצב עבודה:

Configuration Bit	Description	Mode
Oscillator	האוסצילטור קובע את השעון של המעבד. מאחר ואנו עובדים עם גביש חיצוני במהירות של 40 MHz יש לבחור ב-High Speed, כלומר HS.	HS
Fail-Safe Clock Monitor Enable	מפעיל שעון פנימי המשמש כגיבוי לשעון החיצוני במקרה של תקלה.	Disabled
Internal External Switch Over Mode	מאפשר איתחול של המערכת תחילה עם אוסצילטור פנימי, ולאחר שהאוסצילטור החיצוני נכנס לפעולה לעבור אליו.	Disabled
Brown Out Detect	מגלה נפילות מתח שעולות להשפיע על פעילות הבקר, בייחוד בעת הפעלת/כיבוי המתח. ומכבה את הבקר עד לעליית המתח. נבחר Brown Out Voltage: 4.5V	Enabled
Power Up Timer	מוסיף השהיה בין אתחול הבקר לבין תחילת הרצת התוכנית.	Enabled

	בשילוב עם האפשרות הקודמת: אם במהלך ההשהיה ישנה נפילת מתח נוספת, ה-PIC ישאר במצב reset עד להתייצבות המתח.	
Watchdog Timer	נועד למנוע כניסה ללולאות אינסופיות. לאחר סף כלשהו שנקבע ב-Watchdog Postscalar, מתבצע אתחול (reset) של הבקר. מומלץ לבטל זאת בכדי להימנע מאתחולים מסתוריים בשלב הפיתוח של התוכנה.	Disabled
Low Voltage Program	מאפשר את תכנות הבקר באמצעות מתחים נמוכים דרך הרגל RB5. יש לבטל זאת בכדי לעשות שימוש חופשי ביציאה זו. התכנות במקרה שלנו נעשה באמצעות RC 6,7 (serial port).	Disabled

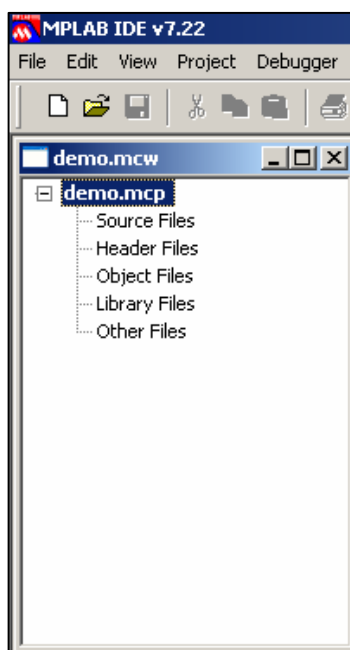
יתר הביטים אינם משמעותיים לענייננו, וניתן להתעלם מהם.

הערה: אם ברצונכם לוודא את בחירת הרכיב, תוכלו להיכנס לתפריט Configure ← Select Device

כעת ניתן להתחיל בתכנות. ליצירת קוד חדש בחרו בתפריט File ← New File
הקוד המינימלי החוקי שהקובץ שלכם יצטרך להכיל בכל מקרה הוא:

```
#include <pic18.h>

void main(void)
{
}
```



לאחר שמירת הקובץ תצטרכו להוסיף אותו בעצמכם לתוך הפרויקט. קבצי הפרויקט כולם מופיעים בתיבה משמאל. על-מנת להוסיף קובץ קיים לפרויקט לחצו בראש הרשימה בקליק-ימני ←
Add Files

קימפול

על-מנת לקמפל את התוכנית יש לבחור בתפריט Project ← Build All.

נשים לב לתיבת האפשרויות הענפות בתפריט Project ← Build Options ← Project בטאבים השונים:

Global: האופציה **Compile for MPLAB ICD** צריכה להיות מופעלת רק בעת עבודה עם הרכיב ICD2 (ראה פרק לעיל).

PICC-18 Compiler: רצוי להפעיל את **Enable assembler optimization** ולבחור Global optimization level: **9**.

PICC-18 Assembler: רצוי להפעיל את **Enable optimization**.

PICC-18 Linker: רצוי לבחור ב-**Integer Only** Printf support: שימו לב לאפשרות Specify offset for ROM (in hex): (ראה פרק (Bootloader).

לאחר סיום ההגדרות קמפלו את התוכנית. אם הוגדר הכל כראוי יתקבל את הטקסט הבא:

```
Clean: Deleting intermediary and output files.
Clean: Deleted file "Z:\helpdesk\demo\main1.obj".
Clean: Deleted file "Z:\helpdesk\demo\main1.cce".
Clean: Deleted file "demo.cof".
Clean: Done.
Executing: "C:\HTSOFT\PIC18\bin\picc18.exe" -C -E"main1.cce" "main1.c" -O"main1.obj" -Zg9
-O -ASMLIST -Q -MPLAB -18F4431
Executing: "C:\HTSOFT\PIC18\bin\picc18.exe" -E"demo.lde" "Z:\helpdesk\demo\main1.obj" -
M"demo.map" -O"demo.cof" -O"demo.hex" -Q -MPLAB -18F4431

Memory Usage Map:

Program ROM $000000 - $000003 $000004 ( 4) bytes
Program ROM $000014 - $00001B $000008 ( 8) bytes
              $00000C ( 12) bytes total Program ROM

Program statistics:

Total ROM used      12 bytes (0.1%)
Total RAM used      0 bytes (0.0%)   Near RAM used      0 bytes (0.0%)

Loaded Z:\helpdesk\demo\demo.cof.
BUILD SUCCEEDED: Tue Jul 04 13:40:17 2006
```

ICD2

לאחר שבידינו תוכנית מקומפלט כראוי עלינו לטעון אותה לרכיב. עומדות לרשותנו שתי דרכים לעשות זאת: האחת, באמצעות חומרת ICD2 בליווי PICDEMO Board (כרטיס עליו יושב הבקר בעת הצריכה) תוך שימוש בסביבת MPLAB; והשנייה באמצעות כרטיס ותוכנת Bootloader היושבת על הרכיב, ותקשורת סריאלית אל המחשב.

