

**תוכן עניינים**

- A. הקדמה: ..... 2
- B. הקדמה ופתיחת פרויקט חדש בסביבת פיתוח CW IDE: ..... 3
- C. הידור ובניית הפרויקט (Building Project): ..... 7
- D. מצב Debug: ..... 8
- E. תיאור משימת קטע קוד לדוגמא: ..... 10
- F. ביצוע DEBUG - תוכנית לדוגמא: ..... 11
- G. שכבות קוד והפרדה בצורה נכונה לצורך תכלול ותחזוקה של המערכת: ..... 13
- H. העולם שלפני ה main – Startup Code: ..... 14
- I. שאלות חלק תיאורטי – בנושא קוד לדוגמה: ..... 16

## שליבי פתיחת פרויקט בסביבת CodeWarrior

### A. הקדמה:

חלק זה מהווה הקדמה לדו"ח מכין של ניסוי מעבדה 2 ובו תצטרכו להריץ קוד נתון בשפת C בסביבת הפיתוח הנקראת CodeWarrior ולענות על שאלות תיאורטיות בלבד.

באופן כללי סביבת הפיתוח משמשת ליצירת קוד מכונה (בינארי) מתוך טקסט (קוד) הנרשם ב-Editor של סביבת הפיתוח שאותו יש לצרוב לתוך זיכרון הבקר לצורך ביצוע התוכנית.

- 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 המעבד מול רכיבים פריפריאליים מתקדמים (DMA, Communication methods, Flash Memory controller). כתיבת הקוד תהיה בשפת C. בגישה של כתיבת 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 IDE **CW** (IDE = Integrated Development Environment) בה נעבוד בקורס. 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. 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.

### בפיתוח קוד בסביבת הפיתוח ישנם 3 שלבים:

- ♦ **סימולציה** – סביבת הפיתוח משמשת סימולטור לבקר שלנו. את הקוד שכתבנו נפעיל **במצב סימולטור** והוא ירוץ בפועל ב-PC (מחשב אישי) בלבד, לצורך דימוי הבקר.  
**הערה:** מצב סימולציה אינו קיים בסביבת IDE **CW** (היא קיימת בסביבת IDE **IAR** אולם היא מוגבלת 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').

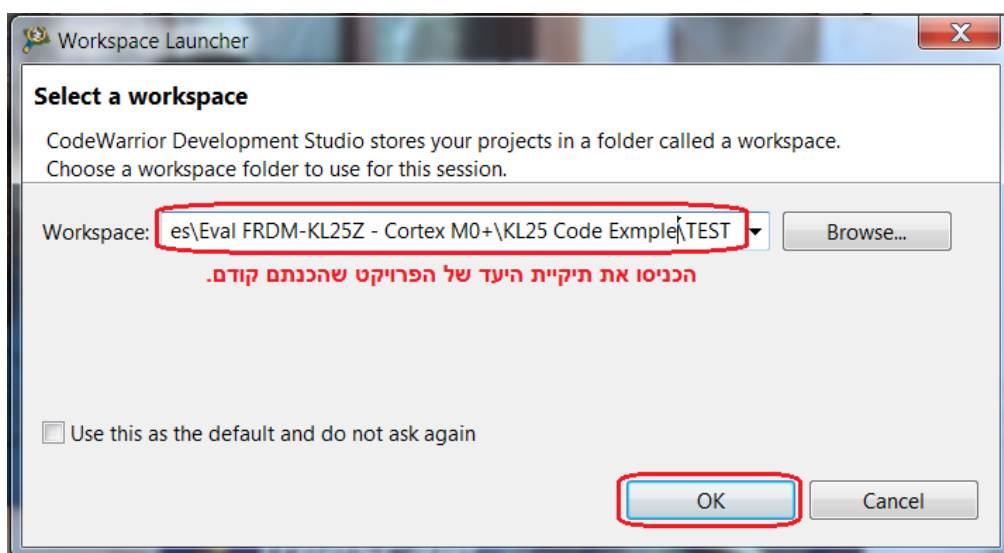
- ♦ **Active Application** – הקוד שכתבנו ייצרב לבקר (מה-PC) ובהפעלתו הוא ירוץ בבקר בלבד ללא קשר עם ה-PC (בשונה ממצב DEBU). מצב זה מונה גם Stand Alone, מאחר ובמצב זה הבקר בפני עצמו ללא קשר ל-PC.

## B. הקדמה ופתיחת פרויקט חדש בסביבת פיתוח CW IDE:

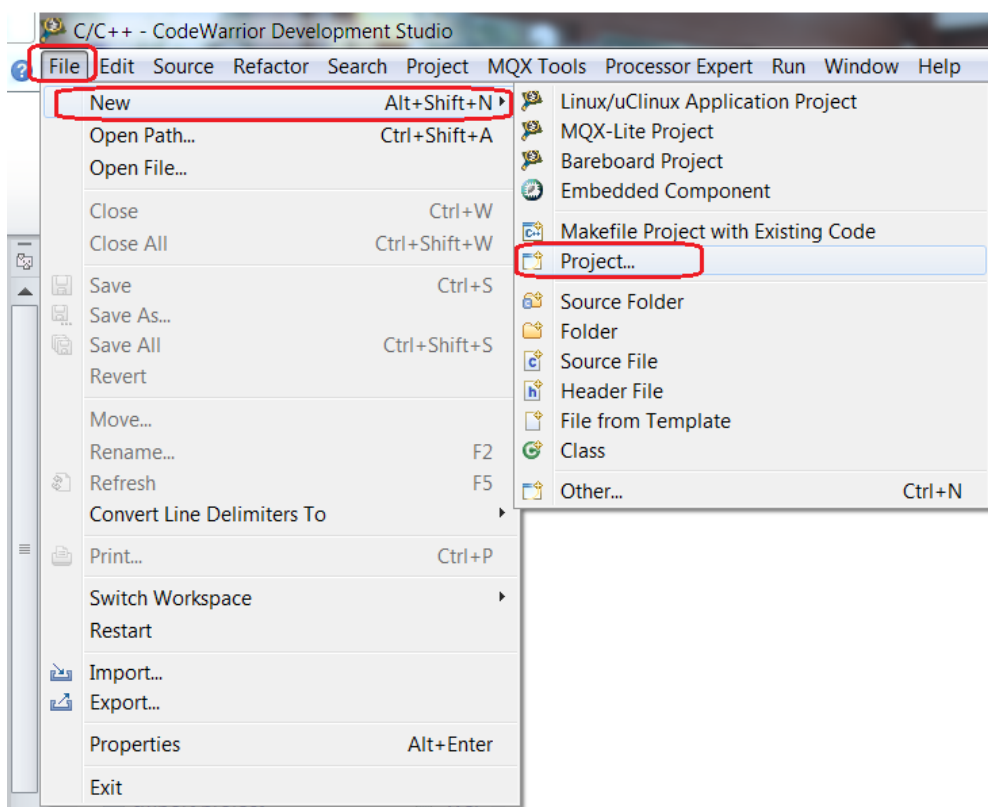
1. תוכנת CW IDE סביבת הפיתוח המותקנת במעבדה, ניתן להורידה ולהתקינה במחשבכם האישי.  
ניתן להורדה מהקישור הבא:

### Code Warrior IDE setup

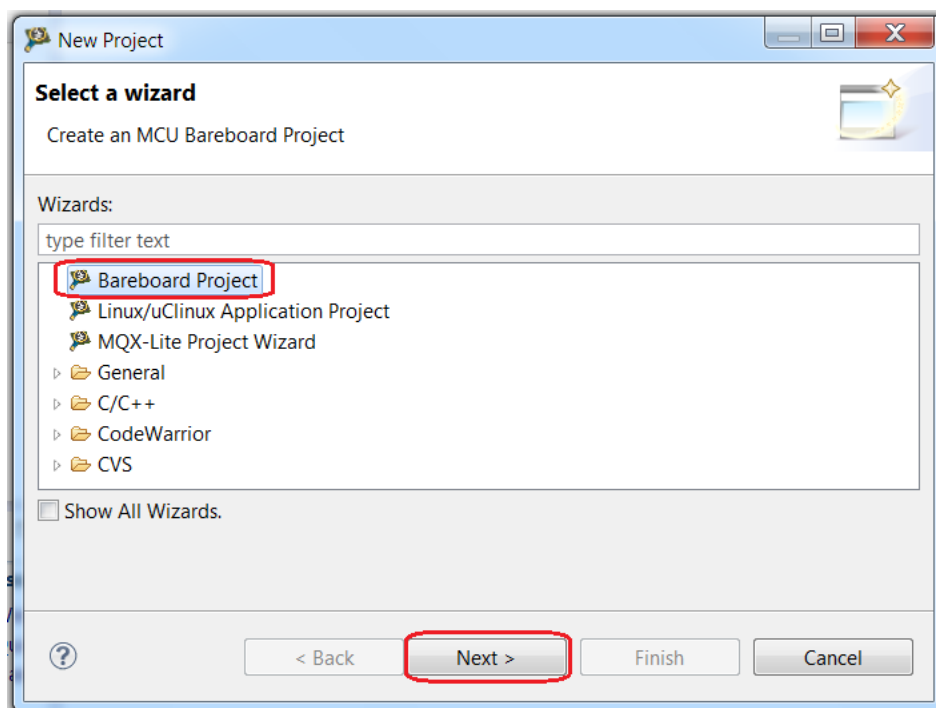
2. נכין במחשב תיקייה לפרויקט תבחרו שם TEST (בכונן שנבחר). לאחר מכן נפתח את תוכנת CodeWarrior וצריך להיפתח חלון הבא.



