בס"ד

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

קובץ הכנה ניסוי מעבדה מס' 5

Tutorial 5.1 – Basic Timer1

מעבדת מיקרומחשבים – המחלקה להנדסת חשמל ומחשבים

מס' קורס - 361.1.3353

כתיבה ועריכה: חנן ריבוא

מהדורה 1 – שנה"ל תשע"ו

A. הקדמה:

a. הצורך **בתזמון מדויק** של מערכת זוהי דרישה בסיסית חשובה מאוד לפעולה תקינה של אפליקציית זמן-אמת. רכיב Timer מהווה למעשה רכיב פריפריאלי ל- CPU לצורך הגדרות תזמון מסוגים שונים הנקבעות בתוכנה ע"י כתיבה לרגיסטרי הבקרה השולטים עליהם. **בצורתו הבסיסית טיימר הוא מונה (מכיל רגיסטר המקודם ב-1 בכל מחזור שעון) את מחזורי השעון המזין אותו ויוצר פסיקה בהגיעו ל- overflow**. תדר השעון המקסימאלי אליו ניתן להגיע משתנה בין סוגי הבקרים.

b. סוגים שונים של תזמון:

1. הגדרת תזמון כדי להכתיב את המקור הקובע את תדר העבודה של ה- CPU (שעון MCLK) או של הרכיבים הפריפריאליים בבקר (SMCLK, ACLK).
2. יצירת פסיקות באופן מחזורי לפי בחירתנו.
3. עירור מחזורי ממצב שינה.
4. ספירת עליות / ירידות באות דיגיטלי.
5. יצירת השהייה כך שבמקביל ה- CPU יבצע פעולות או יכנס למצב שינה.
6. יצירת אות ריבועי (אות מחזורי עם 2 ערכים של '1' ו- '0' בלבד) בתדר וב- Duty Cycle הניתנים לתכנות.

c. סוגי Timers (ראה דיאגרמה עמ' 4):

1. **Basic Timer (ספר המעבדה עמודים 433 – 425):**

טיימר זה מאפשר פעולה בסיסית של יצירת פסיקה באופן מחזורי בתדר הניתן לתכנות.

2. **Real-Time Clock (RTC) (ספר המעבדה עמודים 448 – 435):**

טיימר זה משמש בעיקר כשעון זמן אמת המונה **שניות, דקות, שעות, ימים, לילות, חודשים ושנים** (מאפשר בנוסף יצירת פסיקה באופן מחזורי בכל שינוי של אחד מהם).

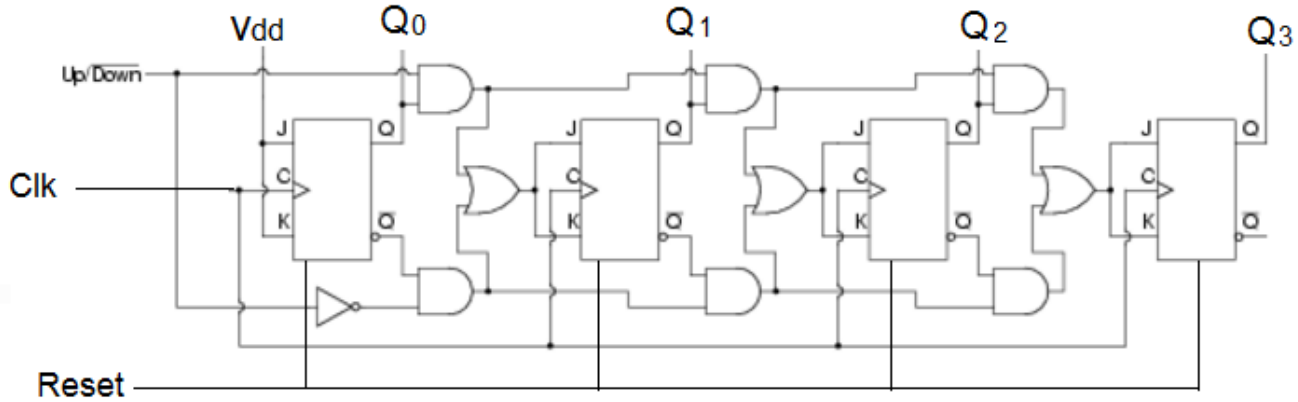
3. **Watchdog timer (ספר המעבדה עמודים 424 – 415):**

