תוכן עניינים

LAB1 preface - C language for MCU

A. הקדמה:

- לאחר קורס "מבוא למחשבים" משולב מעבדה (מעבדת מיקרו-מחשבים) בה למדנו את עבודת המעבד מול הזיכרון והרכיבים הפריפריאליים הבסיסיים (GPIO, Interrupts, TIMERS, ADC, DAC) החל משכבת ה-ISA דרך כתיבת קוד אסמבלי ועד לשכבת האפליקציה, דרך שכבת קוד HAL (Hardware Abstraction Layer). בשימוש קוד אסמבלי היה באפשרותנו לרדת עד לרמת הרגיסטרים במעבד ולרזולוציית זמנים של מחזורי שעון המעבד MCLK.
- במעבדה הצמודה לקורס DCS אנו נעסוק בעבודת המעבד מול הזיכרון והרכיבים הפריפריאליים הבסיסיים שעסקנו בקורס המבוא (GPIO, Interrupts, TIMERS, ADC, DAC) ונעלה דרגה ונעסוק בעבודת במעבד מול רכיבים פריפריאליים מתקדמים (DMA, Communication modules, Flash Memory controller) .

כתיבת הקוד תהיה בשפת C. בגישה של בכתיבת קוד בשפה עילית תהיה מעל הרזולוציה של ליבת המעבד ע"י עבודה ישירות מול הרכיבים הפריפריאליים.

- בהמשך למטלת הבית בשפת C אשר הייתה מיועדת לכתיבת אפליקציה ולא לכתיבת מערכת נעזרנו בפונקציות ספרייה stdio לצורך ממשק קלט ופלט אשר נתמכות ע"י מערכת ההפעלה של המחשב האישי. שימוש בשפת C לצורך תכנות מערכת משובצת מחשב לא מאפשרת עבודה עם פונקציות ספרייה stdio לצורך ממשק קלט ופלט (ללא תמיכה מתאימה בקוד המערכת אותה נבצע בהמשך).
- במסך זה נלמד לפתוח פרויקט בסביבות עבודה **CCS**, **IAR** איתם נעבוד בקורס וחשובים להבנת עיקרון העבודה עם סביבות פיתוח שונות וההתמצאות בהם. מטרת מסמך זה היא לעשות את המעבר מקוד פשוט בשפת C שנועד לריצה על גבי PC מקוד פשוט בשפת C שנועד לריצה על גבי MCU.
- **בחלק זה המהווה הקדמה לדו"ח מכין של ניסוי מעבדה 1, תצטרכו להריץ קוד נתון ולענות שאלות תיאורטיות בלבד.**

B. שלבים פתיחת פרויקט חדש בסביבת פיתוח IAR או CCE:

1. הקדמה:

תחילה נתמקד בכתיבת קוד בשפת C והרצה וביצוע debug בסביבת הפיתוח הנקראת **IAR** IDE ולאחר מכן בסביבת הפיתוח **CCS** IDE מבוססת Eclipse בה נעבוד באופן בלעדי מניסוי מעבדה 4 ואילך.

סביבת פיתוח נקראת **IDE** = **I**ntegrated **D**evelopment **E**nvironment .

באופן כללי סביבת הפיתוח משמשת ליצירת קוד מכונה (קוד בינארי) מתוך טקסט (קוד) הנכתב ב-Editor של סביבת הפיתוח. לאחר מכן נצרוב את קוד מכונה לזיכרון FLASH של הבקר דרך סביבת הפיתוח לצורך ביצוע התוכנית על גבי הבקר.

2. שלושה שלבים בפיתוח קוד בסביבת IDE:

- **סימולציה** – סביבת הפיתוח משמשת סימולטור לבקר שלנו. את הקוד שכתבנו נפעיל ב**מצב סימולטור** והוא ירוץ בפועל ב-PC (מחשב אישי) בלבד, לצורך דימוי הבקר.

הערה: מצב סימולציה קיים ב **IAR** IDE בלבד והוא מוגבל לפעולת הליבה בלבד (ולא עם עבודה של המעבד מול רכיבים פריפריאליים)

- **Debug** – הקוד שכתבנו ייצרב לבקר (מה-PC) ובהפעלתו הוא ירוץ בבקר ולא ב-PC, אולם ישנה תקשורת בין הבקר ל-PC לצורך תמיכה ב-DEBUG (נקודות עצירה, ריצה בצעדים, בדיקת ערכי רגיסטרים, ערכים בזיכרון וכו').
- **Active Application** – הקוד שכתבנו ייצרב לבקר (מה-PC) ובהפעלתו הוא ירוץ בבקר בלבד ללא קשר עם ה-PC (בשונה ממצב DEBU). מצב זה מונה גם Stand Alone, מאחר ובמצב זה הבקר בפני עצמו ללא קשר ל-PC.

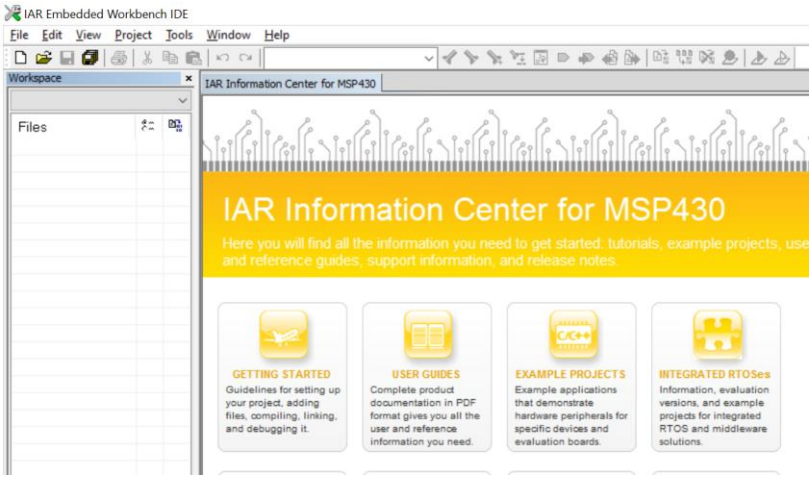
3. התקנה ופתיחת פרוייקט חדש בסביבת פיתוח IAR IDE:

- תוכנת **IAR** IDE סביבת הפיתוח המותקנת במעבדה, ניתן להורידה ולהתקינה במחשבכם האישי.