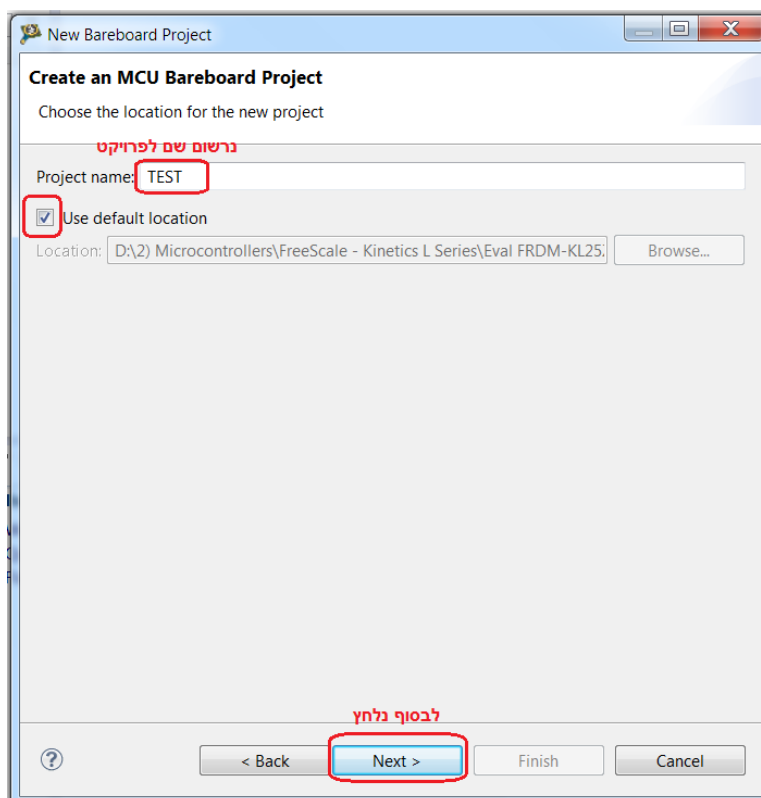
#### File → New → Project .4



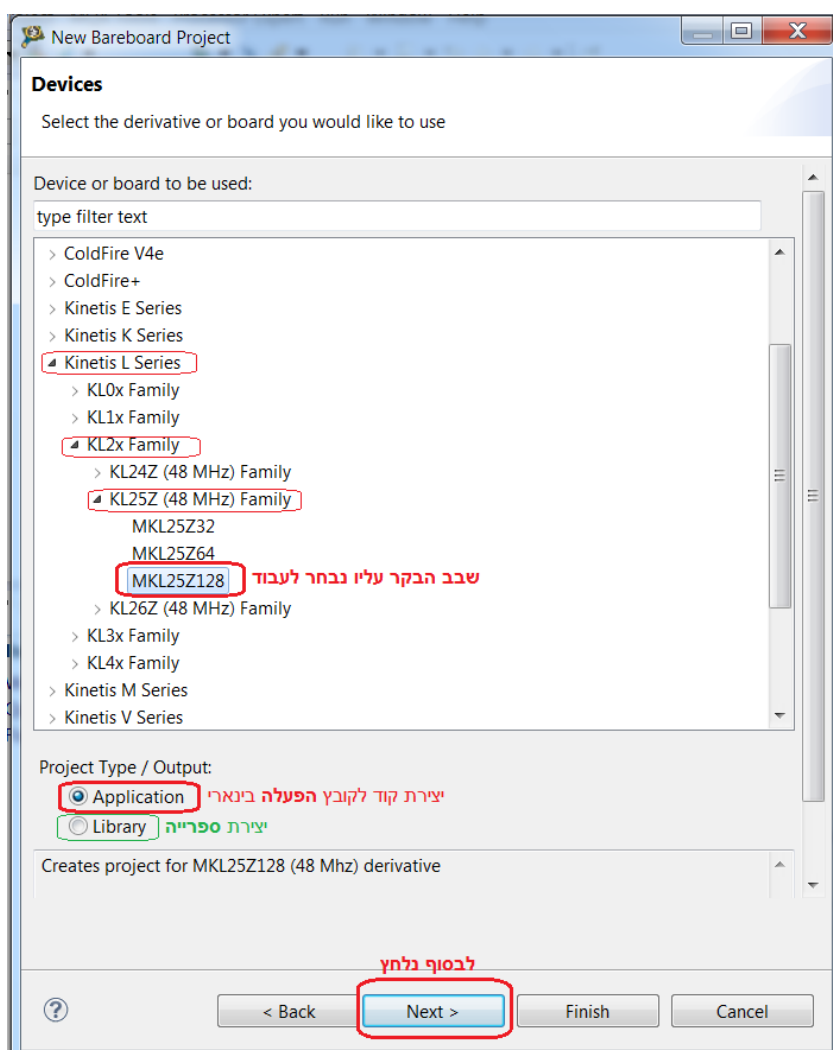
#### Bareboard project → Next :נפתח חלון בשם "New Project" ובו נלחץ: .5



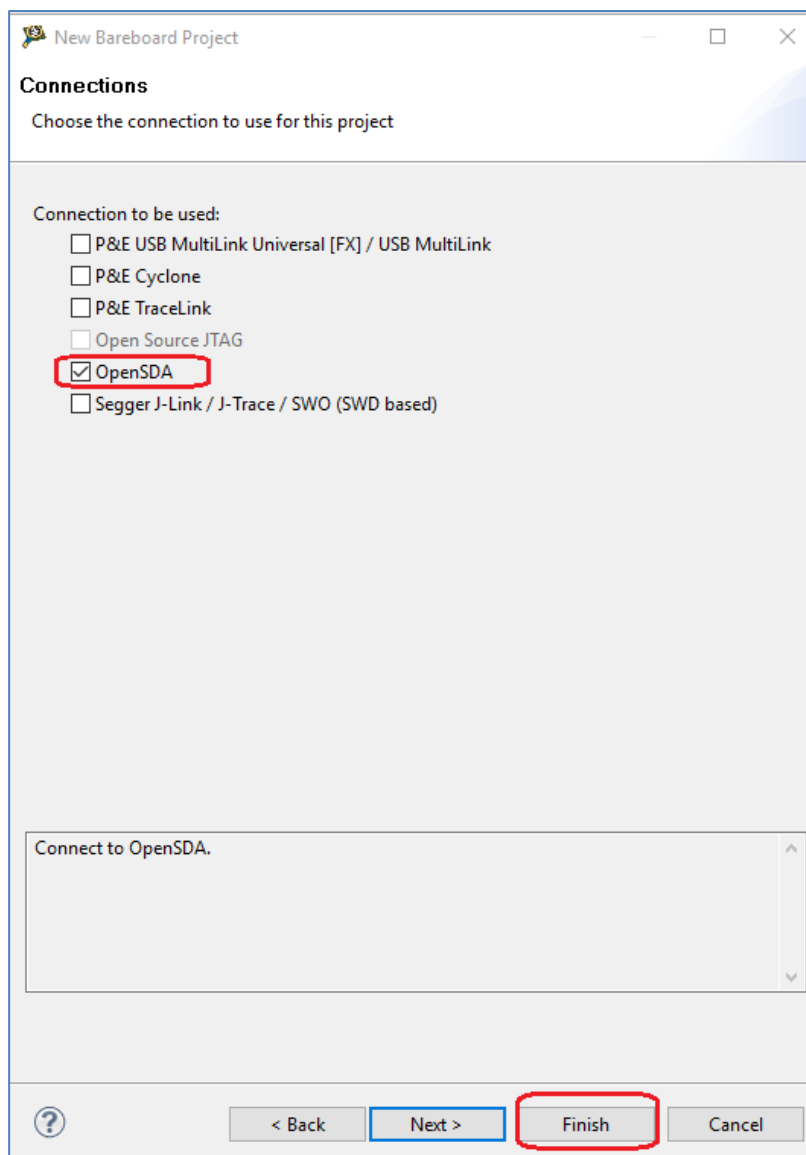
6. נפתח חלון בשם "New Bareboard Project" ובו נרשום שם לפרויקט ולאחר מכן נלחץ Next.



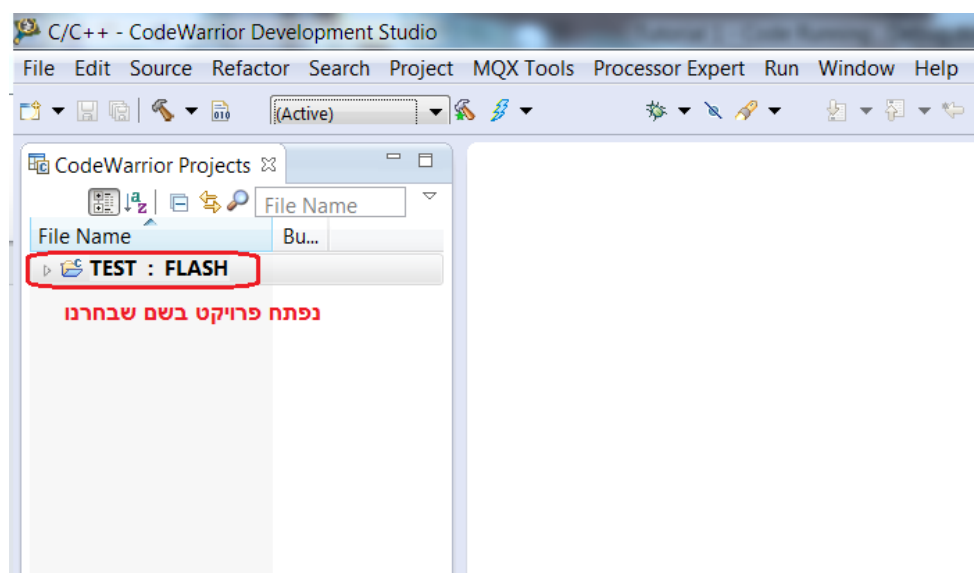
7. ביצוע בחלון הבא:



## 8. ביצוע בחלון הבא:



## 9. נפתח פרויקט חדש בשם שבחרנו.



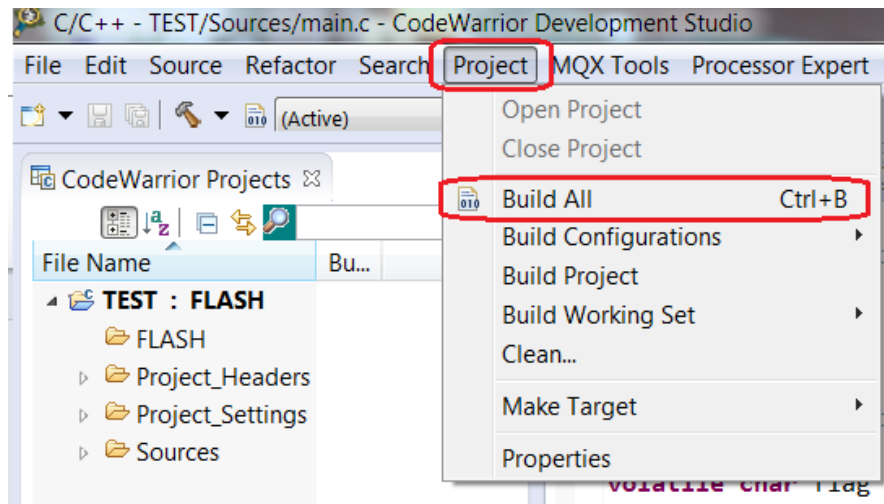
**10. קובצי ברירת מחדל:**

בלחיצה על שם הפרויקט (ראה סעיף 9) תוכל לראות **שהקבצים הבסיסיים** (מלבד קובצי האתחול הנמצאים בתיקיית Startup\_Code → Project\_Settings) **הנוצרים בעת פתיחת פרויקט חדש הם:**

- בתיקיית **Source** נוצרים הקבצים – **main.c** , **BoardSupport.c**
- בתיקיית **Project\_Headers** נוצרים הקבצים – **bme.h** , **derivative.h** , **MKL25Z4.h**

**11. הרצת קוד דוגמה בשפת C – לצורך מענה על שאלות בסעיף I:**

נדרש להעתיק את תוכן קובץ **main.c** דרך הקישור ולדרוס את תוכן קובץ **main.c** בפרויקט שפתחתם. תיאור המשימה שמבצע קטע קוד לדוגמא מופיע **בסעיף E**.

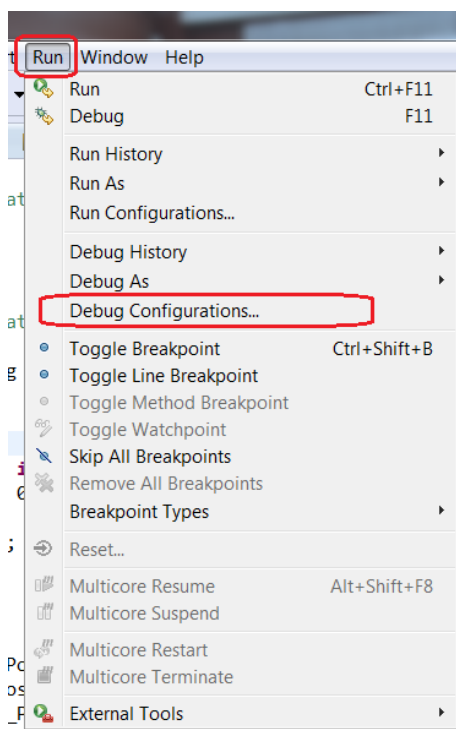
**C. הידור ובניית הפרויקט (Building Project):**

D. מצב Debug:

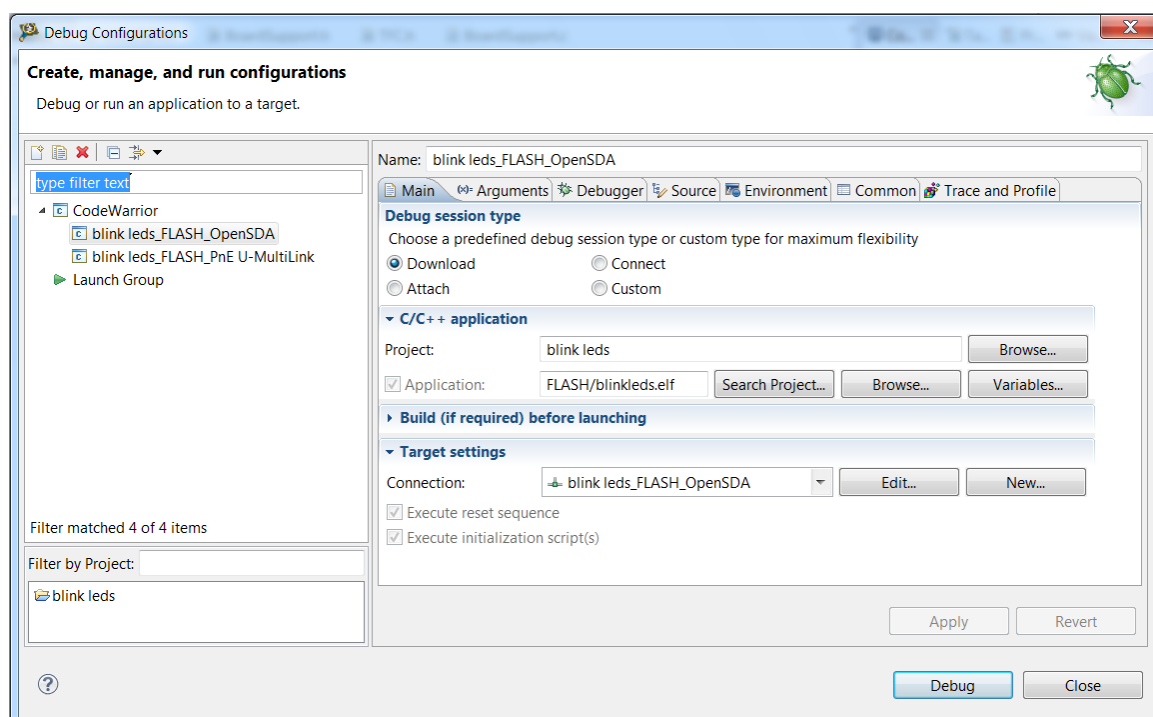
בסביבת הפיתוח CodeWarrior **לא קיים סימולטור (ביצוע הדמיה לבקר ע"י ה-PC) עבור סדרת בקרים**

**מסוג Kinetics series**. לכן, קיימים 2 מצבים בלבד, מצב **DEBUG** ומצב **Active Application**.

במצב **DEBUG** (צריבת קוד לזיכרון ה-FLASH של הבקר והרצתו בבקר בלבד עם יכולות **DEBUG** דרך ה-PC) באפשרותנו להריץ את הקוד במצב אמת כאשר הקוד רץ בבקר ולא ב-PC, אולם השליטה על הרצת הקוד (נקודות עצירה, הרצה בצעדים וכו') היא דרך ה-PC.