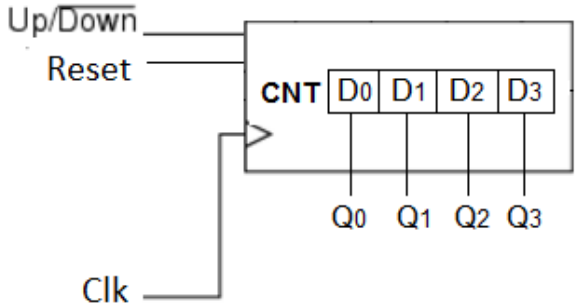
טיימר זה מאפשר יצירת פסיקה באופן מחזורי, **אולם שימושו העיקרי** הוא לצורך גילוי ויציאה מתקלות חומרה הקורות במהלך ביצוע התוכנית. הבקר באופן קבוע ובתזמון מסוים מאתחל טיימר זה כדי **שלא יעבור את ערכו המקסימאלי.** במקרה של תקלת חומרה ה- CPU לא מצליח לאתחל את הטיימר כך הוא עובר את ערכו המקסימאלי (נוצר overflow) המייצר אות המשמש טריגר לפעולת תיקון המעבירה את ערך ה- PC למצב בטוח (בד"כ ביצוע RESET). שימוש במצב זה מצוי במערכות שתיקון השגיאות והתגברות עליהן מצריך תגובה מהירה בזמן-אמת וללא התערבות חיצונית

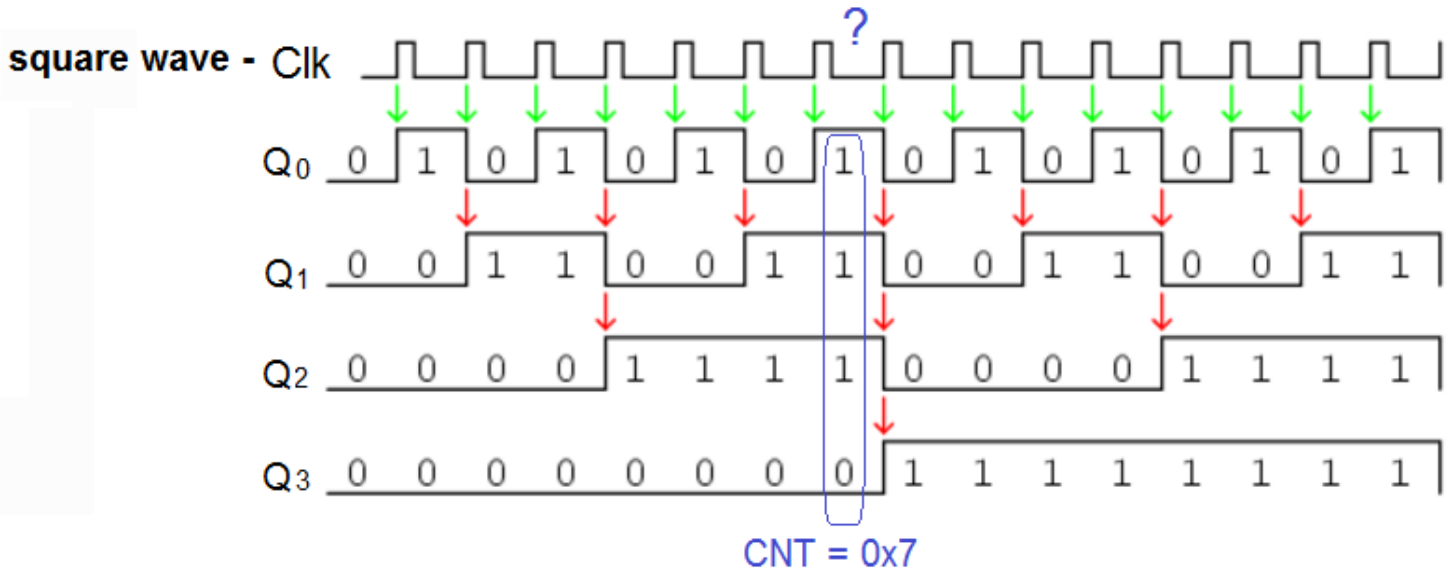
4. **Timer_A , Timer_B (ספר המעבדה עמודים 498 - 473):**