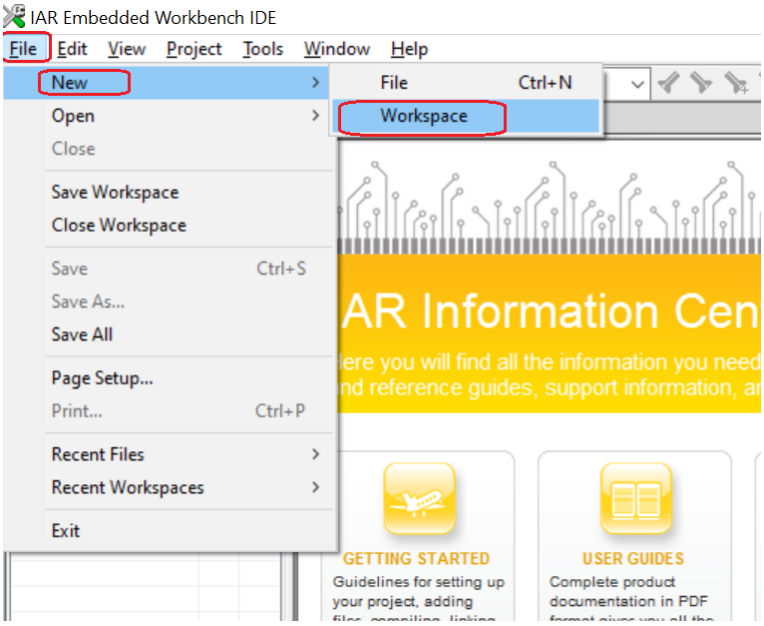
ניתן להורדה מהקישור הבא:

IAR - IDE setup.zip

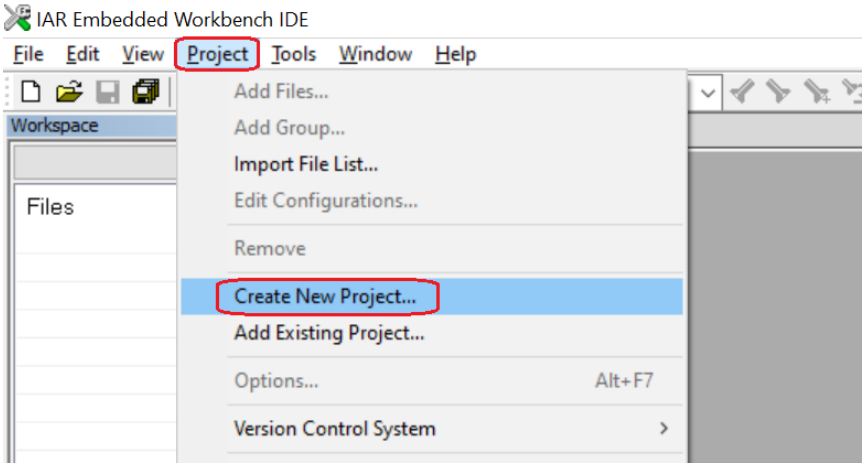
- נכין תיקייה במחשב הייעודית לפרויקט, נבחר לדוגמה שם TEST לתיקייה.
- נפתח את תוכנת IAR בלחיצה על קיצור הדרך, צריך להיפתח חלון הבא.



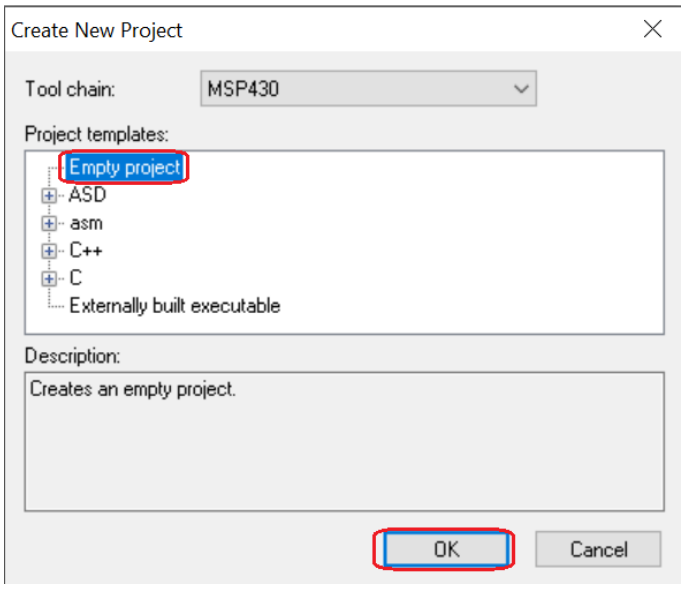
- נפתח workspace חדש: **File → New → Project**



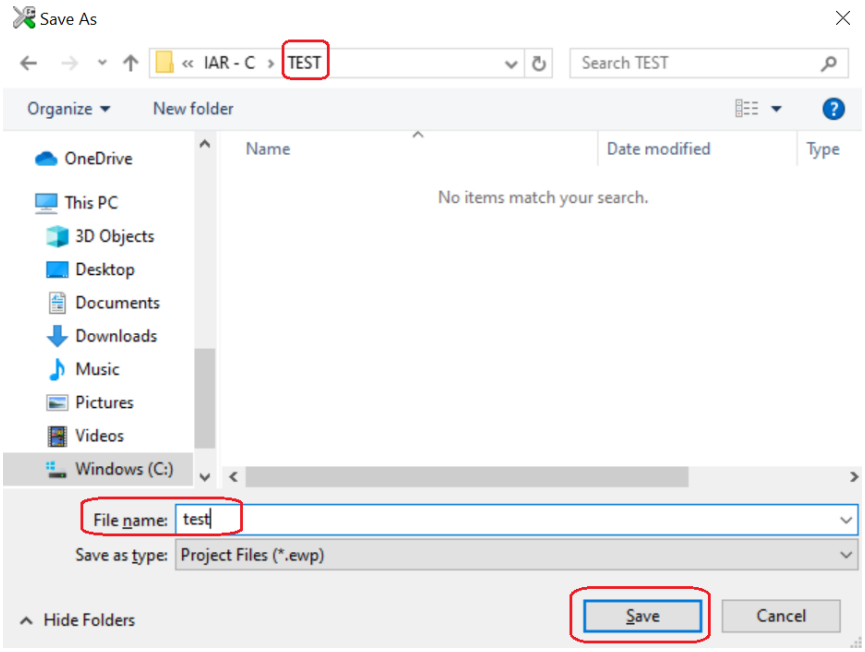
- <u>נפתח פרויקט חדש</u>: **Project → Create New Project**



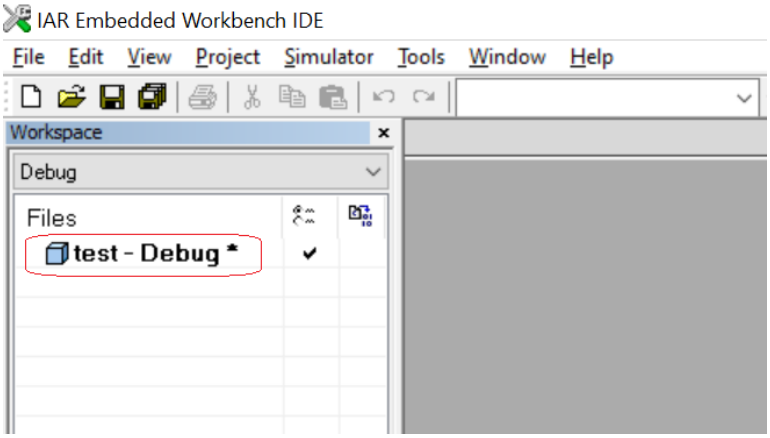
בחלון שניפתח נבחר פרויקט ריק.