1. לשינוי הגדרות ברירת מחדל של ה-Debugger


זוהי הגדרת ברירת המחדל.

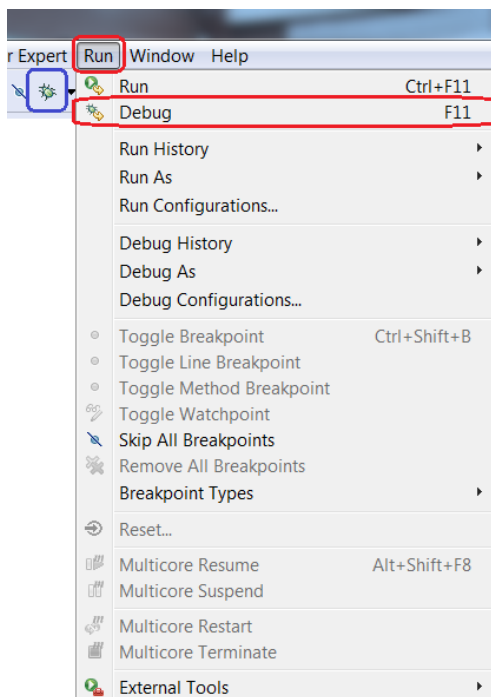




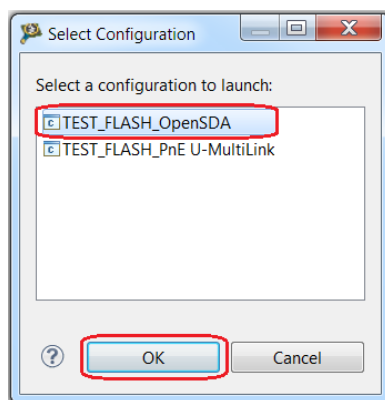
2. הכרטיס עצמו מחובר למחשב דרך USB. לצריבת הקוד לזיכרון הבקר לצורך הרצה, נדרש להתקין Driver. הסבר להתקנה נמצא בקובץ "FRDM-KL25Z Quick Start Guide" (בתיקיית מעבדה Lab2).

3. ביצוע DEBUG:

Run → Debug **OR** F11 **OR** push on 



4. בחלון שנפתח יש לפעול כמו בצילום הבא.



5. נכנסתם למצב DEBUG, אתם יכולים לבצע הרצה של הקוד ולבחון את פעולת הבקר (כמו שלמדתם במעבדת "מבוא למחשבים").

**E. תיאור משימת קטע קוד לדוגמא:**

התוכנית מגדירה בזיכרון ה-RAM מערך דו-מימדי בגודל  $N \times N$  ומאתחלת את אברי המטריצה בערכים מ-0 עד 99 לפי הנוסחה הבאה,  $Mat1[i][j] = i \cdot N + j$ . 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 שינה שעדיין לא למדתם בבקר זה) ובהתאם לערכו מבצעת על המטריצה **Mat1** פעולה מתאימה מהתפריט הבא (כתיבת תוצאת הפעולה תיכתב למטריצה **Mat2** באותם הממדים של **Mat1**):

1. כאשר **Selector = 0** : לא נעשה כלום.

2. כאשר **Selector = 1** : חישוב עקבה של המטריצה וכתיבתה לתוך משתנה בשם **Trace**.

3. כאשר **Selector = 2** : ביצוע transpose למטריצה (כתיבה למטריצה **Mat2**).

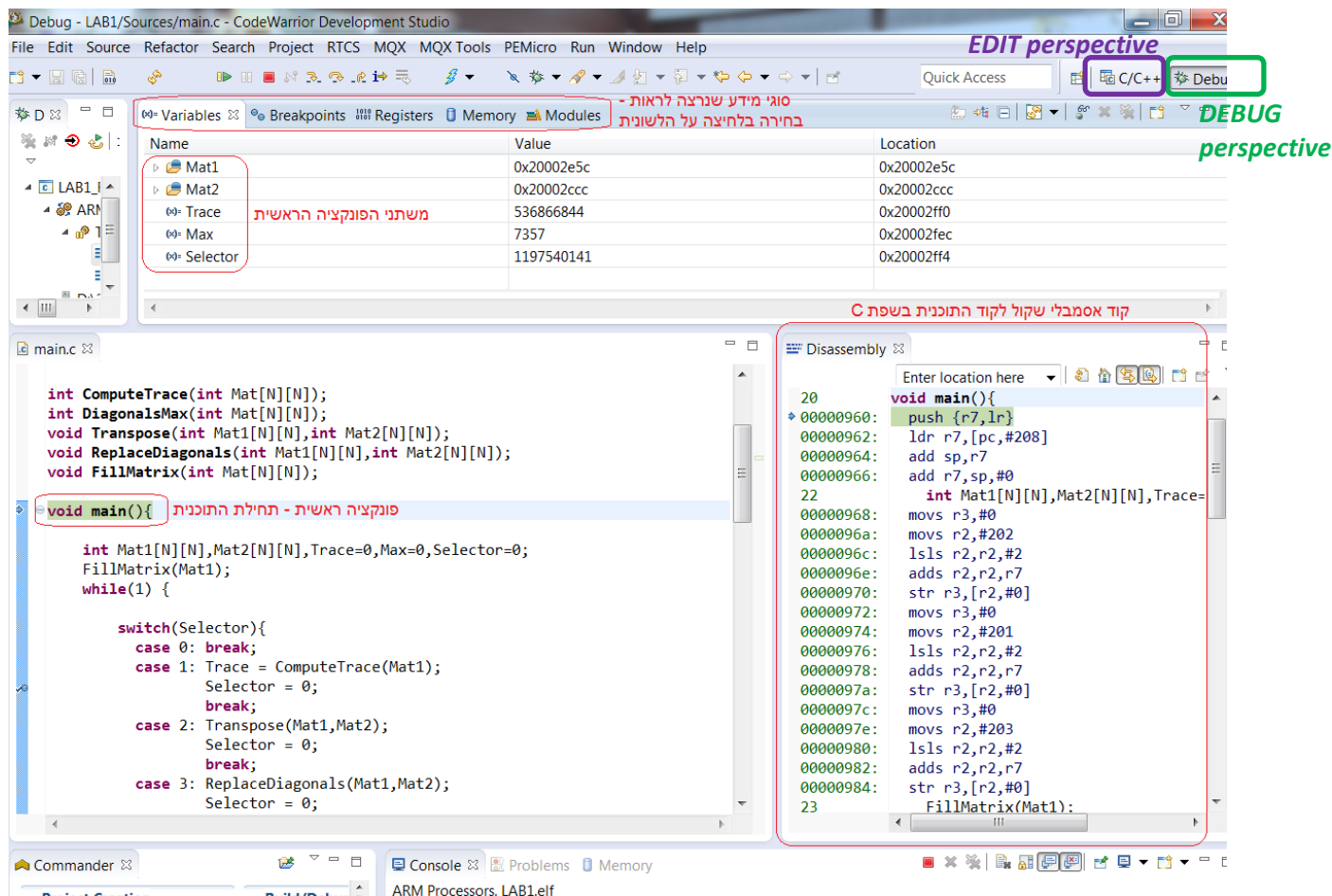
4. כאשר **Selector = 3** : החלפת אלכסוני המטריצה, בין אלכסון ראשי למשני (כתיבה למטריצה **Mat2**).