לפני העבודה יש לחבר את ה-ICD2 (דיסקה עגולה כחולה-אדומה) למחשב בכבל USB מצד אחד, ומצידה האחר לחבר אותה ל-PICDEMO Board "בכבל הטלפון" המצורף. נורת מתח ירוקה צריכה להידלק ב-ICD2 עם החיבור למחשב. את ה-PICDEMO Board יש להזין ב-9V (שנאי מצורף).

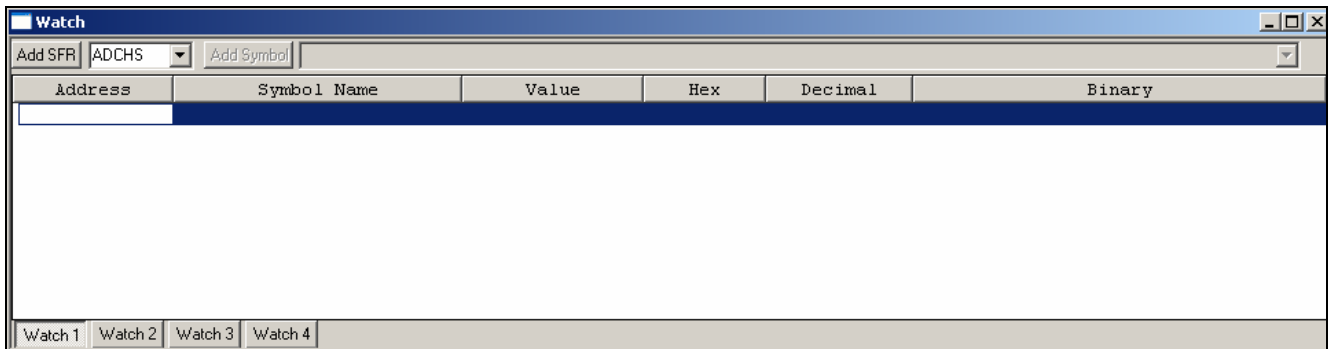
התמורה שמציע לנו רכיב ה-ICD2 לעלותו הגבוהה מתבטאת בשני מצבי עבודה (ללא חפיפה):
מצב Programmer: במצב זה הצריכה של התוכנה לבקר מאפשרת לו לפעול באופן עצמאי לגמרי. למן הרגע שהוא מחובר לספק מתח התוכנה שכתבתם תרוץ ותנהל את העניינים.

מצב Debugger: במצב זה התוכנה נצרכת לבקר, אולם הרצתה מתאפשרת רק כל עוד קיים ממשק בין הבקר לבין סביבת MPLAB דרך ה-ICD2 וה-PICDEMO Board. כתוצאה מכך ניתן להריץ (Run, F9) ולהפסיק (Halt, F5) את פעולת הבקר כאוות נפשכם, להריץ שורות בודדות (Step, F7/F8) או לעיין בזיכרון הבקר באין-מפריע (במצב Halt בלבד, באמצעות תפריט View ← Watch).

לכל אחד מן המצבים תפריט משלו. על-מנת לעבור ממצב אחד למשנהו יש להיכנס לתפריט-המצב הרצוי ← Select Programmer/Tool ← **MPLAB ICD 2** מסך התחברות לצורך יצירת הממשק יופיע ויתריע בפני תקלות אפשריות. ניתן לנסות ולהתחבר מחדש באמצעות Connect הנמצא בכל אחד משני התפריטים.

אם הכל תקין, כל שצריך לעשות הוא לבחור בפקודת Program בתפריט המתאים. פקודות ההרצה של ה-Debugger מופיעות בתפריט המיועד לו.

על-מנת להשתמש ב-Watch, נכנס לתפריט View ← Watch. יפתח החלון הבא:



תפריט האפשרויות השמאלי מאפשר לבחור ברגיסטרים השמורים של ה-PIC, ובאמצעות Add SFR להוסיף אותם לרשימת כתובות הזיכרון הנצפות. תפריט האפשרויות הימני מאפשר לבחור במשתנים גלובאליים שהגדרתם בתוכנית ולהוסיפם באמצעות Add Symbol. ערך ששונה לאחרונה יסומן באדום. שימו לב: לעתים יש לבצע רענון (קליק-ימני בטבלה ← Refresh) על-מנת לקבל תוצאות מעודכנות.

Bootloader

ה-Bootloader מאפשרת צריבה ל-PIC ללא תלות בחומרה נוספת, דרך ערוץ התקשורת הסריאלית. הדבר מתאפשר באמצעות תוכנה קטנה היושבת בתחילת זיכרון התוכנה של הבקר והיא הראשונה לרוץ לאחר כל אתחול של הבקר. כאשר היא עולה מתאפשרת הבחירה בין שליחת קוד מקומפל חדש בכדי שייצרב בזיכרון התוכנה במקום המיועד, לבין ההרצה של הקוד הקיים כעת בזיכרון. תוכנה זו היא ה-Bootloader.

הערה: לשם הבלבול אנו מכנים גם כל כרטיס כללי בעל שקע D-type לכבל סריאלי, עליו יושב ה-PIC במקום על ה-PICDEMO Board, בשם Bootloader. לפרטים נוספים, פנו אל ספק הכרטיסים שלכם (קובי את אורלי אנלימיטד).

קימפול התוכנית

כפי שהזכרנו, תוכנת ה-Bootloader נמצאת תמיד על זיכרון הבקר ומנהלת את טעינת התוכניות אליו. בגלל הקדימות שלה, היא ממוקמת בכתובות הזיכרון הראשונות של הבקר. לכן, כל תוכנית שנוסיף תצטרך להיות ממוקמת עמוק יותר בזיכרון. לשם כך יש לוודא כי קובץ ה-hex של התוכנית, המכיל את פקודות המכונה וכתובות הזיכרון המיועדות להן, יותאם לקראת צריבה באמצעות ה-Bootloader. על-כן:

היכנסו לתפריט Project ← Build Options ← PICC-18 Linker

וקיבעו Specify offset for ROM (in hex):

(זאת משום שגודלה של תוכנת ה-Bootloader בה אנו משתמשים הוא 300 מלים. נדגיש שוב כי בעת הצריבה באמצעות ערכת ICD2 אין צורך בהסחה).