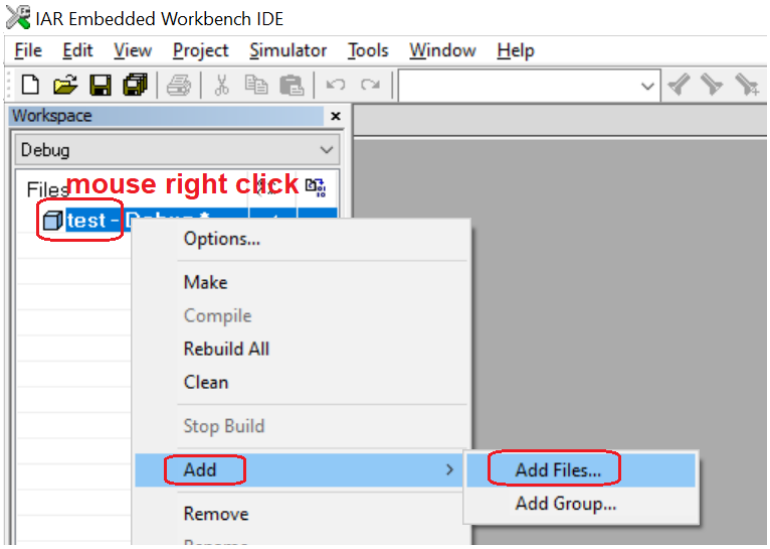
נמקם את הפרויקט בתיקייה שהכנו מראש, בשם TEST ונקרא לפרויקט בשם test.



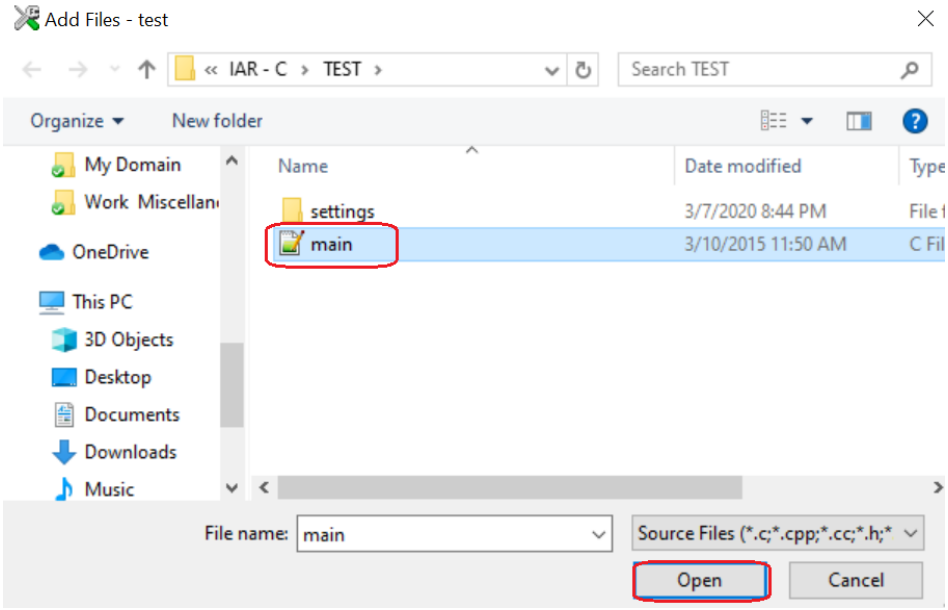
נשים לב שבחלון ה- workspace נפתח פרויקט חדש בשם test:



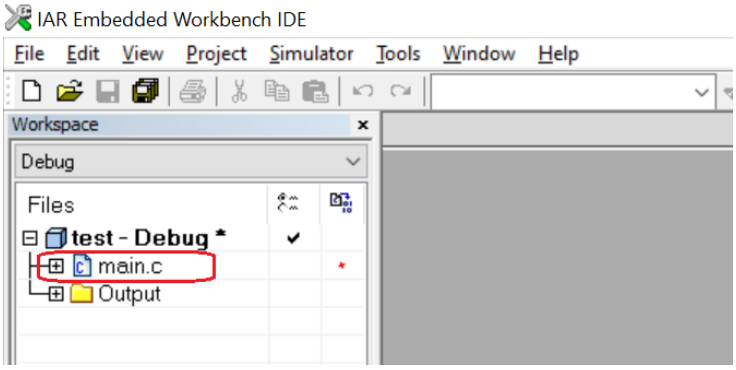
- **נצרף את קובצי המקור של הפרויקט לסביבת הפיתוח:**



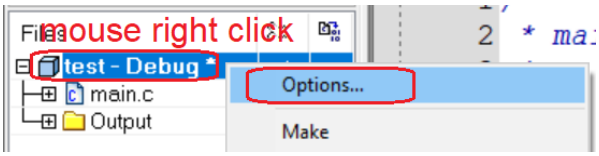
נבחר את הקובץ / קבצים שנרצה לצרף לפרויקט:



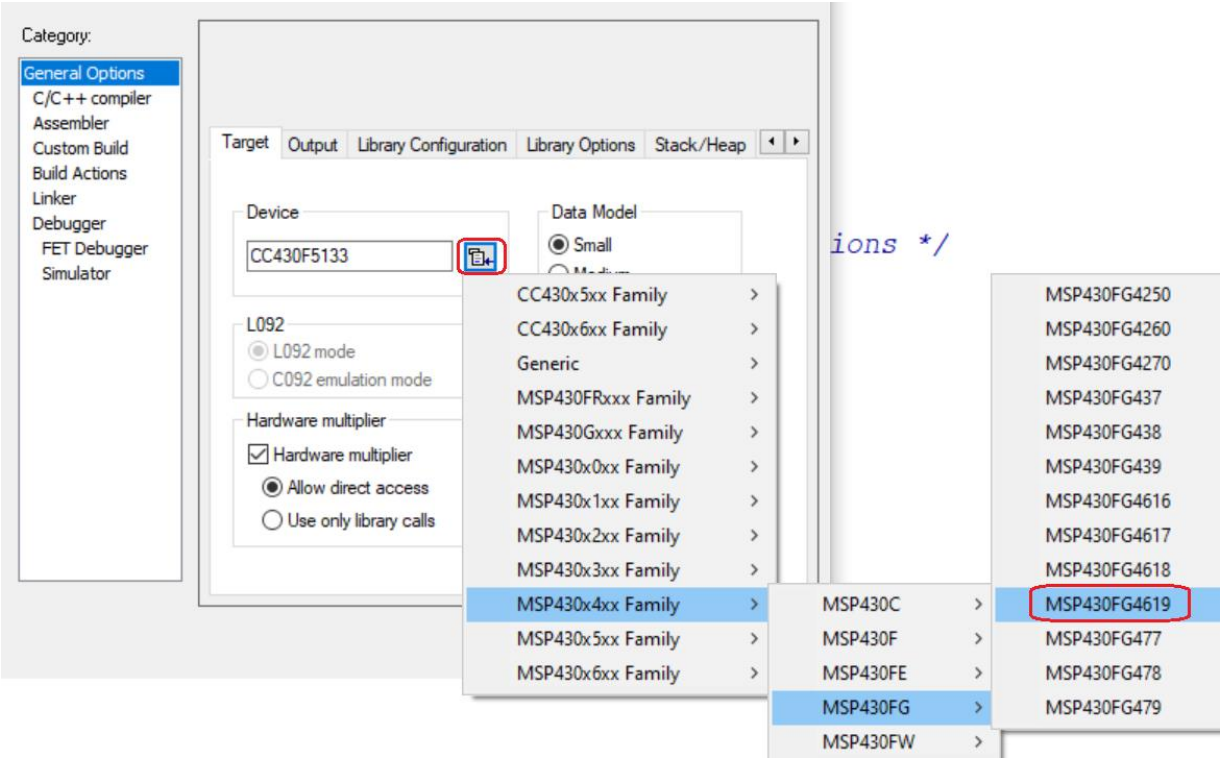
נשים לב שהקובץ / קבצים מצורפים לפרויקט (לפתיחת הקובץ לחץ עליו double click).