טיימרים אלו מורכבים יותר ומשמשים לצורך של - יצירת פסיקות באופן מחזורי לפי בחירתנו, עירור מחזורי ממצב שינה, ספירת עליות / ירידות באות דיגיטלי, יצירת אות ריבועי בתדר וב- Duty Cycle הניתנים לתכנות.

d. הסבר פעולת Timer בסיסי – 4-bit Up Timer:



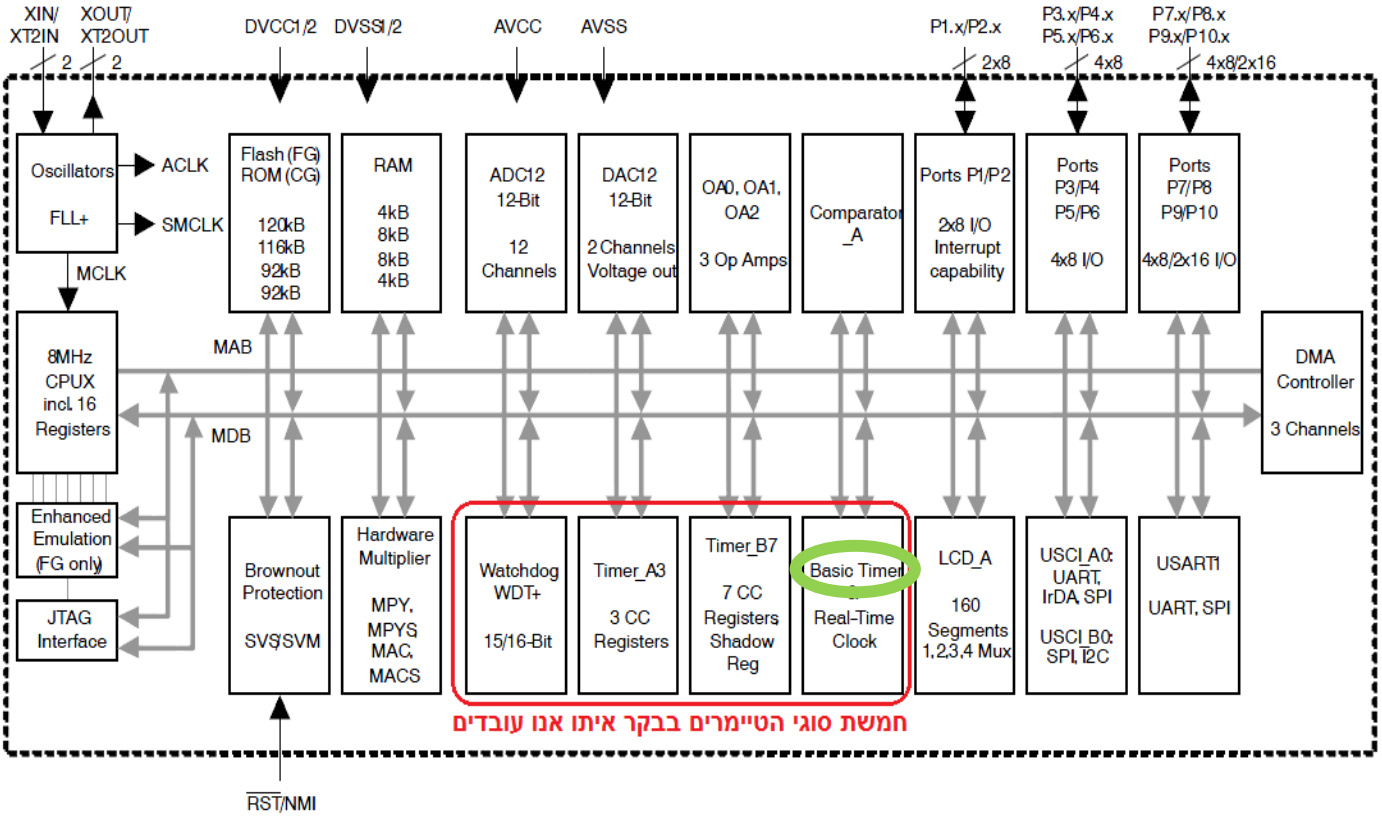




שימו לב:

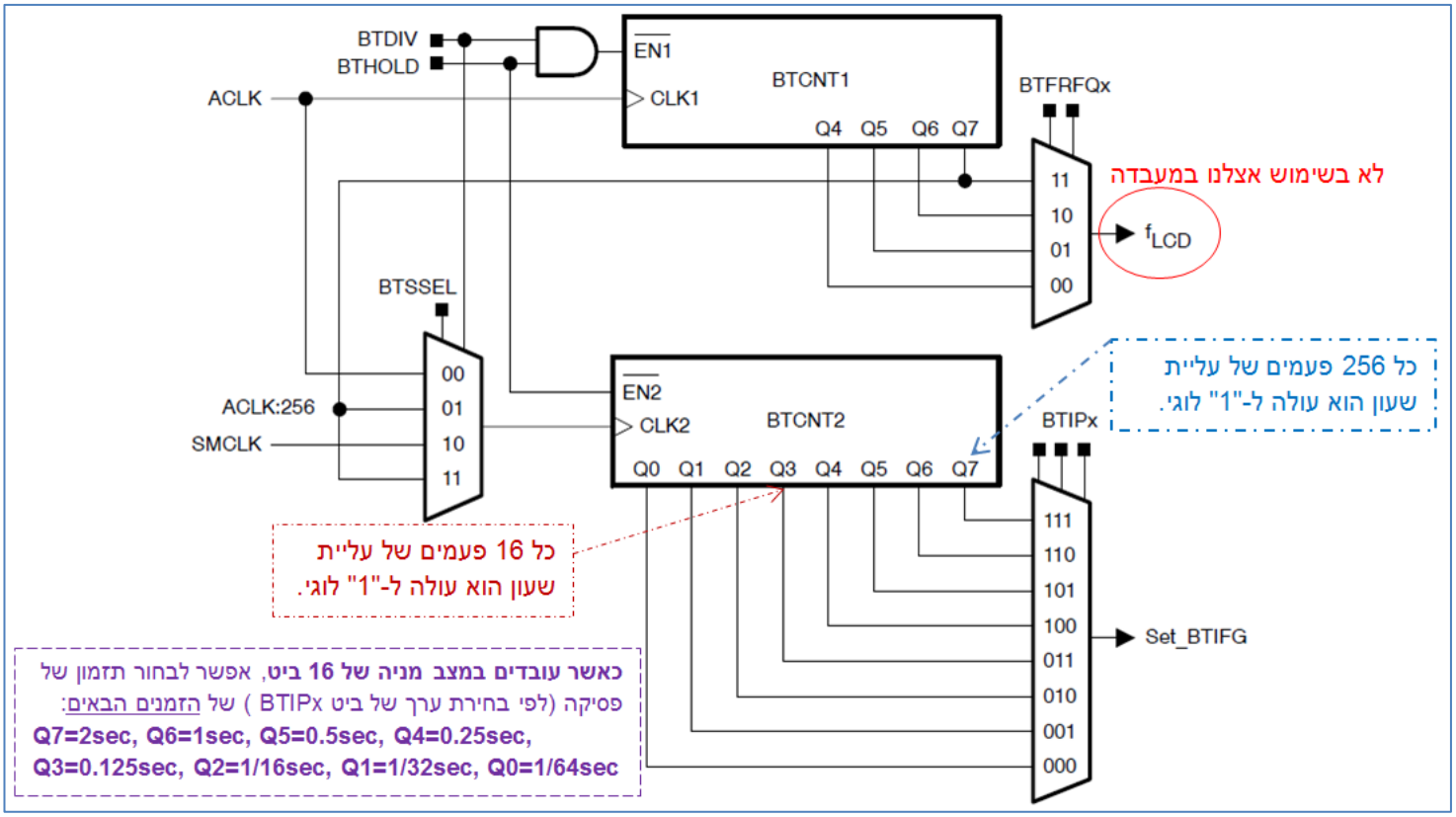
עליית רמה (rising edge) של אות Q_i נובע עקב over-flow של סיביות $D_{i-1},\ldots,D_0$ ברגיסטר CNT