5. כאשר **Selector = 4** : חישוב ערך מקסימאלי בין 2 אלכסוני המטריצה וכתיבתו לתוך משתנה בשם **Max**.

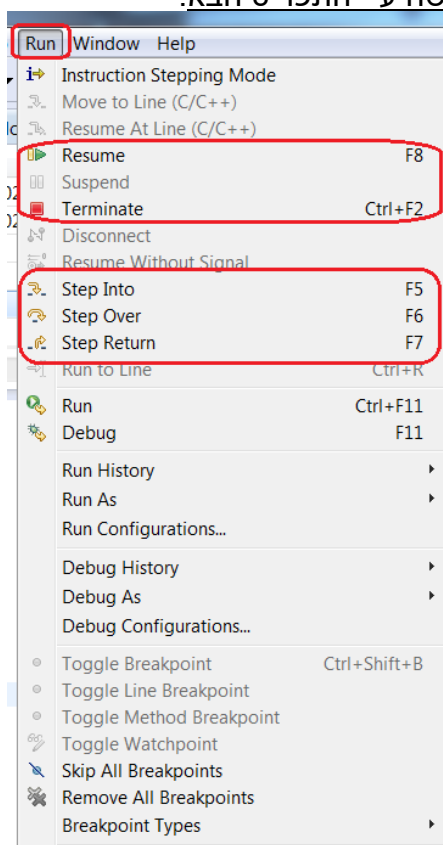
## F. ביצוע DEBUG - תוכנית לדוגמא:

תזכורת: בפרויקט שפתחתם, נתבקשתם להחליף את קובץ **main.c** הנמצא בתיקיית הפרויקט בתיקיית **Source** עם קובץ **main.c** הנמצא באתר Moodle בתיקיית LAB1.

1. לאחר סיום צריבת התוכנית לבקר באופן עבודה **DEBUG**, חץ ההרצה הירוק יעמוד על שורת **main**.



2. ריצה בצעדים תיעשה ע"י התפריט הבא:



3. בצעד ע"י F5 לשורה הממלאת את Mat1 לפי הנוסחה  $Mat1[i][j] = i \cdot N + j$ .

Memory window (Virtual:Mat1):

Address	0 - 3	4 - 7	8 - B	C - F
20002E50	4A6C4BB6	1FDD2273	00000000	00000000
20002E60	00000001	00000002	00000003	00000004
20002E70	00000005	00000006	00000007	00000008
20002E80	00000009	0000000A	0000000B	0000000C
20002E90	0000000D	0000000E	0000000F	00000010
20002FA0	00000011	00000012	00000013	00000014

Source code (main.c):

```
#define max(x,y) x>y ? x : y

int ComputeTrace(int Mat[N][N]);
int DiagonalsMax(int Mat[N][N]);
void Transpose(int Mat1[N][N],int Mat2[N][N]);
void ReplaceDiagonals(int Mat1[N][N],int Mat2[N][N]);
void FillMatrix(int Mat[N][N]);

void main(){
    int Mat1[N][N],Mat2[N][N],Trace=0,Max=0,Selector=0;
    FillMatrix(Mat1);
    while(1){
        switch(Selector){
            case 0: break;
            case 1: Trace = ComputeTrace(Mat1);
                    Selector = 0;
                    break;
            case 2: Transpose(Mat1,Mat2);
                    Selector = 0;
                    break;
            case 3: ReplaceDiagonals(Mat1,Mat2);
                    break;
        }
    }
}
```

Disassembly window:

```
00000996: movs r2,#203
00000998: lsls r2,r2,#2
0000099a: adds r2,r2,r7
0000099c: ldr r3,[r2,#0]
0000099e: cmp r3,#4
000009a0: bhi main+0x34 (0x994) ; 0x00
000009a2: movs r2,#203
000009a4: lsls r2,r2,#2
000009a6: adds r2,r2,r7
000009a8: ldr r3,[r2,#0]
000009aa: lsls r2,r2,#2
000009ac: ldr r3,[pc,#136]
000009ae: adds r3,r2,r3
000009b0: ldr r3,[r3,#0]
000009b2: cpy pc,r3
28 case 1: Trace = ComputeTra
000009b4: movs r2,#202
000009b6: lsls r2,r2,#1
000009b8: adds r3,r7,r2
000009ba: mov r0,r3
000009bc: bl ComputeTrace (0xa3c) ; 0x000
000009c0: mov r3,r0
```

4. בלשונית המשתנים שנו את ערכו של משתנה Selector לערך 1 תצעדו לפקודת חישוב ה-Trace של המטריצה וראו שהמשתנה מקבל ערך חדש,  $Trace=495$ .

