





















## חלק תיאורטי – עבור IAR.

1. משתנה הינו מקום בזכרות שיכול להכיל מידע ממגוון סוגים, אשר נעשה בו שימוש בקטע מהקוד, או בקוד כולו. החלק בקוד בו נעשה שימוש במשתנה נקרא הסקופ (scope) של המשתנה.  
 משתנה גלובלי הינו משתנה שמוכר לכל חלק בקוד. משתנה כזה יוגדר בצורה שכל הפונקציות יוכלו לגשת אליו ולשנות אותו, הסקופ של משתנה כזה הוא כל הקוד.  
 משתנה לוקאלי או מקומי הוא משתנה שיוגדר בפונקציה ספציפית ורק בתוכה נוכל לגשת אליו ולשנות אותו. יתרון אחד של משתנה כזה הוא שניתן "למחזר" אותו, כגון המשתנה i שבא לידי שימוש בלולאות שונות לכל אורך הקוד, מבלי לדרוס אחד את השני.  
 דוגמה למשתנה גלובלי בקוד לדוגמה הוא maxTrace הסקופ שלו הוא כל הקוד. דוגמה למשתנה לוקלי הוא selector, הסקופ שלו הוא פונקציית ה-main.  
 2. כתובת המערך Mat2 בזיכרון היא 0x26C וטווח הכתובות הוא 200 כתובות או 0x40. סוג זיכרון זה הוא Stack מכיוון שהמשתנה הוא לוקלי בתוך פונקציית Main.

Mat2	<array>	Memory : 0x26C
 [0]	<array>	Memory : 0x26C
 [1]	<array>	Memory : 0x280
 [2]	<array>	Memory : 0x294
 [3]	<array>	Memory : 0x2A8
 [4]	<array>	Memory : 0x2BC
 [5]	<array>	Memory : 0x2D0
 [6]	<array>	Memory : 0x2E4
 [7]	<array>	Memory : 0x2F8
 [8]	<array>	Memory : 0x30C
 [9]	<array>	Memory : 0x320
 [0]	15171	Memory : 0x320
 [1]	31089	Memory : 0x322
 [2]	-23927	Memory : 0x324
 [3]	-15876	Memory : 0x326
 [4]	-23963	Memory : 0x328
 [5]	28105	Memory : 0x32A
 [6]	-30253	Memory : 0x32C
 [7]	-23219	Memory : 0x32E
 [8]	-4619	Memory : 0x330
 [9]	8493	Memory : 0x332

3. מיקום המחסנית הנקבע ע"י המהדר הוא 0x3FF.

The screenshot displays two windows from a debugger:

- Stack 1:** A table showing memory locations and their data.
 

Location	Data
0x03FE	0x14
0x03FF	0xC0
- Disassembly:** A list of assembly instructions with their addresses.
 

Address	Instruction
0003DA	3F73 jmp 0x
0003DC	899F 26E1 6A16 sub.w 0x
0003E2	1C36 01DF rrax.a ??
0003E6	FEB4 51A4 and.w @R
0003EA	6F1D 4798 addc.w 0x
0003EE	D174 bis.b @C
0003F0	3F89 jmp 0x
0003F2	B7F5 7A32 bit.b @R
0003F6	01CF mova SF
0003F8	DAE7 3478 bis.b @R
0003FC	95A4 C014 cmp.w @R
000400	0000 ????
000402	0000 ????
000404	0000 ????
000406	0000 ????
000408	0000 ????
00040A	0000 ????
00040C	0000 ????
00040E	0000 ????
000410	0000 ????
000412	0000 ????
000414	0000 ????

4. תוכן ה-SP כאשר ה-PC מצביע על השורה הראשונה בפונקציית ComputeTrace הוא 0x266, כתובת הפקודה שקוראת לפונקציה לראשונה. זאת מכיוון שכאשר הקוד ירוץ על כל הפונקציה, הוא יטען את הכתובת ל-PC וכך התוכנה תחזור לעבוד מאיפה שהפונקציה נקראה.