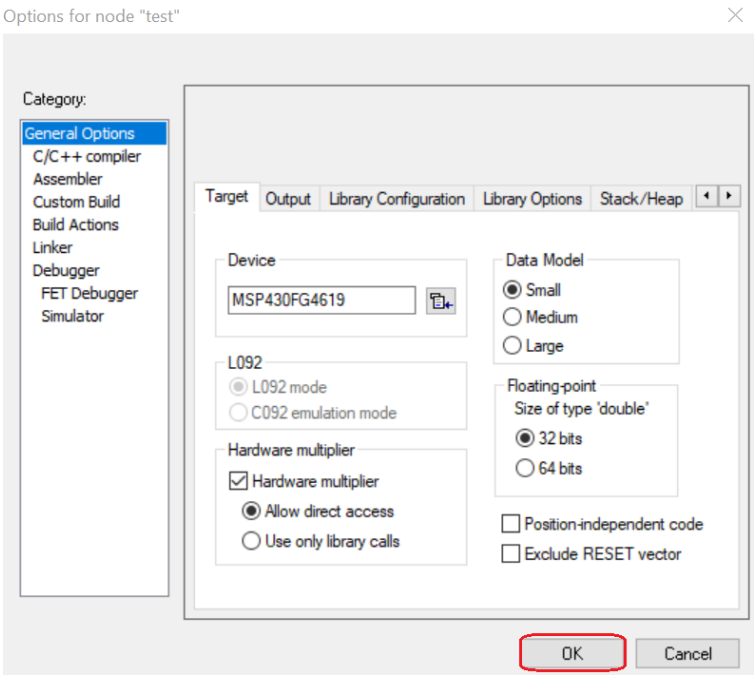
- בחירת הבקר אליו מיועד להיצרב הקוד:



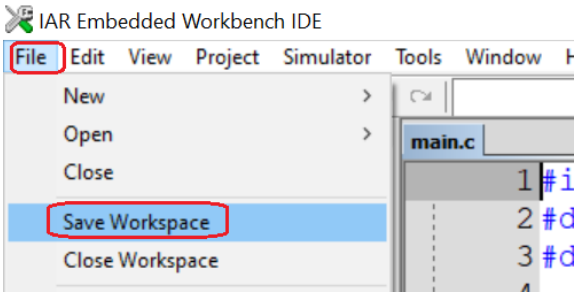
נפתח החלון הבא:



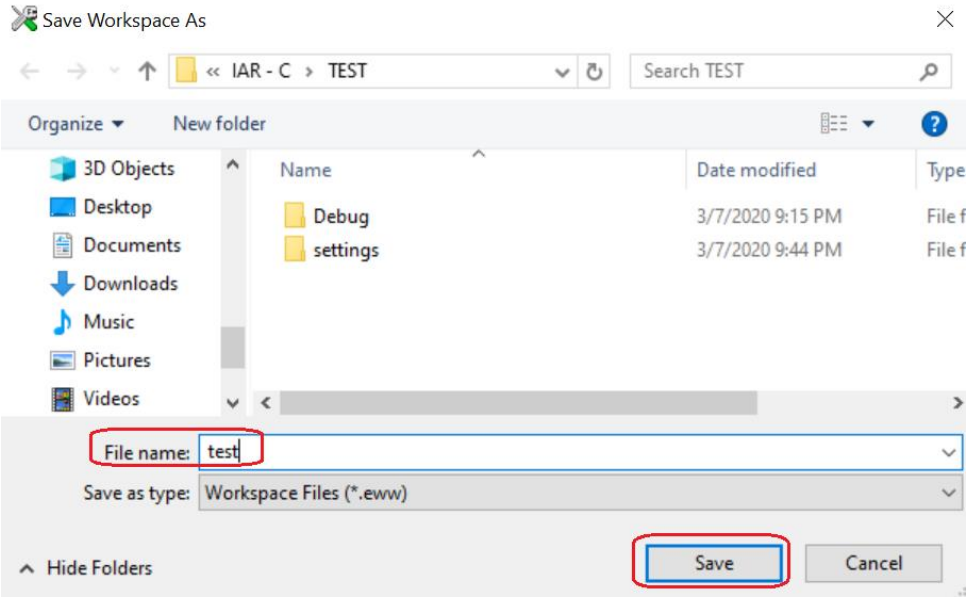
לבסוף נלחץ OK



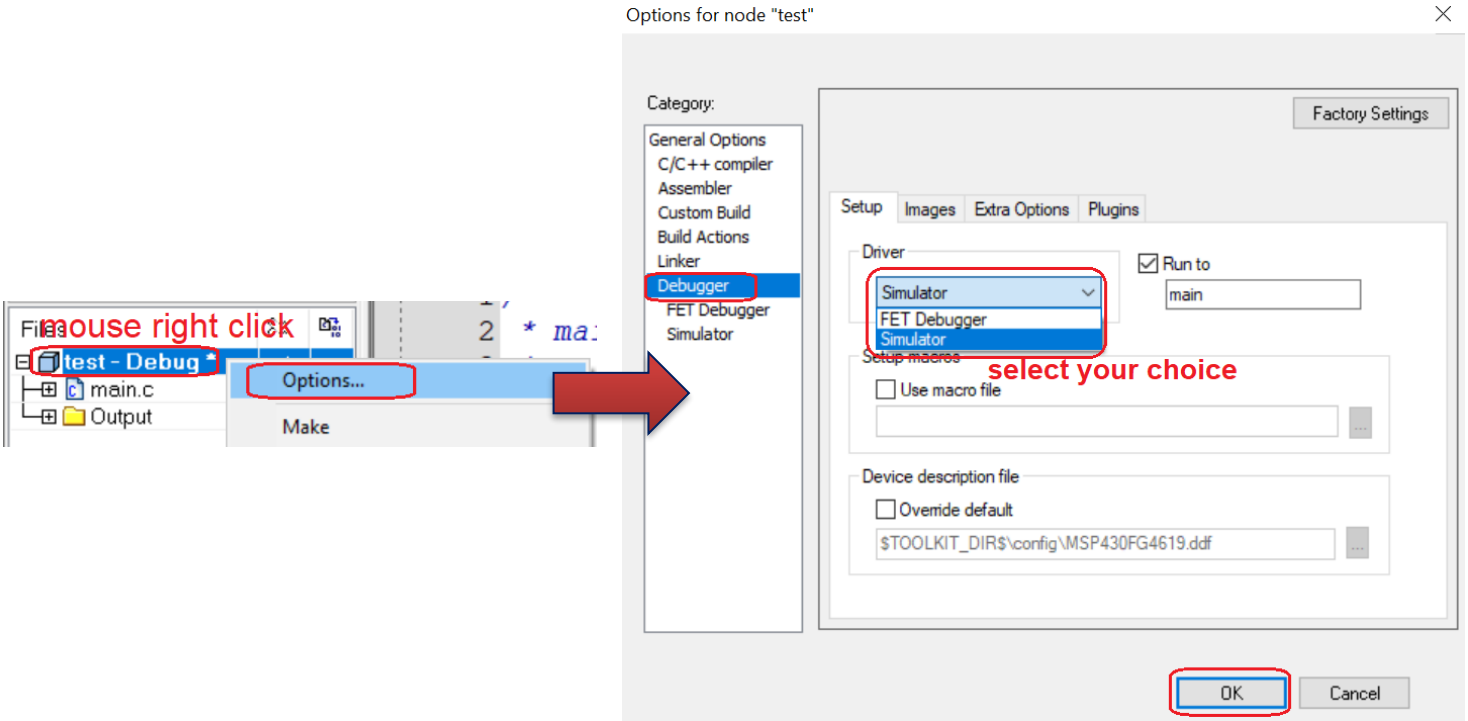
- **לבסוף נשמור את ה- workspace שהגדרנו:**