צריבת התוכנית

כאמור, הצריבה נעשית בתקשורת סריאלית, ולכן ניתן לעשותה באמצעות כל טרמינל (בכל זאת רצוי להימנע מ-HyperTerminal, ומומלץ להשתמש ב-Terminal המצורף). את התוכנית נשלח בצורת קובץ hex המתקבל כפלט לאחר הקימפול ונמצא בתיקיית הפרויקט.

1. חברו את הכבל הסריאלי ל-COM1/2 ומצידו השני לכרטיס ה-Bootloader.
2. הפעילו את הטרמינל והתחברו. הפעילו את ספק המתח של ה-PIC.
3. על המסך יופיע הטקסט הבא :

```

HI-TECH Software (C)2002
Download-5
Download-4
Download-3
Download-2
Download-1

```

4. לאחר תום הספירה לאחר תוכנת ה-Bootloader עוברת להריץ את התוכנית הנוכחית השמורה בזיכרון. בהעדר כל תוכנית כזו (זיכרון ריק) הספירה תתחיל מחדש שוב ושוב. כל עוד הספירה לאחר מתבצעת ניתן לעצור אותה באמצעות שליחת בייט כלשהו מחלון ה-Transmit התחתון. בכך נסמן ל-Bootloader כי ברצוננו לשלוח תוכנית חדשה לצריבה. במידה והספירה הסתיימה לפני שהספקתם לשלוח את הבייט אתחלו את הבקר באמצעות כפתור reset שעל הכרטיס.
5. לאחר עצירת הספירה, לחצו על Send File ובחרו את קובץ ה-hex של תוכניתכם (בדרך כלל תחת השם [Project Name].hex). בייטים ירוצו על המסך בעת השידור. העברה מוצלחת תסתיים בהכרח בבייט 'ני'. מאפיין נוסף של צריבה שנכשלה : אתחול מחדש של הבקר (הופעת הודעת הפתיחה של ה-Bootloader).

צריבת ה-Bootloader

על-מנת לצרוב את תוכנת ה-Bootloader על הבקר יש להשתמש בערכת ICD2, כפי שהוצג בפרק הקודם.

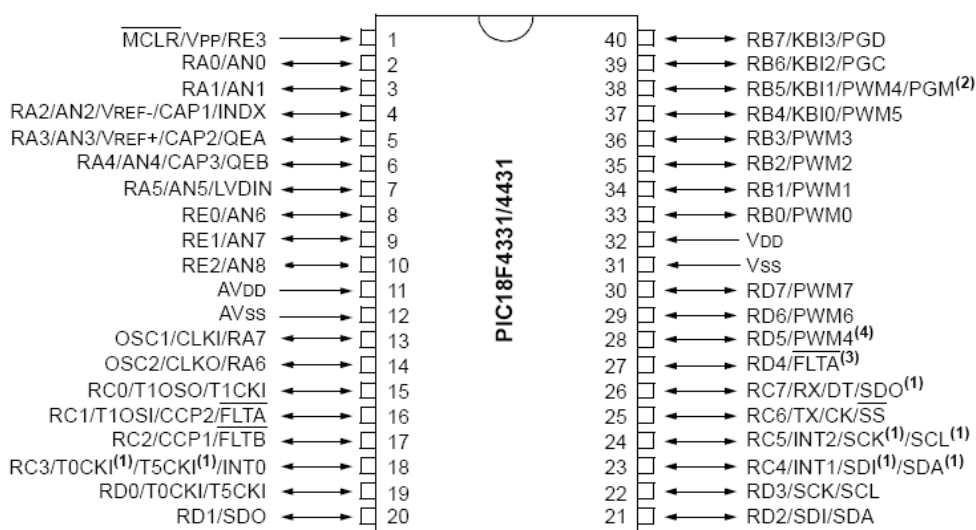
1. פיתחו פרויקט חדש והתאימו את הגדרותיו כבפרק הקודם.
2. בחרו בקובץ bootloader.hex, המצורף, בתפריט File ← Import File
3. בחרו את ICD2 בתפריט Programmer ← Select Programmer
4. לסיום לחצו על Programmer ← Program

I/O

לאחר ההקדמות המפרכות, הגיע הזמן לגשת לתכנות עצמו. ראשית כל, עליכם להצטייד בדף הנתונים של הבקר שלכם מאתר החברה: <http://www.microchip.com>. מסמך זה מכיל את כל המידע הנחוץ לשימוש בבקר, ותעשו בו שימוש תדיר.

בעמ' 4 של דף הנתונים תמצאו את סכמת הרגליים (Pin Diagram) של הבקר:

40-Pin PDIP



Note 1: RC3 is the alternate pin for T0CKI/T5CKI; RC4 is the alternate pin for SDI/SDA; RC5 is the alternate pin for SCK/SCL.

2: Low-voltage programming must be enabled.

3: RD4 is the alternate pin for \overline{FLTA} .

4: RD5 is the alternate pin for PWM4.

כל רגל של הבקר מיוצגת ע"י ביט יחיד בתוכנה, והן מקובצות בחמישה ports לפי ה-ABC. לכל port רגיסטר בן 8 ביטים המתייחס ל-8 הרגליים שלו (למעט PORTE, המתייחס רק ל-3 רגליים). לדוגמא, פירוט הביטים עבור PORTA:

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7 ⁽¹⁾	RA6 ⁽¹⁾	RA5	RA4	RA3	RA2	RA1	RA0

תוכלו לחפש את הרגליים RA0 וגוי בדיאגרמה המצורפת ולהוכיח כי לא כולן קרובות במיקומן על הבקר, והן אינן מהוות רצף (רגליים 2-7, וכן 13-14). כמו כן, לכל רגל הוענקו מספר שמות (ואפילו 5 שונים!). שמות אלה משמשים להתייחסות לאותה רגל בהקשרים שונים ומרמזים על פונקציות שונות

בהן היא משמשת. לדוגמא, השם המקביל ל-RA0 – AN0 מתייחס לפונקציה הנוספת של הרגל כאחת מ-9 הכניסות האנלוגיות של ה-PIC. השימוש הרגיל ביציאות/כניסות ה-PIC הוא ברמות לוגיות '0' ו-'1' בתוכנה, המתורגמות ל-0V ו-5V.