לכן מתקיים $T_{Q_i} = \mathrm{T}_{Clk} \cdot 2^{i+1}$ או במילים אחרות $f_{Q_i} = \frac{f_{Clk}}{2^{i+1}}$



B. טיימר בסיסי – Basic Timer1:

a. טיימר זה הוא רכיב פריפריאלי המשמש טיימר בסיסי. Basic Timer מורכב מ-2 טיימרים נפרדים באורך 8bit המחוברים בטור, כך שניתן להשתמש ב- Basic Timer כטיימר **8bit** וכטיימר **16bit**. שעון המזין את הטיימר ניתן לבחירה (ACLK , SMCLK). ניתן לבחור ערך כך כשהטיימר יגיע אליו תיווצר בקשת פסיקה. בתחילת השימוש יש לאפס את ערך הטיימר (מכיל "זבל"). שליטה על הטיימר מתבצע ע"י כתיבה לרגיסטר BTCTL באורך 8bit (שימוש ב- byte instruction).



b. רגיסטר BTCNT1 מוזן **רק** ממוצא שעון **ACLK בתדר 32768Hz** (בכל מחזור ACLK ערך BTCNT1 מתקדם ב-1) ואפשר לעוצרו ע"י כתיבת '1' לביטים **BTHOLD** ו- **BTDIV**.

c. רגיסטר **BTCNT2** גם הוא באורך 8bit וניתן להזנה מ-3 מקורות (ACLK, ACLK/256, SMCLK). המקור שלו נקבע ע"י ביטים **BTSSEL** ו- **BTDIV** (כאשר ביט **BTDIV='1'** הטיימר באורך 16bit ו-2 הרגיסטרים משורשרים, מקור BTCNT1 הוא ACLK ומקור BTCNT2 הוא ACLK/256).

ניתן לעוצרו ע"י ביט **HOLD** לצורך חסכון בצריכת אנרגיה. לרגיסטר ישנם 8 מוצאים Q0 – Q7 לצורך בקשת פסיקה במרווחי זמן מסוימים (תלוי במקור ההזנה של הטיימר).

מרווח הזמן לבקשת פסיקה ממוצא Q_i הוא $Q_i = CLK2_{cycle} \cdot 2^{i+1}$ וכאשר המקור של **BTCNT2** הוא ACLK/256 מרווחי הזמן (נקבעים ע"י 3 הביטים **BTIPx**) האפשריים הם:

Q7=2sec, Q6=1sec, Q5=0.5sec, Q3=0.125sec, Q2=1/16sec, Q1=1/32sec, Q0=1/64sec Q4=0.25sec,

הסיבה שתדר ACLK ערכו 32768Hz כי לפי הנוסחה של Q_i מתקבלים מרווחי זמן נוחים מאוד.

d. פסיקות Basic Timer:

1. וקטור הפסיקה נקרא BASICTIMER_VECTOR
2. דגל הפסיקה BTIFG (ממוקם ברגיסטר IFG2) עולה ל- '1' כאשר ערך הטיימר מגיע למרווח הזמן המוגדר בביטים **BTIPx** ויורד ל-'0' באופן אוטומטי בכניסה ל- ISR (ניתן לאיפוס בתוכנה).
3. בקשת פסיקה דורשת אפשור מקומי (BTIE = '1') ברגיסטר IE2 וגלובלי (GIE = '1').

C. פירוט רגיסטרים – Basic Timer:

Basic Timer1 Registers

Register	Short Form	Register Type	Address	Initial State
Basic Timer1 Control	BTCTL	Read/write	040h	Unchanged
Basic Timer1 Counter 1	BTCNT1	Read/write	046h	Unchanged
Basic Timer1 Counter 2	BTCNT2	Read/write	047h	Unchanged
SFR interrupt enable register 2	IE2	Read/write	001h	Reset with PUC
SFR interrupt flag register 2	IFG2	Read/write	003h	Reset with PUC

Note: The Basic Timer1 registers should be configured at power-up. There is no initial state for BTCTL, BTCNT1, or BTCNT2.

a) **רגיסטר בקרה:** ישנו רגיסטר בקרה אחד, הנקרא BTCTL השולט על המודול.