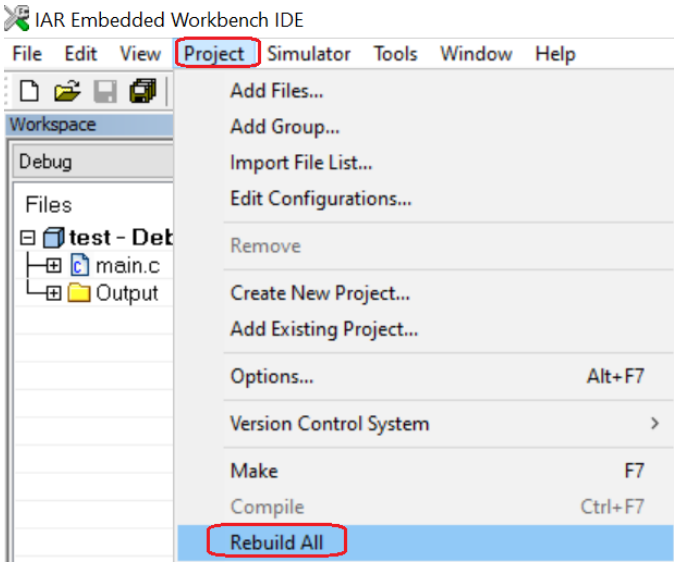
ניפתח חלון, בתוכו ניתן שם ל- workspace.



- הגדרת מצב עבודה debugger / simulator :



- הידור ובניית הפרויקט (Building Project) :



4. התקנה ופתיחת פרויקט חדש בסביבת פיתוח CCS IDE (מבוססת Eclipse):

- תוכנת **CCS** IDE על גבי מערכת הפעלה *Win10*, ניתן להורידה ולהתקינה במחשבכם האישי.

  ניתן להורדה מהקישור הבא:

  CCS - IDE setup.zip

- תוכנת **CCS** IDE על גבי מערכות הפעלה *Linux, MacOS*, ניתן להורידה ולהתקינה במחשבכם האישי.

  ניתן להורדה מהקישור הבא:

  CCS version7 download

- הסבר לשלבי פתיחת פרויקט:

  How to create a new project in code composer studio (see time 0-9:30)

5. הרצת קוד דוגמה בשפת C – מענה על שאלות:

בתרגיל הכנה זה נשתמש בקוד לדוגמה הנמצא בקובץ **main.c** (נמצא במודל בתיקיית קוד לדוגמה).

נצרף את הקובץ לפרויקט שהכנו מראש לצורך הרצה במצב simulator. בסעיף זה נרצה להריץ את קוד הדוגמה במטרה לתרגל כתיבת קוד בשפת C.

התוכנית לדוגמה, מגדירה בזיכרון ה- RAM מערך דו-מימדי בגודל NxN ומאתחלת את אברי המטריצה בערכים מ-0 עד 99 לפי הנוסחה הבאה, $\mathbf{Mat1[i][j] = i \cdot N + j}$. נגדיר משתנה בשם **Selector**, התוכנית בודקת את ערכו **בלולאה אינסופית** (בשונה מ- PC העובד תחת מערכת הפעלה, תכנות בקר חייב להתבצע במעטפת של לולאה אינסופית, או שימוש בפקודת הכנסה של הבקר למצב שינה) ובהתאם לערכו מבצעת על המטריצה **Mat1** פעולה מתאימה מהתפריט הבא.

כתיבת תוצאת הפעולה תיכתב למטריצה **Mat2** באותם הממדים של **Mat1**.

- כאשר Selector = 0 :

  לא נעשה כלום.

- כאשר Selector = 1 :

  חישוב עקבה של המטריצה וכתיבתה לתוך משתנה בשם Trace.

- כאשר Selector = 2 :

  ביצוע transpose למטריצה (כתיבה למטריצה **Mat2**).

- כאשר Selector = 3 :

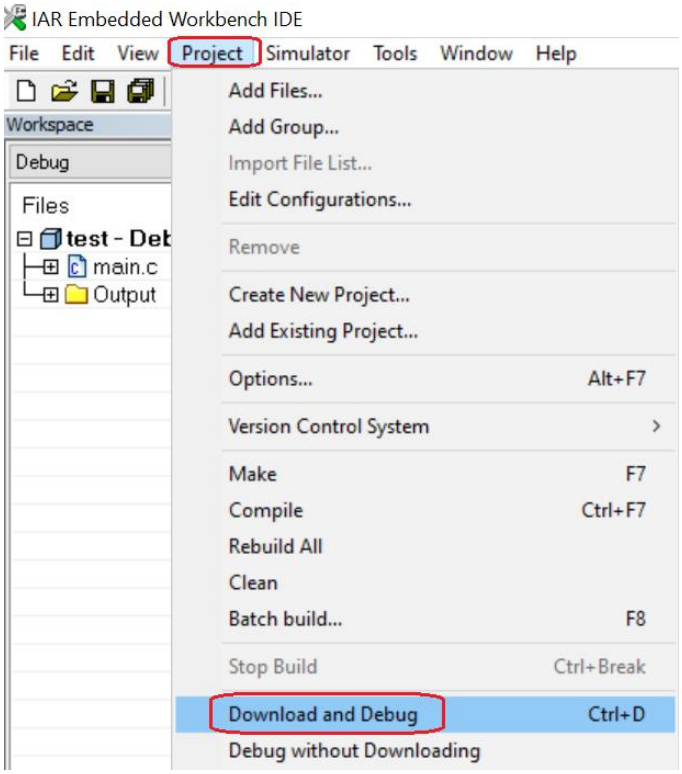
  החלפת אלכסוני המטריצה, בין אלכסון ראשי למשני (כתיבה למטריצה **Mat2**).

- כאשר Selector = 4 :

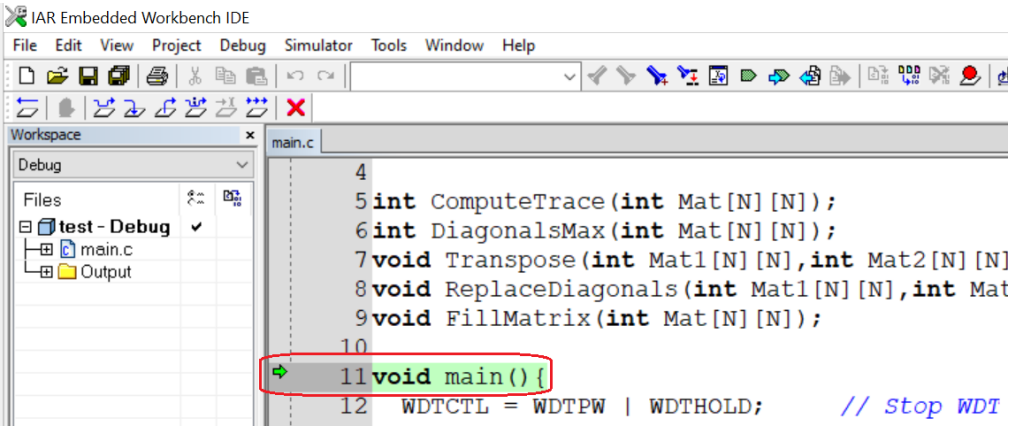
  חישוב ערך מקסימאלי בין 2 אלכסוני המטריצה וכתיבתו לתוך משתנה בשם Max.