פירוט מלא של ה-ports מופיע בפרק 10 של דף הנתונים : I/O Ports.

על-מנת להתייחס לרגליים בתוכנה יש תחילה להגדירן כרגל כניסה או יציאה. (כל רגל יכולה לשמש רק כאחת מן השתיים ברגע נתון.) זאת נעשה באמצעות רגיסטר TRIS. לכל port רגיסטר כזה משלו, וכל אחד משמונת הביטים שלו מגדיר את הכיווניות של הרגל: '0' עבור Output ו-'1' עבור Input. כך, לדוגמא, אם נרצה להגדיר את RA 0-3 כיציאה ואת RA 4-7 ככניסה, נכתוב את השורה הבאה:

```
TRISA = 0b11110000;
```

הקידומת 0b מסמנת כי המספר נכתב בייצוג בינארי. מספרים ללא קידומת יקבלו ייצוג דצימלי, ומספרים עם הקידומת 0x יקבלו ייצוג הקסדצימלי.

באופן דומה, רגיסטרים ANSEL0, ANSEL1 נועדו להגדרת רגליים כאנלוגיות/דיגיטליות. ההבדל בין כניסות אנלוגיות לדיגיטליות הוא מהותי, בכך שלא ניתן להתייחס לרמות המתח בצורה לוגית כביטים בודדים, אלא יש לבצע המרה של רמת המתח לערך ספרתי. כפי שניתן לראות בטבלה למטה, הביטים של שני הרגיסטרים מתייחסים לתשע כניסות ANS (ליתר שבעת הביטים אין שימוש מעשי והגדרתם אינה משנה), 6 מתוך PORTA ו-3 מ-PORTE (ר' דיאגרמת רגליים). הייצוג ברגיסטרים אלה הוא '0' עבור Digital I/O ו-'1' עבור כניסה אנלוגית. בהמשך נתייחס ליציאות כדיגיטליות בלבד. כמו כן, יתר רגלי ה-PIC פועלות רק במצב דיגיטלי. פירוט לגבי המרת A/D בכניסות אנלוגיות ניתן למצוא בפרק 20 של דף הנתונים.

ANSEL0	ANS7 ⁽²⁾	ANS6 ⁽²⁾	ANS5 ⁽²⁾	ANS4	ANS3	ANS2	ANS1	ANS0
ANSEL1	—	—	—	—	—	—	—	ANS8 ⁽²⁾

בכדי לקרוא מרגלי כניסה, או לכתוב לרגלי יציאה נשתמש ברגיסטר PORT המוכר כבר. כאמור, כל אחד מן הביטים שלו מתייחס לאחת מן הרגליים. כמו כן, כפי שניתן לראות בטבלה בעמ' הקודם, לכל אחד מהביטים שם שמור משלו (RA0, RA1, etc.). על-מנת להתייחס לרגל ניתן לקרוא/לכתוב אל ה-port כולו, או להשתמש בשם השמור לביט הספציפי הרצוי. על-כן, שלושת הקודים הבאים שקולים:

```

PORTA = 0;
RA0 = 1;
|
PORTA = 0;
PORTA |= 0b00000001;
|
PORTA = 0;
PORTA = 0b00000001;

```

(XOR: ^ NOT: ! AND: & OR: |)

עתה נכתוב קטע קוד קצר, העושה שימוש בידע שנצבר עד עתה.

```
#include <pic18.h>

void main(void)
{
    int i = 0;

    ANSEL0 = 0;
    ANSEL1 = 0;
    TRISA = 0;
    PORTA = 0;

    while(1)
    {
        RA0 = 1 - RA0;
        for(i=0;i<500;i++);
    }
}
```

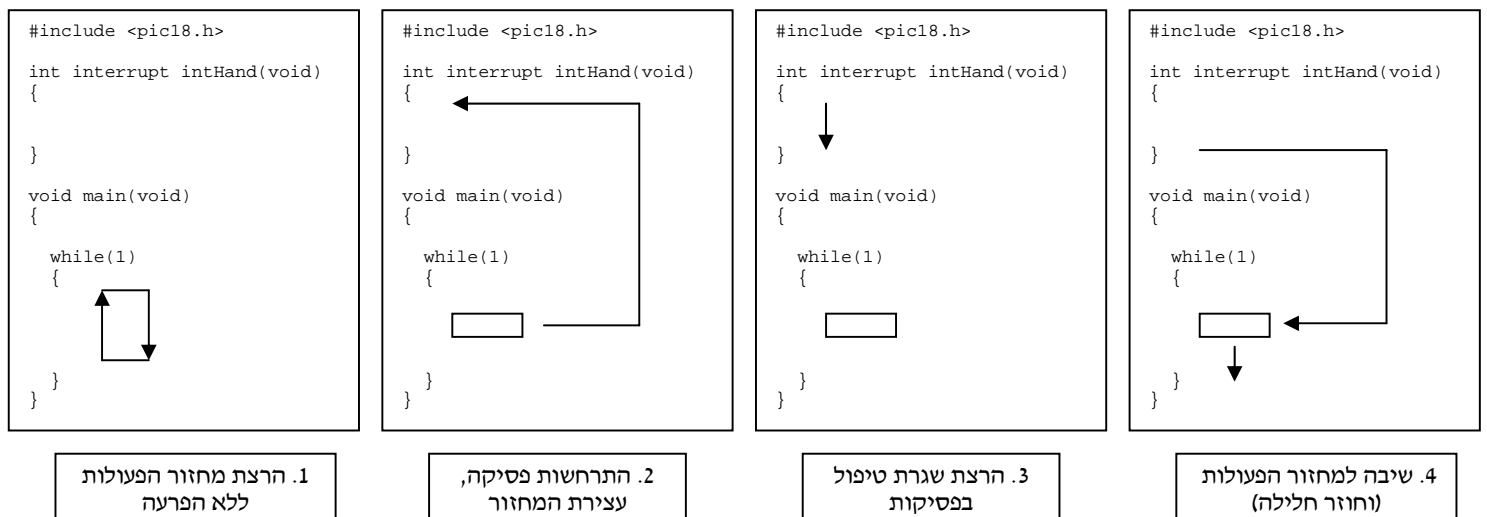
(מה עושה הקוד? מה תפקידו של ה-for?)