Variables window:

Name	Value	Location
Mat1	0x20002e5c	0x20002e5c
Mat2	0x20002ccc	0x20002ccc
Trace	495	0x20002ff0
Max	0	0x20002fec
Selector	1	0x20002ff4

Source code (main.c):

```
void main(){
    int Mat1[N][N],Mat2[N][N],Trace=0,Max=0,Selector=0;
    FillMatrix(Mat1);
    while(1){
        switch(Selector){
            case 0: break;
            case 1: Trace = ComputeTrace(Mat1);
                    Selector = 0;
                    break;
            case 2: Transpose(Mat1,Mat2);
                    Selector = 0;
                    break;
            case 3: ReplaceDiagonals(Mat1,Mat2);
                    Selector = 0;
                    break;
            case 4: Max = DiagonalsMax(Mat1);
                    Selector = 0;
                    break;
        }
    }
}
```

Disassembly window:

```
000009bc: bl ComputeTrace (0xa3c) ; 0x000
000009c0: mov r3,r0
000009c2: movs r2,#202
000009c4: lsls r2,r2,#2
000009c6: adds r2,r2,r7
000009c8: str r3,[r2,#0]
29 Selector = 0;
000009ca: movs r3,#0
000009cc: movs r2,#203
000009ce: lsls r2,r2,#2
000009d0: adds r2,r2,r7
000009d2: str r3,[r2,#0]
30 break;
000009d4: b main+0xd2 (0xa32) ; 0x000
31 case 2: Transpose(Mat1,Mat
000009d6: movs r3,#202
000009d8: lsls r3,r3,#1
000009da: adds r2,r7,r3
000009dc: adds r3,r7,#4
000009de: mov r0,r2
000009e0: mov r1,r3
000009e2: bl Transpose (0xb6c) ; 0x000
```

5. המשיכו כך לשאר הסעיפים בכדי להבין טוב יותר את התוכנית.

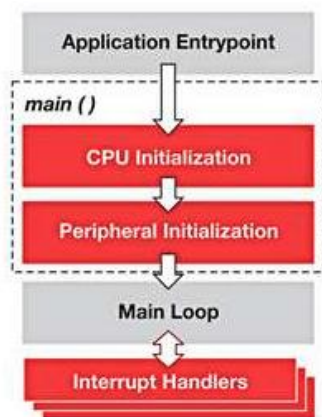
## G. שכבות קוד והפרדה בצורה נכונה לצורך תכלול ותחזוקה של המערכת:

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).

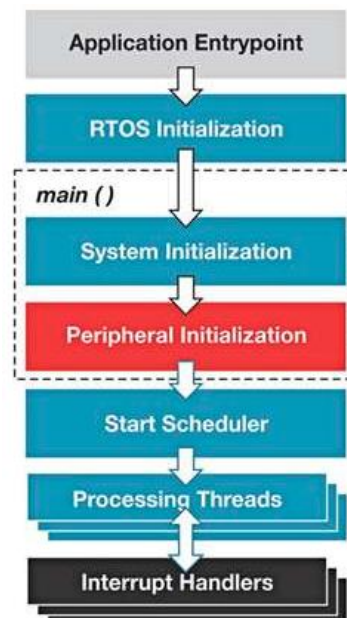
לצורך תכנון מערכת Embedded מורכבת יש צורך לנהל את קוד המערכת בחלוקה של שכבות קוד נכונה כך שהקשר בין שכבות הקוד הוא ע"י API (Application Programming Interface). שפות הקוד הנפוצות לשימוש הן שפות C/C++.

- **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 המבוססת 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.**



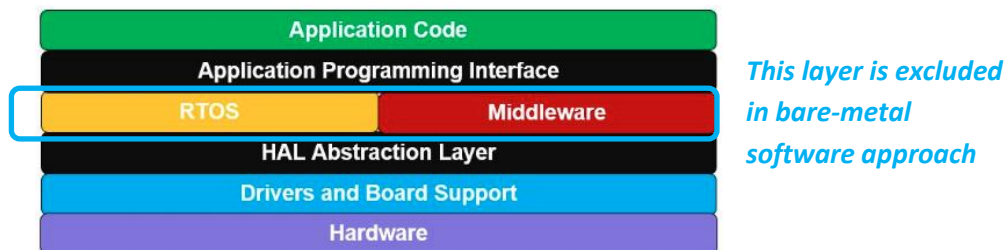
Software flow diagram for bare-metal application



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 הבסיס, 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 ה-Application היא כבר שכבת קוד הגבוהה ביותר בה מתקיים הממשק עם המשתמש.