6. ביצוע DEBUG - תוכנית לדוגמא:

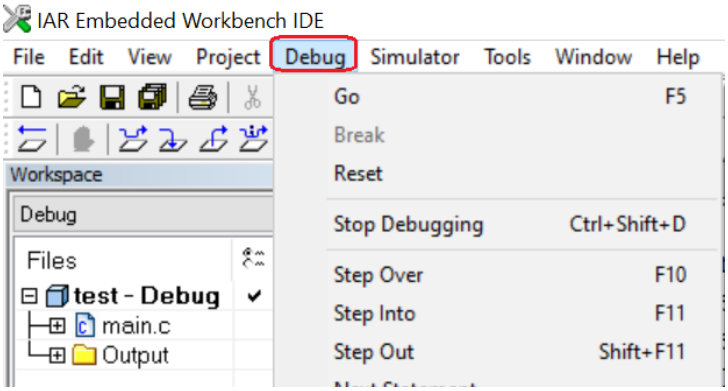
- נבצע debug לתוכנית לדוגמה:



לאחר כניסה למצב **DEBUG**, חץ ההרצה הירוק יעמוד על שורת main.

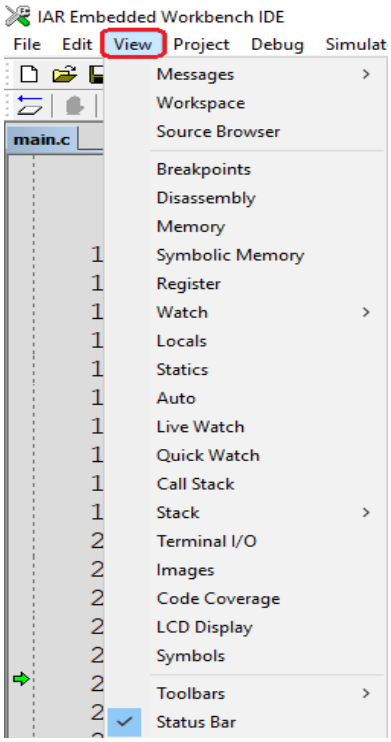


סוג הריצה נבחר בצורה הבאה

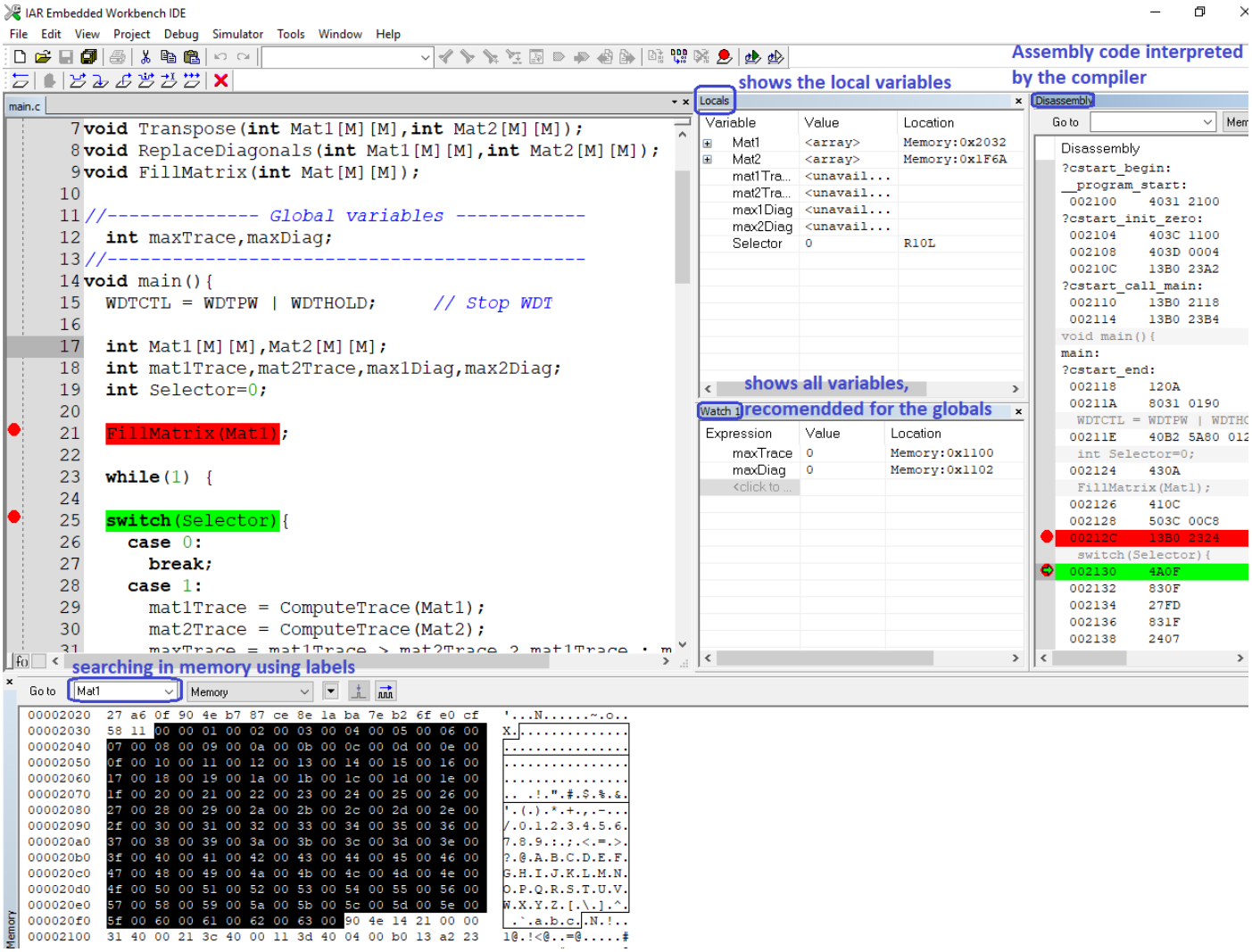


- לבחינת פעולת הקוד נפתח את חלונות המתארים את הזיכרון הראשי, המשתנים, disassembly, וכו'.