BTCTL (Control Register):

7	6	5	4	3	2	1	0
BTSSEL	BTHOLD	BTDIV	BTFRFQx		BTIPx		
rw	rw	rw	rw	rw	rw	rw	rw

BTSSEL	Bit 7	BTCNT2 clock select. This bit, together with the BTDIV bit, selects the clock source for BTCNT2. See the description for BTDIV.
BTHOLD	Bit 6	Basic Timer1 hold 0 BTCNT1 and BTCNT2 are operational 1 BTCNT1 is held if BTDIV=1 BTCNT2 is held
BTDIV	Bit 5	Basic Timer1 clock divide. This bit together with the BTSSEL bit, selects the clock source for BTCNT2.

BTSSEL	BTDIV	BTCNT2 Clock Source
0	0	ACLK
0	1	ACLK/256
1	0	SMCLK
1	1	ACLK/256

BTFRFQx	Bits 4-3	f_{LCD} frequency. These bits control the LCD update frequency. 00 $f_{ACLK}/32$ 01 $f_{ACLK}/64$ 10 $f_{ACLK}/128$ 11 $f_{ACLK}/256$
BTIPx	Bits 2-0	Basic Timer1 interrupt interval 000 $f_{CLK2}/2$ 001 $f_{CLK2}/4$ 010 $f_{CLK2}/8$ 011 $f_{CLK2}/16$ 100 $f_{CLK2}/32$ 101 $f_{CLK2}/64$ 110 $f_{CLK2}/128$ 111 $f_{CLK2}/256$

irrelevant for us

b) **רגיסטר מידע – ערך המנייה:**

BTCNT1, Basic Timer Counter 1

7	6	5	4	3	2	1	0
BTCNT1x							
rw	rw	rw	rw	rw	rw	rw	rw

BTCNT1x	Bits 7-0	BTCNT1 register. The BTCNT1 register is the count of BTCNT1.

BTCNT2, Basic Timer Counter 2

7	6	5	4	3	2	1	0
BTCNT2x							
rw	rw	rw	rw	rw	rw	rw	rw

BTCNT2x	Bits 7-0	BTCNT2 register. The BTCNT2 register is the count of BTCNT2.

c) רגיסטר אפשור פסיקה:

IE2, Interrupt Enable

7	6	5	4	3	2	1	0
BTIE							
rw-0							

BTIE Bit 7 Basic Timer1 interrupt enable. This bit enables the BTIFG interrupt. Because other bits in IE2 may be used for other modules, it is recommended to set or clear this bit using `BIS.B` or `BIC.B` instructions, rather than `MOV.B` or `CLR.B` instructions.

0 Interrupt not enabled
1 Interrupt enabled

d) רגיסטר דגלי פסיקה:

IFG2, Interrupt Flag

7	6	5	4	3	2	1	0
BTIFG							
rw-0							

BTIFG Bit 7 Basic Timer1 interrupt flag. Because other bits in IFG2 may be used for other modules, it is recommended to clear BTIFG automatically by servicing the interrupt, or by using `BIS.B` or `BIC.B` instructions, rather than `MOV.B` or `CLR.B` instructions.

0 No interrupt pending
1 Interrupt pending

e. צורת כתיבה לרגיסטרים של הטיימר:

```
mov.b   #BTDIV+BT_fCLK2_DIV16,&BTCTL   ; ACLK/(256*16)
```

mov.b #0x20+0x03 , &BTCTL

BTCTL

7	6	5	4	3	2	1	0
BTSSEL 0	BTHOLD 0	BTDIV 1	BTFRFQx 00		BTIPx 011		
rw	rw	rw	rw	rw	rw	rw	rw