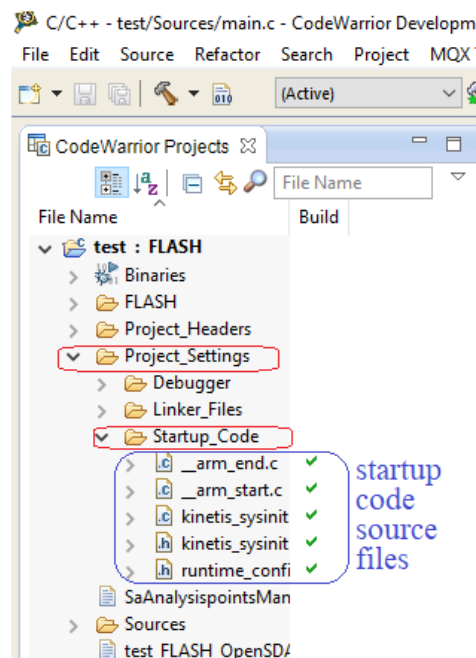
## H. העולם שלפני ה-main – Startup Code

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

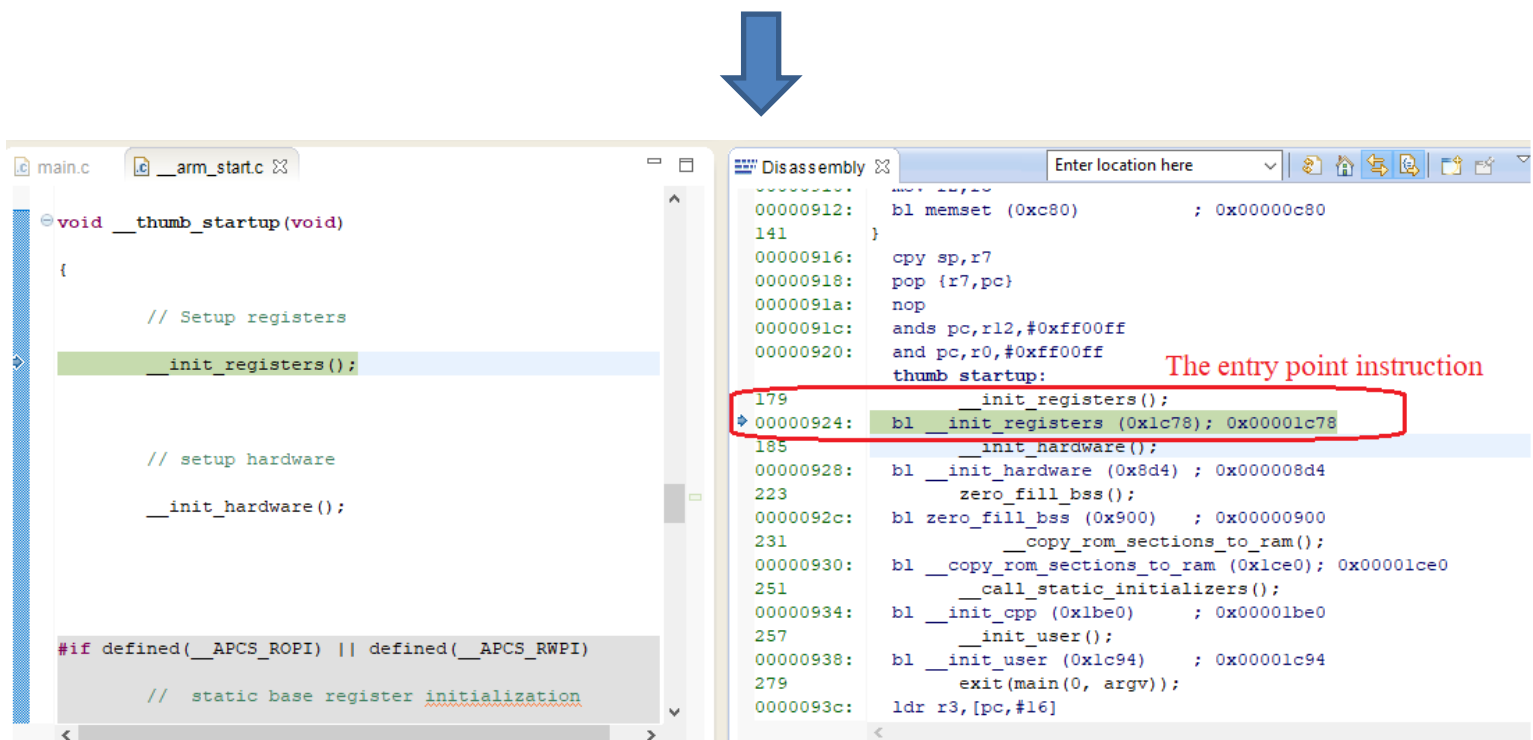
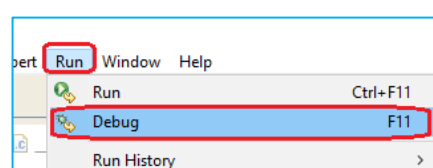
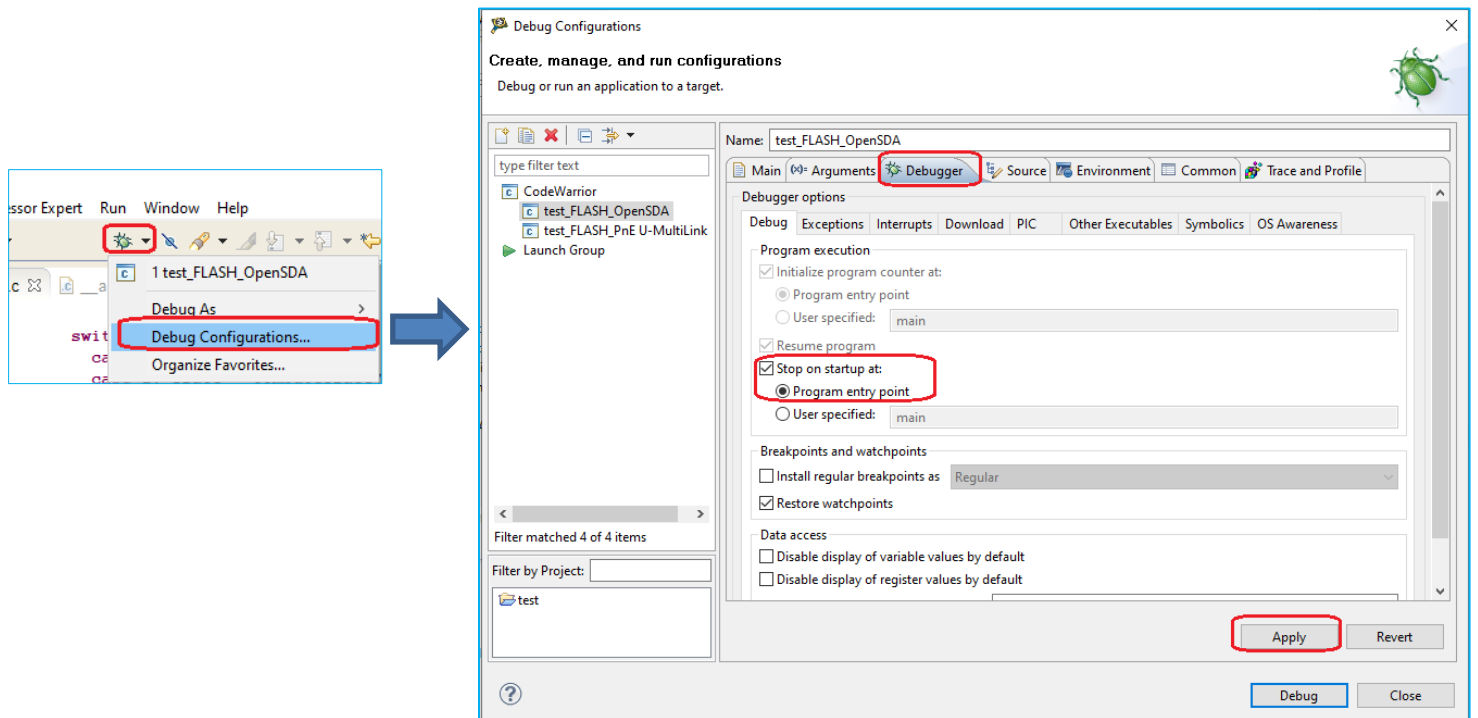
קוד זה מכיל רוטינות המעתיקות את ערכי המשתנים הגלובאליים מה FLASH לזיכרון ה RAM במצב של RESET (משתנים גלובאליים לא מאותחלים בקוד, ערכם יאותחל לאפס) עם ערכי ה- Hard coded אותם הגדרנו בקוד.

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.



כדי להגיע לקוד של startup code לפני הגעת PC לפונקציית main, ניתן לבצע את השלבים הבאים:  
הגדירו את ה- DEBUGGER שה entry point שלו לא תדלג על קטע ה- startup code.



ניתן להריץ צעד אחר צעד לחייווי קטע קוד של ה- startup code עד להגעה לפונקציה הראשית main() של הקוד שלכם.

**א. שאלות חלק תיאורטי – בנושא קוד לדוגמה:**

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

$$\text{maxTrace} = \text{mat1Trace} > \text{mat2Trace} ? \text{mat1Trace} : \text{mat2Trace};$$

**בהצלחה.**