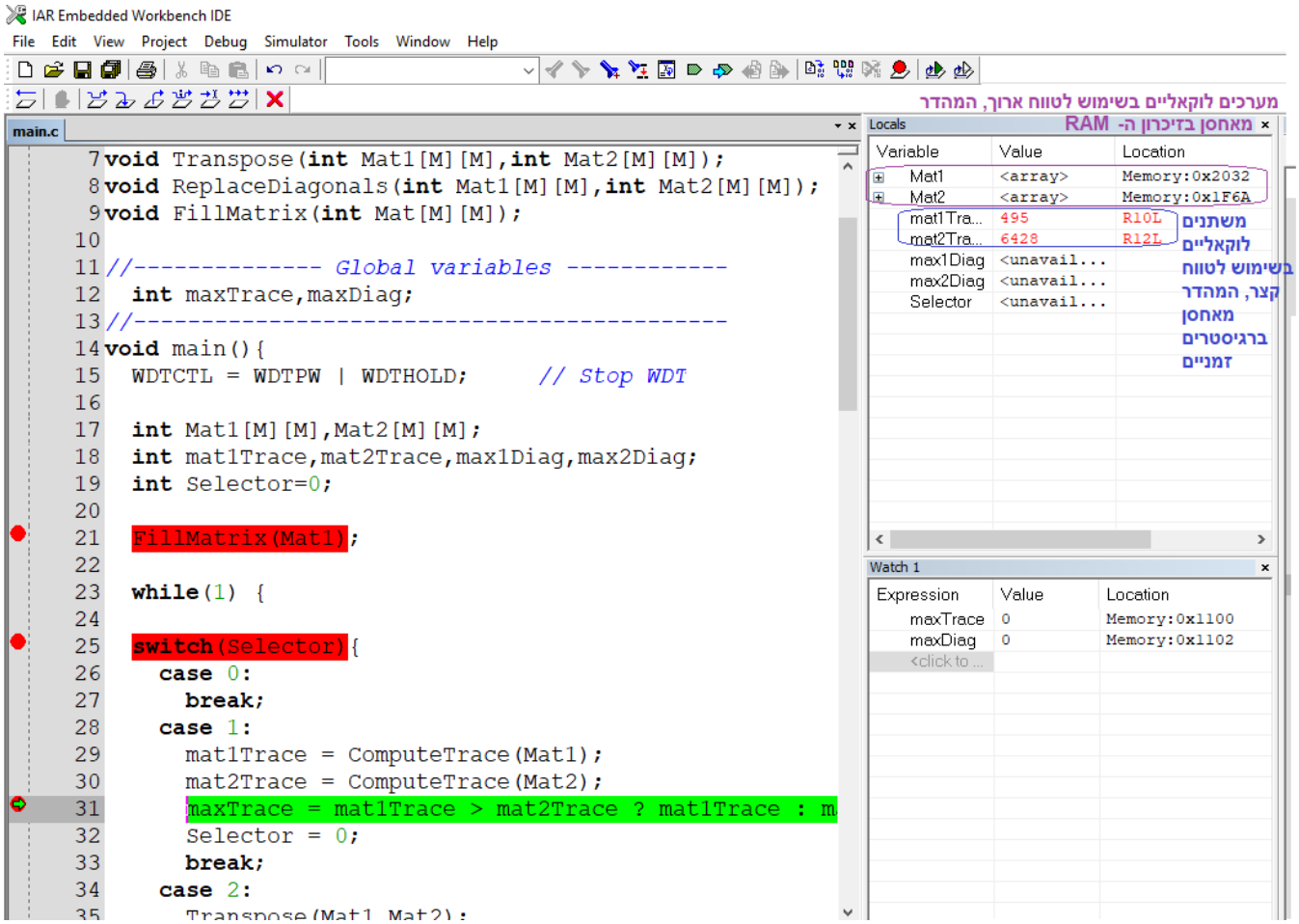
ע"י בחירה מהתפריט הבא:



- לבחינת הפונקציה FillMatrix נצעד לשורה 21 בקוד (נמקם בה נקודת עצירה) ע"י F5 לשורה הממלאת את Mat1 לפי הנוסחה $\mathbf{Mat1[i][j] = i \cdot N + j}$.



- לבחינת המשתנים כאשר ערכו של משתנה Selector=1 (הכניסו ערך 1 דרך חלון locals) תצעדו לפקודת חישוב ה-Trace של המטריצות (שורה 31) ובחנו את ערך המשתנים הלוקאליים והגלובאליים.



המשיכו כך לשאר הסעיפים ובחנו את הקוד בצורה מעמיקה יותר בכדי להבין טוב יותר את התוכנית.

C. שכבות קוד והפרדה בצורה נכונה לצורך תכלול ותחזוקה של המערכת:

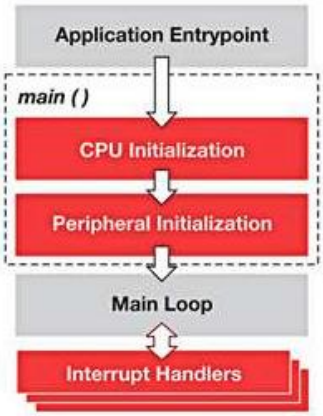
מערכת Embedded היא למעשה מערכת משובצת מחשב המבוססת על MCU (Micro Controller Unit) המכיל מעבד המחובר לזיכרון ורכיבים פריפריאליים. בחלק היישומי של קורס "מבוא למחשבים" התנסיתם באופן בסיסי בתכנון מערכת משובצת מחשב על בסיס MCU מסוג MSP430 כאשר קוד המערכת נכתב בשפת Assembly שמשמעותה ניהול קוד המערכת בצורה מוגבלת, קשה לתכלול, תחזוק והגנה (היתרון הגדול היה שבתכנון נכון התקורה של ניהול הקוד יכולה להיות מינימאלי וכך ניתן לעמוד בדרישות Hard Real Time נוקשות).

לצורך תכנון מערכת Embedded מורכבת יש צורך לנהל את קוד המערכת בחלוקה של שכבות קוד נכונה כך שהקשר בין שכבות הקוד הוא ע"י API (Application Programming Interface). שפות הקוד הנפוצות לשימוש הן שפות C/C++.

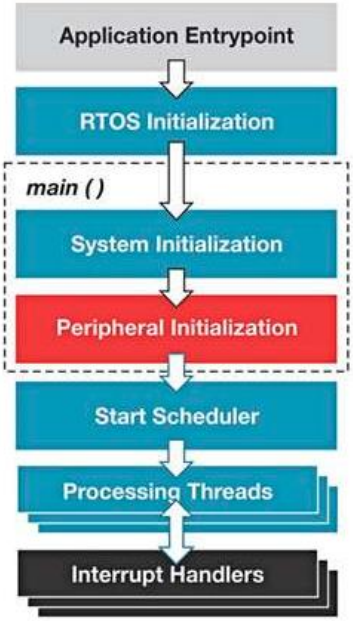
- ***שפת C - שפה פרוצדוראלית*** (שפה המבוססת על פונקציות אשר פועלות על מבני נתונים חופשיים בכל מרחב הקוד) ועל כן כתיבת קוד המערכת בשפה זו תהיה בגישה פרוצדוראלית. היתרון הגדול של שפה זו היא שנפח תקורת הקוד (נפח קוד אסמבלי המתורגם ע"י הקומפיילר) קטן יותר ולכן גם מהיר יותר מאשר שפה התומכת ב OOP (כמו C++).
- ***שפת C++ - שפה מונחית עצמים*** (שפה המבוססת פרדיגמת תכנותית הנקראת תכנות מונחה עצמים, OOP המבוסס על שלושה עקרונות– Encapsulation, Inheritance, Polymorphism). בגישה זו מרחב הקוד הוא למעשה מרחב של אובייקטים תכנותיים בעלי יחסים היררכיים ביניהם וכל ישות תכנותית היא אובייקט (של מחלקה) בעלת מאפיינים ופעולות משלה הקיימת כיחידה סגורה ועצמאית.