#include <msp430xG46x.h>

```
/* Interrupt interval time fINT coded with Bits 0-2 in BTCTL */
#define BT_fCLK2_DIV2        (0x00)                  /* fINT = fCLK2:2 (default) */
#define BT_fCLK2_DIV4        (BTIP0)                 /* fINT = fCLK2:4 */
#define BT_fCLK2_DIV8        (BTIP1)                 /* fINT = fCLK2:8 */
#define BT_fCLK2_DIV16       (BTIP1+BTIP0)           /* fINT = fCLK2:16 */
#define BT_fCLK2_DIV32       (BTIP2)                 /* fINT = fCLK2:32 */
#define BT_fCLK2_DIV64       (BTIP2+BTIP0)           /* fINT = fCLK2:64 */
#define BT_fCLK2_DIV128      (BTIP2+BTIP1)           /* fINT = fCLK2:128 */
#define BT_fCLK2_DIV256      (BTIP2+BTIP1+BTIP0)     /* fINT = fCLK2:256 */
```

D. דוגמה:

בדוגמא זו לד P9.1 מהבהב ע"י כיבוי והדלקה לסירוגין במרווחי זמן של 0.125sec בפעולת toggle (פעולת not ע"י xor עם '1') בתוך ISR של Basic Timer.

```
#include  <msp430xG46x.h>
;-------------------------------------------------------------------------------
            RSEG    CSTACK                  ; Define stack segment
;-------------------------------------------------------------------------------
            RSEG    CODE                    ; Assemble to Flash memory
;-------------------------------------------------------------------------------
RESET       mov.w   #SFE(CSTACK),SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL  ; Stop WDT
SetupFLL    bis.b   #XCAP14PF,&FLL_CTL0     ; Configure load caps

            bis.b   #BIT1,&P9DIR                    ; Set P9.1 as Output
SetupBT     mov.b   #BTDIV+BT_fCLK2_DIV16,&BTCTL    ; ACLK/(256*16)
            clr.b   &BTCNT1
            clr.b   &BTCNT2
            bis.b   #BTIE,&IE2                      ; Enable BT interrupt
Mainloop    bis.w   #LPM3+GIE,SR                    ; Enter LPM3, enable interrupts
            nop                                     ; Required for only for Debugger
;-------------------------------------------------------------------------------
Basic_Timer_ISR;  // Basic Timer Interrupt Service Routine
;-------------------------------------------------------------------------------
            xor.b   #BIT1,&P9OUT            ; Toggle P9.1 (LED)
            reti
;-------------------------------------------------------------------------------
            COMMON  INTVEC                  ; Interrupt Vectors
;-------------------------------------------------------------------------------
            ORG     RESET_VECTOR        כתובת ווקטור הפסיקה של RESET, תווית זו מהווה את כתובתו
            DW      RESET               תוכן ווקטור הפסיקה, תווית זו מהווה את כתובת תחילת התוכנית
            ORG     BASICTIMER_VECTOR   כתובת ווקטור הפסיקה של Basic Timer, תווית זו מהווה את כתובתו
            DW      Basic_Timer_ISR     תוכן ווקטור הפסיקה, תווית זו מהווה את כתובת ISR של פסיקת של Basic Timer
            END
```

E. מניית תדר שעון חיצוני – מימוש Counter (שיטה אחת מתוך שלוש, השאר נלמד בהמשך):

תזכורת: תדר = מספר מחזורים בשנייה

שלבי עבודה:

1. נחבר את אות השעון החיצוני לאחת מרגלי הבקר PORT1/PORT2 במצב של פסיקת digital input. נבחר למשל לחבר לרגל P2.2 במצב של פסיקת input. נגדיר את הטיימר לגודל 16bit במצב פסיקה במרווחי זמן של 1sec. נאפס את ערך הטיימר (רגיסטרים BTCNT2, BTCNT1) ונבצע עצירה לטיימר (BTHOLD=0).

2. ב- rising edge התחלתי של השעון, תתבצע בקשת פסיקה של P2.2 ובתוך רוטינת השירות שלה נפעיל את הטיימר (BTHOLD=1). בכל כניסה ל- ISR של P2.2 (עקב עליית שעון) נבצע בתוכה counter++ , מנייה ע"י משתנה counter.

3. לאחר שנייה תתקבל בקשת פסיקה מהטיימר. בתוך ה- ISR של הטיימר נקרא את ערך משתנה counter המהווה למעשה את תדר אות השעון החיצוני המחובר לרגל P2.2.