Register		×
CPU Registers		▼
PC	=	0xC2B2
SP	=	0x0266
⊕ SR	=	0x0000
R4	=	0x5DC1
R5	=	0x2810
R6	=	0x3A30
R7	=	0x7D95
R8	=	0x1F47
R9	=	0x2AFB
R10	=	0x0001
R11	=	0x3774
R12	=	0x0334
R13	=	0x000A
R14	=	0x000A
R15	=	0x0000
CYCLECOUNTER	=	3046
CCTIMER1	=	3046
CCTIMER2	=	3046
CCSTEP	=	14

5. הכתובת של הפונקציה FillMatrix היא 0xC262 כמו שניתן לראות בחלון watch והdisassembly:

The screenshot shows a debugger interface with two windows. The 'Watch' window on the left displays a list of expressions and their values. The 'Disassembly' window on the right shows the assembly code for the 'FillMatrix' function, starting at address 0xC262. The code includes nested loops for 'i' and 'j', and operations like 'push.w', 'clr.w', and 'jmp'.

טווח הכתובות של הפונקציה היא 0xC2AA-0xC262 אורך הפונקציה היא 0x48 כתובות. הפונקציה מצויה בזיכרון ה-Flash.

6. בכדי להבין מהו זמן ריצת הפונקציה נציב breakpoint בפקודה הראשונה של הפונקציה. בנוסף לכך, נציב breakpoint נוסף בפקודה האחרונה של הפונקציה ונאמר כי ההפרש ביניהם הוא זמן ריצת הפונקציה.

7.  $CYCLECOUNTER@End - CYCLECOUNTER@Start = 3013 - 86 = 2927_{MCLK}$  mat2Trace הוא משתנה לוקאלי שממוקם בתוך פונקציית החישוב ולכן הסקופ של המשתנה הוא ה-main. בזמן scopen, מיקומו של המשתנה הוא במחסנית (המכילה משתנים לוקאליים) בהתאם לתצורת הזיכרון.

8. נשים Breakpoint על השורה הזאת בקוד, נריץ ב-debugger ונצפה בחלון disassembly מהן פקודות האסמבלי עבור שורה זאת:

The screenshot shows a debugger interface with two windows. The left window displays a C code snippet with a 'while' loop and a 'switch' statement. The right window shows the assembly code for the 'while' loop, starting at address 0xC054. The code includes instructions like 'mov.v', 'call', 'cmp.w', 'jge', 'mov.v', 'jmp', and 'clr.v'.

## חלק תיאורטי – עבור CCS.

2. כתובת מערך Mat2 הינה 0x04C6 ואורך המערך הוא 200 כתובות, 2 בתים לכל ערך במטריצה. סוג זיכרון זה הוא זיכרון דינאמי.

Name	Type	Value	Location
Mat2	int[10][10]	[[0,0,0,0,0...],[0,0,0,0,0...],[0,0,0,0,0...]]	0x04C6
> [0]	int[10]	[0,0,0,0,0...]	0x04C6
> [1]	int[10]	[0,0,0,0,0...]	0x04DA
> [2]	int[10]	[0,0,0,0,0...]	0x04EE
> [3]	int[10]	[0,0,0,0,0...]	0x0502
> [4]	int[10]	[0,0,0,0,0...]	0x0516
> [5]	int[10]	[0,0,0,0,0...]	0x052A
> [6]	int[10]	[0,0,0,0,0...]	0x053E
> [7]	int[10]	[0,0,0,0,0...]	0x0552
> [8]	int[10]	[0,0,0,0,0...]	0x0566
> [9]	int[10]	[0,0,0,0,0...]	0x057A
[0]	int	0	0x057A
[1]	int	0	0x057C
[2]	int	0	0x057E
[3]	int	0	0x0580
[4]	int	0	0x0582
[5]	int	0	0x0584
[6]	int	0	0x0586
[7]	int	0	0x0588
[8]	int	0	0x058A
[9]	int	0	0x058C

3. את תחילת המחסנית נמצא באמצעות הביטוי "stack\_end", המראה 0x400.

Expression	Type	Value	Address
> _STACK_END	void *	0x0400	

4. תוכן רגיסטר SP כאשר רגיסטר PC מצביע על הפקודה הראשונה לביצוע בפונקציית

ComputeTrace היא 0x0268.