הערה חשובה: תכנות מונחה עצמים אינו כינוי לשפת תכנות אלא לפרדיגמה תכנותית ולכן ניתן לתכנת בגישת OOP גם בשפת C (מאחר ושפה זו פרוצדורלית ניהול הקוד יהיה "טיפה" מורכב).

קוד המערכת משובצת מחשב מחולק לשני סוגים, מערכת Embedded מבוססת מערכת הפעלה OS (חלקית / מלאה) או שאינה מבוססת OS (Bare-metal). ***בקורס זה אנו נלמד לתכנן מערכת Embedded מסוג Bare-metal.***



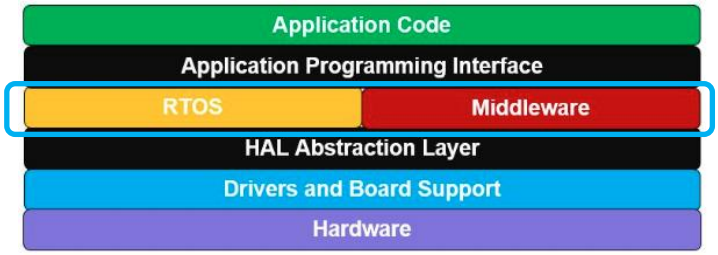
Software flow diagram for bare-metal application



Software flow diagram for RTOS application

תכנון מערכת Embedded בצורה נכונה, מורכב משכבות, כאשר הקשר בין השכבות הוא באמצעות API.

i. שכבת ה- Hardware מכילה קוד הקשור לקנפוג אופני עבודה של המעבד, מצב עבודה של שעון MCLK, SMCK, אתחול טבלאות interrupt vectors רמות העדיפות וכו'
ii. שכבת BSP (Board Support Package) מכילה קוד לקנפוג רגיסטרים של הרכיבים הפריפריאליים השונים של הבקר, קנפוג הבסים, תדרי העבודה שלהם וכו.
iii. שכבת ה- HAL (Hardware Abstraction Layer) מכילה קוד המנהל את הממשק עם הרכיבים הפריפריאליים.
iv. שכבת ה- API מכילה את הקוד בו 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.
v. שכבת ב- Application היא כבר שכבת קוד הגבוהה ביותר בה מתקיים הממשק עם המשתמש.



This layer is excluded in bare-metal software approach

D. העולם שלפני ה Startup Code – main:

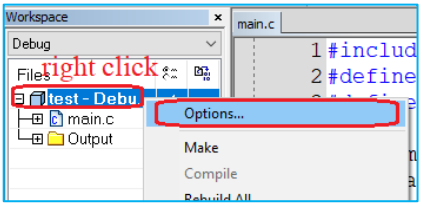
במערכת Embedded הכתובה בשפת C, ישנו קוד הרץ לפני הגעה ערך רגיסטר PC לפונקציית ()main. קוד זה נקרא startup code המצורף ע"י הקומפיילר ומסופק כחלק מה- IDE ותפקידו לבצע אתחול של סגמנטי מידע השונים בזיכרון המערכת (RAM, STACK, vector table).

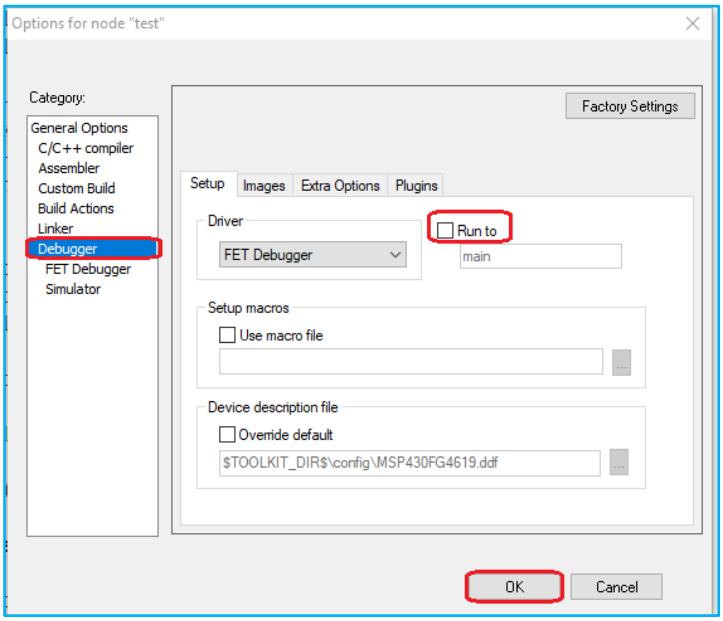
קוד זה מכיל רוטינות המעתיקות את ערכי המשתנים הגלובאליים מה FLASH לזיכרון ה RAM במצב של RESET (משתנים גלובאליים לא מאותחלים בקוד, ערכם יאותחל לאפס) עם ערכי ה- Hard coded אותם הגדרנו בקוד.

במקרה של חריגות (Exceptions=NMI) כתוצאה מ- Stack over flow, unaligned access to memory, BUS fault, etc. רוטינות קוד של exception handlers מסופקות עם ה- IDE כחלק מה- startup code ומכילות לולאה אינסופית (ביכולתנו לכתוב קוד בגוף exception handlers אלו).

נושא של ה- startup code הינו נושא חשוב אך בקורס זה הוא מובא לצורך ידיעת קיומו והיכרותו בצורה בסיסית.

<u>כדי להגיע לקוד של startup code לפני הגעת PC לפונקציית main, ניתן לבצע את השלבים הבאים:</u>





Run the program using Debug mode

E. שאלות חלק תיאורטי – הרצת קוד לדוגמה בסביבת IAR ו- CCS (שימו לב: שאלות 2-8 נדרש מענה נפרד, עבור סביבת IAR ועבור סביבת CCS):

1. הסבר את ההבדל בין משתנים גלובליים ומשתנים לוקאליים.
   רשום דוגמה למשתנה מכל סוג מהקוד לדוגמה, ציין מה הסקופ של כל אחד מהם.
2. מה כתובת המערך Mat2 בזיכרון ומה טווח הכתובות אותו הוא מכסה. מהו סוג זיכרון זה ?
3. רשום את כתובת תחילת מיקום המחסנית בזיכרון הנקבע ע"י המהדר.
4. רשום את תוכן SP כאשר רגיסטר PC מצביע על הפקודה הראשונה של פונקציה ComputeTrace .
5. רשום את כתובת הפונקציה FillMatrix בזיכרון.
   מה גודל קוד הפונקציה FillMatrix בבתים ? מהו סוג זיכרון זה ?
6. מהו זמן ריצת הקוד של הפונקציה FillMatrix ביחידות מחזור של MCLK.
7. מהו ה- scope של משתנה mat2Trace בתוכנית, מהו מיקומו בזמן ה- ה- scope.
8. רשום את קוד האסמבלי המתורגם ע"י המהדר עבור שורת הקוד הבאה:

```
maxTrace = mat1Trace > mat2Trace ? mat1Trace : mat2Trace;
```

בהצלחה.