נשים לב כי אלמלא לולאת ה-while האינסופית הייתה התוכנית מגיעה לסיומה. ברגע שתוכנית ב-PIC מגיעה לסיומה, הבקר מבצע איתחול. במקרה שלנו לא הייתה לכך השפעה רבה על פעולת התוכנית, משום שהייתה מתקבלת "לולאה אפקטיבית" כתוצאה מהאיתחולים החוזרים, ואותן פעולות היו מתבצעות (בתדירות שונה). אולם על-פי רוב לא תהיו מעוניינים באיתחולים חוזרים לאורך הריצה, לכן לולאת ה-while היא חלק מקוד המסגרת הקבוע של תוכניות PIC.

Interrupts

כפי שראינו בדוגמא האחרונה, תוכניות ה-PIC מתבססות על מחזור פעולות קבוע, הנובע מלולאת ה-while האינסופית. לעובדה זו יש השלכות ברורות על עיצוב התוכנית שלכם, כך שעל-פי רוב היא תפעל במחזורים לצורך ביצוע משימתה המוגדרת. הפסקה של המחזור – לצורך המתנה, למשל, לקלט מסויים - עשויה לגרום שיתוק של חלקים מסויימים במערכת, תוצאה שאינה רצויה ברוב המקרים. מאידך, ככל שהתוכנית נעשית מורכבת וארוכה יותר, מחזור הפעולות עשוי לתפוח למימדים שמשפיעים על זמן התגובה של המערכת. על-מנת לשפר את יכולת התגובה של התוכנית שלכם ולעשותה לורסטילית יותר קיימות הפסיקות (Interrupts).

פסיקות מתרחשות כתוצאה מאירוע כלשהו בחומרה (במקרה זה, בקר ה-PIC). כאשר מתרחשת פסיקה, התוכנית עוברת משורת הקוד הנוכחית אל שגרה שייעודה הוא טיפול בפסיקות (שלב 2 בסכמה למטה). בתום הרצת שגרה זו, התוכנית תחזור למחזור הפעולות, אל המקום שבו הופסקה. בבקר ה-PIC פסיקות רבות המתייחסות הן לשינויים בקלט, בפלט או בפעילות היחידות הפנימיות של הבקר (למשל, timer). בשגרה המטפלת בפסיקות תוכלו לאתר את הפסיקה המסויימת שארעה, לבצע מספר פעולות הנדרשות כתוצאה מכך (שלב 3 בסכמה), ולהמשיך במחזור הפעולות.