The screenshot shows the CCS IDE interface. On the right, the 'Registers' window displays the SP register at address 0x0268. On the left, the 'main.c' file is open, showing the 'ComputeTrace' function. The function signature is 'int ComputeTrace(int Mat[M][M])'. The function body includes a loop that iterates over the matrix elements and a return statement. The 'ComputeTrace' function is highlighted with a red box.

5. באמצעות חלון disassembly ניתן לראות שכתובת הפונקציה FillMatrix בזיכרון היא 0x288, והיא נמשכת עד הכתובת 0xC2D4, כלומר 0x4C, או 76 כתובות. הפונקציה

שמורה ב-Flash.

```
98 void FillMatrix(int Mat[M][M]){
    FillMatrix():
c288: 120A          PUSH    R10
c28a: 1209          PUSH    R9
c28c: 1208          PUSH    R8
c28e: 4C08          MOV.W   R12,R8
100     for(i=0 ; i<M ; i++){
c290: 430A          CLR.W   R10
c292: 903A 000A     CMP.W   #0x000a,R10
c296: 341E          JGE     ($L23)
101     for(j=0 ; j<M ; j++){
    $C$L20:
c298: 4308          CLR.W   R10
c29a: 903A 000A     CMP.W   #0x000a,R10
c29e: 341E          JGE     ($L23)
102     }
```

6. את זמן הריצה של הפונקציה נמצא באמצעות שתי break points, בתחילת ובסוף הפונקציה וע"י שימוש בטיימר TA0 ואיפוסו לפני הקריאה לפונקציה. ראינו ברגיסטר המנייה את מספר מחזורי השעון (מסומן בצהוב).

Lab1CCS [Code Composer Studio - Device Debugging]  
TI MSP430 USB1/MSP430 [Suspended - SW Breakpoint]  
FillMatrix(int[10][10]) at main.c:115 0xC2E2  
main() at main.c:32 0xC024  
\_c\_int00\_noargs\_noexit() at boot\_special.c:103 0xC326 (the entry point was reached)

Name	Value	Description
Timer0_A3		
TA0IV	0x0000	Timer0_A3 Interrupt Vector Word [Memory Mapped]
TA0CTL	0x0220	Timer0_A3 Control [Memory Mapped]
TA0CCTL0	0x0000	Timer0_A3 Capture/Compare Control 0 [Memory Mapped]
TA0CCTL1	0x0000	Timer0_A3 Capture/Compare Control 1 [Memory Mapped]
TA0CCTL2	0x0000	Timer0_A3 Capture/Compare Control 2 [Memory Mapped]
TA0R	0x21EB	Timer0_A3 Counter Register [Memory Mapped]
TA0CCR0	0x0000	Timer0_A3 Capture/Compare 0 [Memory Mapped]
TA0CCR1	0x0000	Timer0_A3 Capture/Compare 1 [Memory Mapped]
TA0CCR2	0x0000	Timer0_A3 Capture/Compare 2 [Memory Mapped]
Timer1_A3		

main.c  
24 TA0CTL = MC\_0;  
25  
26 // Clear Timer\_A  
27 TA0CTL |= TACLR;  
28  
29 // Set Timer\_A source to MCLK, continuous mode, no division  
30 TA0CTL = TASSEL\_2 | MC\_2 | ID\_0;  
31  
32 FillMatrix(Mat1);  
33  
34 while(1) {  
35

7. mat2Trace הוא משתנה לוקאלי שממוקם בתוך פונקציית החינה main ולכן הסקופ של המשתנה הוא ה-main. בזמן הscope, מיקומו של המשתנה הוא במחסנית (המכילה משתנים לוקאליים) בהתאם לתצורת הזיכרון.

8. נשים breakpoint על שורת הקוד הנ"ל וניפתח את חלון disassembly:

```
42     maxTrace = mat1Trace > mat2Trace ? mat1Trace : mat2Trace;
c084: 990C          CMP.W   R9,R12
c086: 3401          JGE     ($L7)
c088: 490C          MOV.W   R9,R12
    $C$L7:
c08a: 4C82 0202     MOV.W   R12,&maxTrace
44     break;
c08e: 3FE7          JMP     ($L5)
```