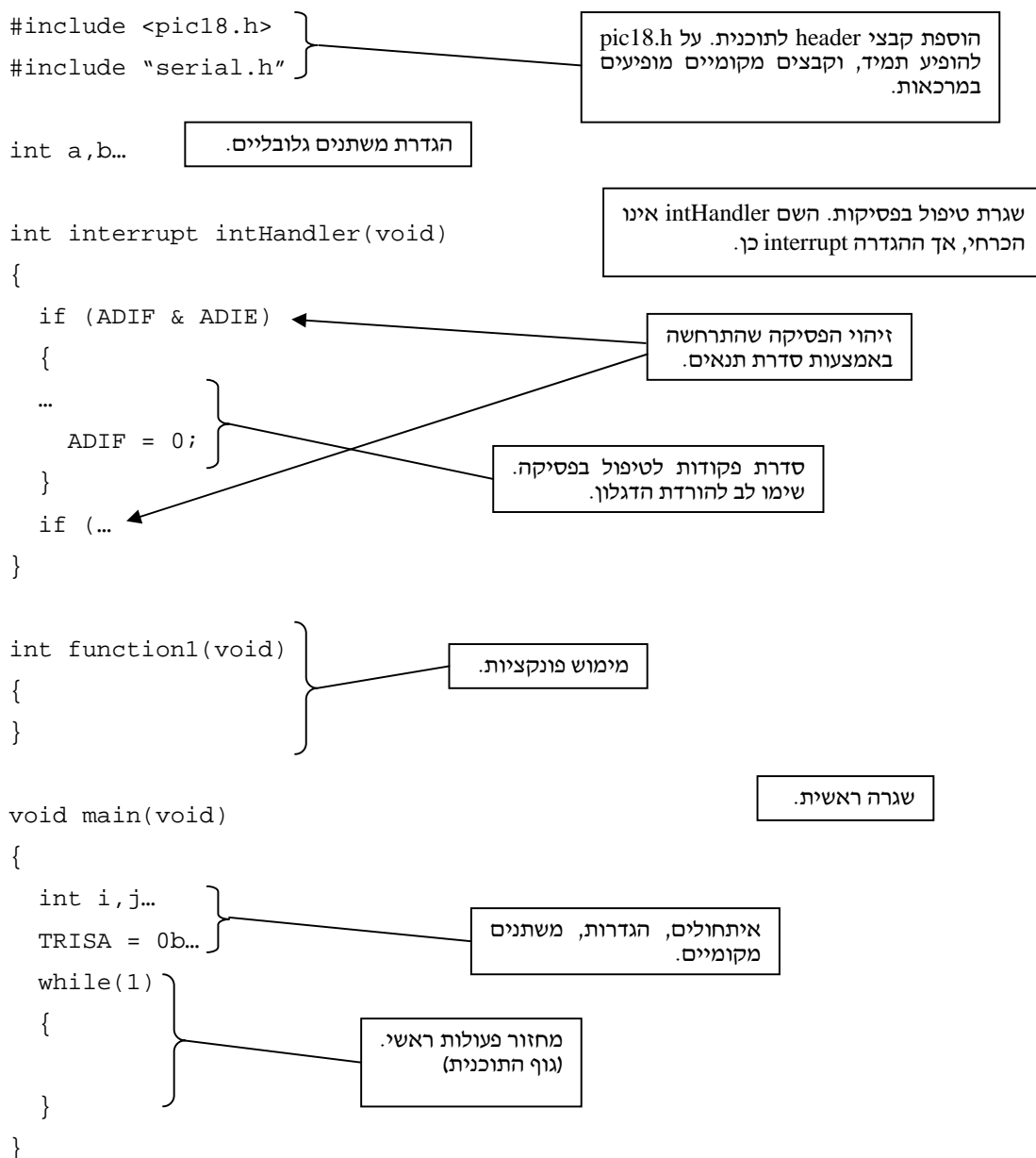


זיהוי הפסיקה נעשה באמצעות דגלון (flag) המורם אוטומטית ע"י הבקר ל-1' לוגי כאשר מתרחשת פסיקה, במקביל לעצירת המחזור. לכל פסיקה דגלון משלה, וכך ניתן באמצעות תנאי פשוט לזהות איזו פסיקה התרחשה. חשוב לזכור, כי הבקר אחראי רק על הרמת הדגלון ואילו על התוכנה שלכם מוטל להוריד את הדגלון חזרה ל-0' לוגי לאחר זיהוי הפסיקה הנכונה. התרשלות בהורדת הדגלון

תוביל לזיהוי חוזר של פסיקות שלא היו ולא נבראו, בכל פעם שהתוכנית תכנס לשגרת הטיפול בפסיקות. הסיומת האופיינית לביט של דגלון-פסיקה היא IF (Interrupt Flag).

קיים ביט נוסף לכל פסיקה המשמש כמפסק, כלומר, מאפשר לכם להתעלם מאותה פסיקה גם כאשר היא מתרחשת ולהמשיך במחזור הפעולות ללא הפרעה. הסיומת האופיינית לביט כזה היא IE (Interrupt Enable), וכאשר הוא מורם פסיקה פעילה.

לסיכום, המבנה האופייני של תוכנית PIC הוא כדלהלן:

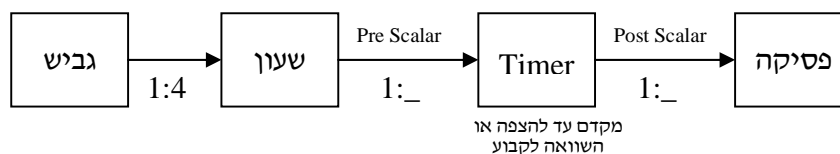


Timer0

לסיום, נשדרג את הדוגמא הקודמת באמצעות שימוש בפסיקות. בפעם הקודמת השתמשנו בלולאת for על-מנת ליצור השהיה בהבהוב הרגל. כאמור, השהיה סדרתית שכזו מעכבת את המחזור ולא מאפשרת ביצוע פעולות במקביל. אילו יכולנו להשתמש בפסיקה על-מנת לממש השהיה היינו מרוויחים הרבה מחזורי שעון מבוזבזים. לשם כך אידיאלי להשתמש באחד מן הטיימרים (Timers) של ה-PIC.

כאשר מופעל טיימר ב-PIC הוא מייצר פסיקה מדי פרק זמן, הניתן להגדרה ע"י המשתמש. בחלק מן הטיימרים הפסיקה מתרחשת כאשר יש הצפה (overflow) של הטיימר, ובחלקם ניתן לקבוע את אורך המחזור. מדי מחזור שעון מקודם רגיסטר בטיימר, עד לקיום תנאי האתחול שלו, והפעלת הפסיקה.

מחזור שעון אורכו ארבעה מחזורי גביש. אם, לדוגמא, הגביש שלנו פועל בתדירות 40MHz, תדירות מחזור השעון תהיה 10Mhz. כמו כן, ניתן לבצע הפחתה נוספת של קצב הפסיקות באמצעות שני קבועי-הפחתה: pre-scalar ו-post-scalar. במקרה הראשון, ההפחתה נעשית לפני קידום הטיימר. לדוגמא, במקרה של יחס pre-scalar של 1:8, רק אחד מכל 8 מחזורי שעון יביא לקידום הטיימר. קיבלנו למעשה הפחתה אפקטיבית פי שמונה של תדר השעון. במקרה השני, ההפחתה נעשית לאחר קידום הטיימר. לדוגמא, במקרה של יחס post-scalar של 1:8, רק אחד מכל 8 איתחולים של הטיימר יביא ליצירת פסיקה. קיבלנו למעשה הפחתה אפקטיבית פי שמונה של תדירות הפסיקות מן הטיימר. ההפחתה הכוללת, כמובן, היא מכפלת קבועי-ההפחתה. כך ניתן להגיע לדיוק רב בבחירת ההשהיה של הטיימר.



בדוגמא שלנו נעשה שימוש ב-Timer0: טיימר של 8-ביט או 16-ביט לפי בחירה המקבל פסיקה לפי הצפה, ובעל pre-scalar של 8-ביט. כמו כן, ניתן לבחור לטיימר זה שעון חיצוני בנפרד משעון המערכת. הגדרות הטיימר מסתכמות ברגיסטר T0CON. פרטים נוספים תוכלו למצוא בעמ' 135 בדף הנתונים.

על-מנת להפעיל את הפסיקה מטיימר זה נרים את ביט TMR0IE, וכן את GIE ו-PEIE שהינם מפסקים כלליים לכל הפסיקות. שני ביטים אלה חייבים להיות מורמים בכדי לקבל פסיקות כלשהן, וכאשר מורידים אותם ניתן לשתק כך את כל הפסיקות.

תוכנית העושה שימוש ב-Timer0 לצורך השהיה עשויה להיראות כך :
(השוו עם הדוגמא בפרק הקודם)

```
#include <pic18.h>

int interrupt intHandler(void)
{
    if (TMR0IF & TMR0IE)
    {
        RA0 = 1 - RA0;
        TMR0IF = 0;
    }
}

void main(void)
{
    ANSEL0 = 0;
    ANSEL1 = 0;
    TRISA = 0;
    PORTA = 0;

    TOCON = 0b00000111;    // See p.135 for details about TOCON.

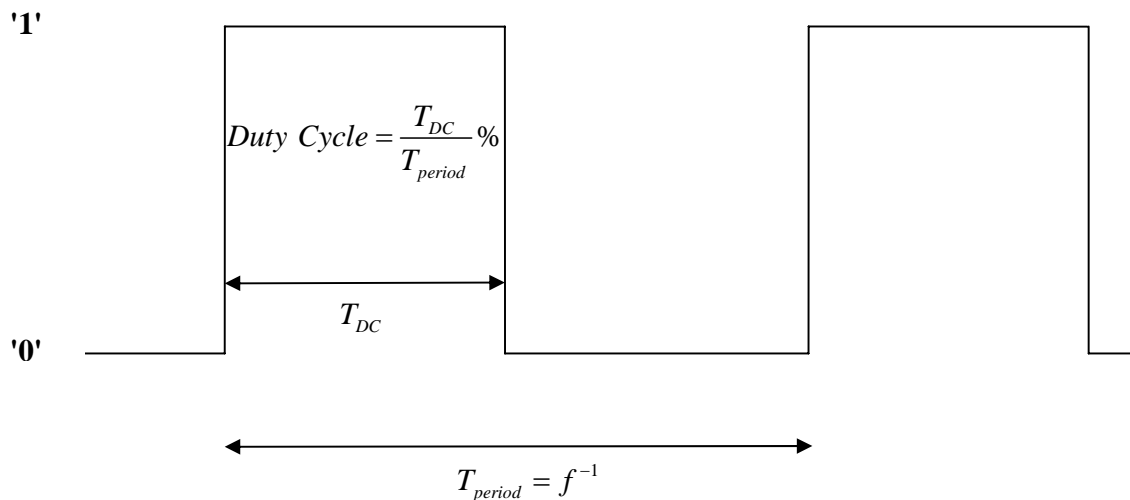
    TMR0IE = 1;
    TMR0IF = 0;
    GIE = 1;
    PEIE = 1;

    TMR0ON = 1;           // Switching Timer0 on (bit 7 on TOCON) after
initialization is over.

    while(1)
    {
    }
}
```

PWM

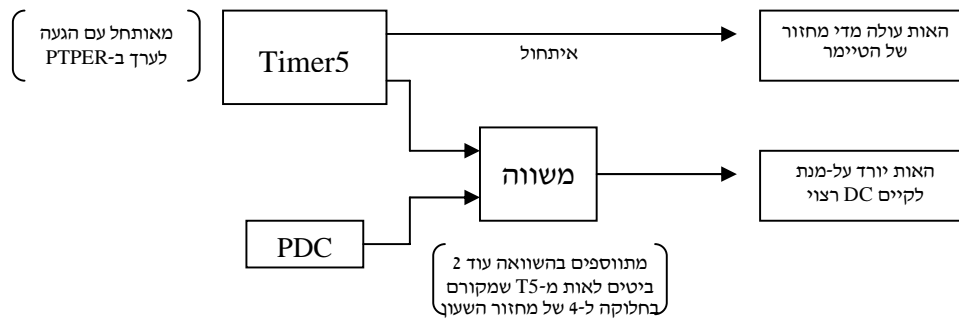
"PWM stands for **P**ulse **W**idth **M**odulation", כלומר, גל ריבועי בעל מחזור קבוע, אשר רוחב הקטע בו האות גבוה ('1' לוגי) משתנה (למעשה, שינוי ה-Duty Cycle של האות). משהו שנראה בערך כך:



אותות בעלי רוחב משתנה אלה משמשים בדרך-כלל לשליטה במנועי סרוו. ה-D.C. (Duty Cycle) של האות יקבע את הזווית אליה יגיע המנוע במקרה של מנוע עם בקרה על הזווית, או שיקבע את מהירות הסיבוב שלו במקרה של מנוע שקיבוע הזווית שלו הוסר.

ב-PIC18F4431 יחידה המיועדת במיוחד להוצאת אות PWM. היא מסוגלת להוציא עד 4 אותות בעלי תדר זהה, ו-D.C. שונה. יחידת ה-PWM משתמשת ב-Timer5 על-מנת לייצר את הגל הריבועי. על-כן לצורך פעולתה יש להגדיר רגיסטרים הקשורים בטיימר (PTCON0, PTCON1). כמו כן נגדיר את היציאות המבוקשות (TRISD, TRISB, PWMCON0). בנוסף, ניתן לתזמן פעולות אחרות ב-PIC, כמו המרת A/D בהתבסס על הטיימר של ה-PWM (מתוזמן עם המחזור שלו), ברגיסטר PWMCON1 (כברירת-מחדל נותר על האפשרות הזו). פרטים נוספים ניתן למצוא בפרק 17: Power Control PWM Module. תיאור מפורט של רגיסטרי ההגדרה של היחידה מופיע החל מעמ' 188.

את זמן המחזור של האות מגדירים באמצעות קבוע PTPER (זוג רגיסטרים PTPERL, PTPERH – בסה"כ 12-ביט), ואילו ה-Duty Cycle עבור כל אחת מהיציאות נקבע ע"י קבוע PDC (ארבעה זוגות PDC0L, PDC0H – בסה"כ 14-ביט). סכמת המערכת מופיעה בעמ' הבא.



ציאות ה-PWM מגיעות בזוגות (בסה"כ 8 רגליים). חפשו את מיקום הרגלים PWM0-7 בתרשים הרגליים של ה-PIC. ישנם שני מצבי עבודה עיקריים: מצב עבודה עצמאי, בו אין קורלציה בין שתי הרגליים. במצב זה ניתן להשתמש ברגל האי-זוגית לצורך הוצאת אות PWM, וברגל הזוגית להשתמש לכל מטרה אחרת. במצב עבודה השני, הרגלים הן משלימות אחת של השנייה ויש בניהן קורלציה (מתאים למערכות הנעה מסויימות). אנו נעבוד במצב הראשון. מצב העבודה לכל אחד מן הזוגות נקבע בביטים 0-3 ברגיסטר PWM0CON.

להלן קוד לדוגמא, המדגים עד כמה פשוט השימוש ביחידת ה-PWM. עם תום ההגדרות הראשוניות, כל שינוי בזמן המחזור או ב-D.C של האות מסתכם בשינוי הקבוע המתאים. כל היתר נעשה באופן אוטומטי ע"י הבקר.

```
#include <pic18.h>

#define getmsb(X) (((X) >> 8) & 0b11111111) // These two macros get
// the 8 most significant and the 8 least significant bits of a value X.
#define getlsb(X) ((X) & 0b11111111)

void main()
{
    TRISB = 0b00000000; // All PortB pins are set for output
    PTCON0 = 0b00011100; // 1:2 PostScalar; 1:64 Prescalar; Free Running
    PWMCON0 = 0b00101111; // PWM0 & PWM1 pins enabled (output on RB1)
    PWMCON1 = 0b00000000;
    PTPERH = getmsb(1562); // 50Hz (designed for a 40MHz oscillator). See
// formula on p. 195 in PIC 18F4431 datasheet.
    PTPERL = getlsb(1562);
    PDC0L = getlsb(312); // i.e. (PDC0+1)/(4*(PTPER+1))=5% Duty Cycle
    PDC0H = getmsb(312);

    GIE = 1; // Global interrupt enabled
    PEIE = 1; // Peripheral interrupt enabled
    PTIP = 1; // PWM TimeBase interrupt - high priority
    PTIE = 0; // PWM TimeBase interrupt disabled
    PTIF = 0; // PEM Timebase interrupt flag cleared
    PTEN = 1; // PWM enabled

    while (1)
    {
    }
}
```

סיכום

כפי שראינו בפרקים הקודמים, ישנם מספר הבדלים מהותיים בין מבנה תוכנית מחשב לבין תוכנית PIC. הצגנו את השימוש בפסיקות, ואת הגדרת יחידות ה-PIC באמצעות רגיסטרי-בקה. לבקר עוד מספר רב של יחידות לצורך ביצוע פעולות המותאמות לצרכים ספציפיים שונים, ובסה"כ דרכי הגדרתן והפעלתן דומות לדוגמאות שהצגנו כאן. כל היחידות מתוארות בפירוט רב בדף הנתונים, אשר אל נפלאותיו התוודענו שוב ושוב.

מאחל לכם הצלחה בפרויקט והנאה מהשימוש ב-PIC!

אלון

טיפים

נקנח במספר עצות שימושיות:

- למה זה לא עובד?
 - יתכן כי הקובץ לא נקלט ב-bootloader: בידקו כי הסחת כתובות הזיכרון ב-300 מילה אכן מוגדרת כראוי ב-Project ← Build Options ← Project ← PICC-18 Linker. כמו כן, יתכנו בעיות בתקשורת בטרמינל. שימו לב כי אכן התקבלו סוגריים סוגרים (" בסוף העברת קובץ ה-hex.
 - חיבורים: להשתדל לעבוד עם חיבורים קבועים, ולא עם Wire-up. יתכנו רעשים על הקווים, וכדאי לבדוק זאת עם סקופ ולהוסיף סינון רעשים בעת הצורך. אולי אתם פשוט לא מחוברים לרגל הנכונה. (רגליים מתחילים לספור מ-0!)
- בסוף כל פרק בדף הנתונים ישנה טבלה המרכזת את כל הרגיסטרים הקשורים לפעילות של היחידה המוצגת בפרק. רגיסטרים או ביטים הרלוונטיים להפעלת היחידה צבועים בלבן, ואילו היתר צבועים באפור.
- ניתן לנהל את הפרויקט במקביל ב-Visual Studio מטעמי נוחות, ולהשתמש בסביבת MPLAB לצורך קימפול ודיבוג בלבד.
- על-מנת להשתמש בספריות קוד שנכתבו מראש, יש להוסיף את קבצי ה-C ואת קבצי ה-header התואמים להם. יש למקם את הקבצים בתוך ספריית הפרויקט, ולהוסיף את הקוד:
`#include "file.h"`

יש לשים לב לשימוש במרכאות עבור קבצים שנמצאים בספריית הפרויקט, לעומת קבצים הנמצאים בספריית הקומפילר כמו pic18.h. כמו כן יש להוסיף את הקבצים לרשימת קבצי הפרויקט ב-MPLAB.

- לא מומלץ לסמוך על ה-Debugger של MPLAB. עדיף במידת האפשר לשרד משוב משלכם הממומש בתוכנה, דרך תקשורת סריאלית למחשב. (אפשר לצפות דרך Terminal, או Serial Watcher המצורפים). ר' קבצי sci.h, sci.c המצורפים המממשים תקשורת סריאלית ב-PIC.
- רצוי לשמור על קוד מינימלי בשגרת הטיפול בפסיקות, וזאת בכדי שתצאו ממנה כמה שיותר מהר ותפנו אותה לטפל בפסיקות חדשות המגיעות מהבקר. ניתן לעשות שימוש בדגלונים מתוצרת עצמית בכדי להעביר מידע על התרחשות פסיקה לתוך מחזור הפעולות הראשי, ושם להפעיל את קטעי הקוד הרצויים.
- מומלץ מאוד להמנע לחלוטין משימוש במשתנים מסוג float מאחר והשימוש בהם גורר הגדלה משמעותית בסיבוכיות מקום וחישוב. אפילו הופעה בודדת של float בתוכנית תביא לתוספת קוד לצורך טיפול בנקודה צפה. כתחליף ניתן להשתמש במשתנים שלמים גדולים יותר, ולהחליט להכפיל את כל הגדלים בקבוע לפי מספר הספרות לאחר הנקודה שדרוש לכם.