בס"ד

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

קובץ הכנה ניסוי מעבדה מס' 6

Tutorial 6 – Advanced Timers

Timer_A , Timer_B

מעבדת מיקרומחשבים – המחלקה להנדסת חשמל ומחשבים

מס' קורס - 361.1.3353

כתיבה ועריכה: חנן ריבוא

מהדורה 1 – שנה"ל תשע"ו

קריאה משלימה ניתן למצוא בקובץ המעבדה MSP430x4xx user guide עמודים 498 - 473

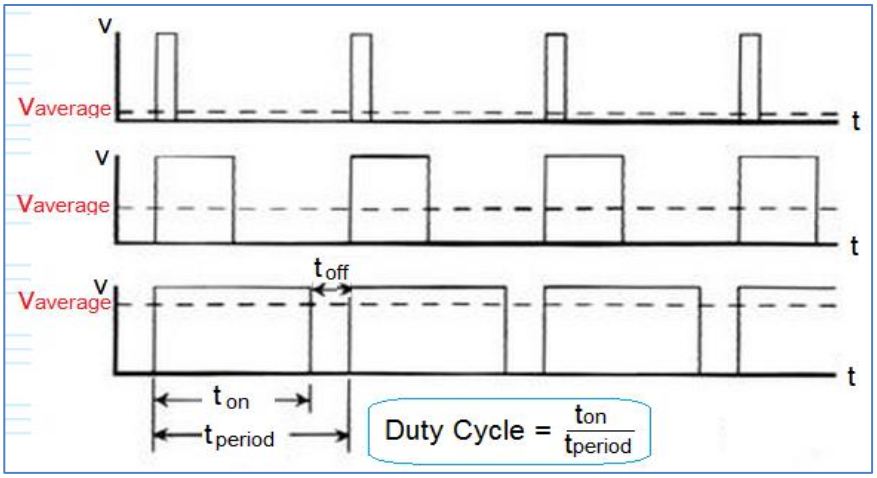
1) הקדמה:

Timer_B הינו מודול חומרה הפועל במקביל ל- CPU (כמו כל רכיב פריפריאלי) ומהווה טיימר באורך 16bit הניתן לתכנות (ישנה אפשרות לקנפג אותו לאורך של 8,10,12,16 bits ע"י 2 ביטים **CNTLx**, במקרים אלו ערכי המנייה המקסימליים הם **0x00FF, 0x03FF, 0x0FFF, 0xFFFF** בהתאמה) המונה מחזורי שעון המזין אותו (אות ריבועי מחזורי מ-3 מקורות הניתן לחלוקה ב- 2/4/8 ע"י 2 ביטים **IDx**) המנייה למעשה **קידום או חיסור ב-1** של רגיסטר **TBR** (באורך 16bit) **בכל עליית שעון**.

ניתן לאפס את תוכן רגיסטר **TBR** ע"י העלאת ביט **TBCLR** ל-'1' (גורם לאיפוס הביטים **IDx** ו- **MCx**).

השעון המזין (בחירה ע"י כתיבה ל-2 ביטים **TBSSELx**) יכול להיות **פנימי** (**SMCLK** , **ACLK** במקרה זה השימוש נקרא timer) או **חיצוני** (**TBCLK**, במקרה זה השימוש נקרא counter) דרך רגל P1.4 (כמובן לפי קינפוג מתאים – ראה datasheet). בנוסף לתכונות הבסיסיות של טיימר, מניית מחזורי שעון ובקשת פסיקות במרווחי זמן שווים (כפולות של מחזורי השעון). במקביל לטיימר מחוברים 7 רגיסטרים הפועלים בנפרד ומקנים לטיימר יכולות נוספות:

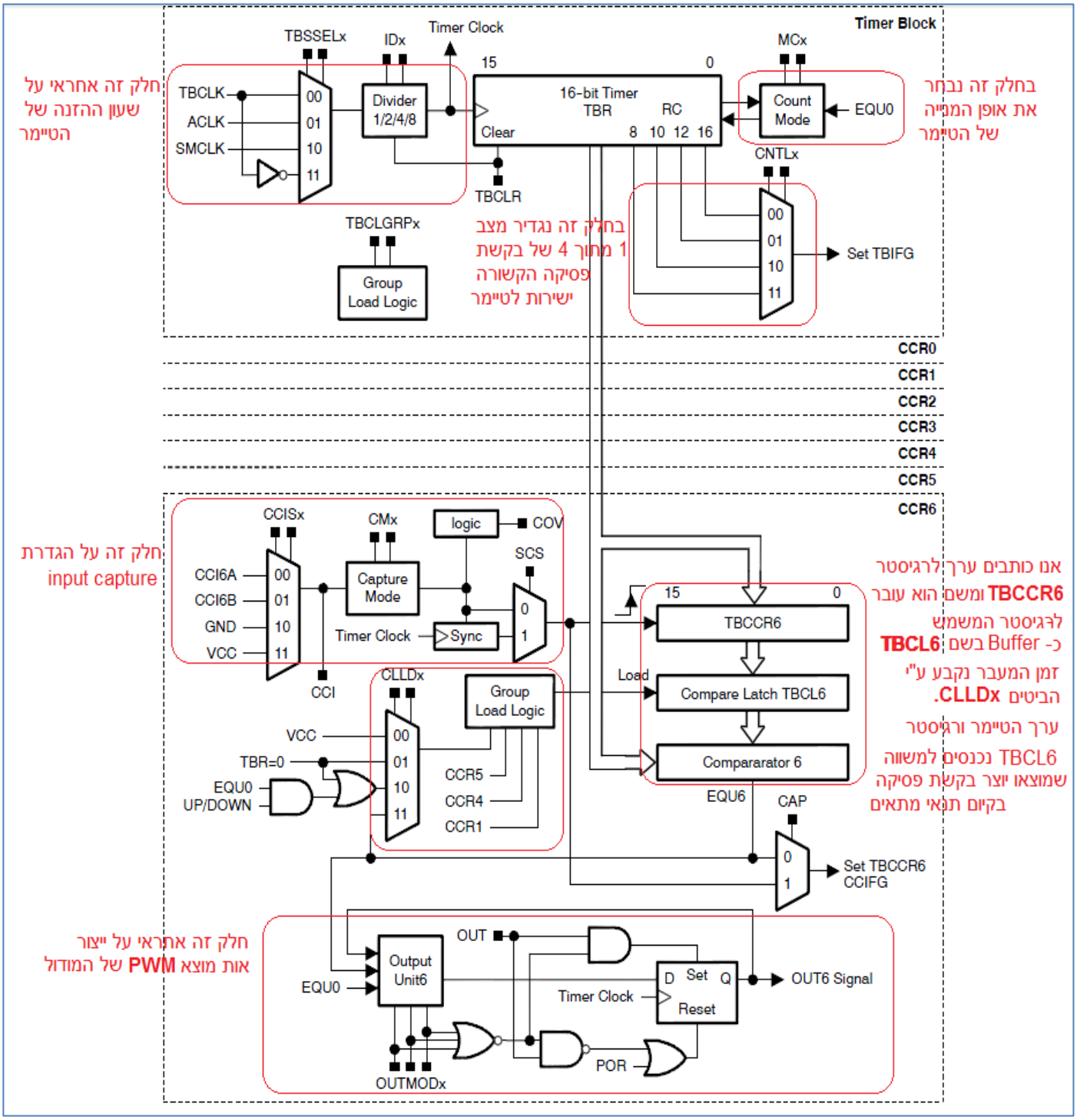
- לכידת ערך הטיימר באחד מ-7 הרגיסטרים (לצורך קריאה בתוכנה) בהינתן טריגר של אות חיצוני ברגל הבקר המתאימה לרגיסטר שבחרנו (3 אפשרויות - עלייה / ירידה / עלייה וירידה) ובקשת פסיקה. תכונה זאת נקראת **capture**
- בקשת פסיקה כאשר הטיימר יגיע לערך שכתבנו באחד מ-7 הרגיסטרים, תכונה זאת נקראת **compare.**
- לכל אחד מ-7 הרגיסטרים מחוברת חומרה המאפשרת ליצור פסיקות במרווחי זמן נוספים ויצירת אות PWM במוצא רגלי הבקר (אות ריבועי שניתן לתכנת את התדר שלו ואת חלק המחזור שהוא ב- '1' לוגי).



PWM signal

- בקשות פסיקה נוצרות ע"י תנאי Overflow של הטיימר או ע"י אחד מ-7 הרגיסטרים **capture/compare**

2) דיאגרמת בלוקים – מודול TIMER B:



3) פירוט רגיסטרים של מודול Timer_B:

a) רגיסטרי בקרה: ישנו רגיסטר בקרה אחד **השולט על ליבת המודול** Timer_B הנקרא **TBCTL** (מכיל ביט אפשור וביט דגל פסיקת ליבת הטיימר). ישנו רגיסטר בקרה אחד לכל מעטפת (ישנן 7 מעטפות) הנקרא **TBCCTLx** (כאשר x = 0,…,6 , מכיל ביט אפשור וביט דגל פסיקת מעטפת). סה"כ ישנם 8 רגיסטרי בקרה.

TBCTL (Control Register):

15	14	13	12	11	10	9	8
Unused	TBCLGRPx		CNTLx		Unused	TBSSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TBCLR	TBIE	TBIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Unused	Bit 15	Unused
TBCLGRP	Bit 14-13	TBCLx group 00 Each TBCLx latch loads independently 01 TBCL1+TBCL2 (TBCCR1 CLLDx bits control the update) TBCL3+TBCL4 (TBCCR3 CLLDx bits control the update) TBCL5+TBCL6 (TBCCR5 CLLDx bits control the update) TBCL0 independent 10 TBCL1+TBCL2+TBCL3 (TBCCR1 CLLDx bits control the update) TBCL4+TBCL5+TBCL6 (TBCCR4 CLLDx bits control the update) TBCL0 independent 11 TBCL0+TBCL1+TBCL2+TBCL3+TBCL4+TBCL5+TBCL6 (TBCCR1 CLLDx bits control the update)
CNTLx	Bits 12-11	Counter length 00 16-bit, $TBR_{(max)}$ = 0FFFFh 01 12-bit, $TBR_{(max)}$ = 0FFFh 10 10-bit, $TBR_{(max)}$ = 03FFh 11 8-bit, $TBR_{(max)}$ = 0FFh
Unused	Bit 10	Unused
TBSSELx	Bits 9-8	Timer_B clock source select 00 TBCLK 01 ACLK 10 SMCLK 11 Inverted TBCLK
IDx	Bits 7-6	Input divider. These bits select the divider for the input clock. 00 /1 01 /2 10 /4 11 /8
MCx	Bits 5-4	Mode control. Setting MCx = 00h when Timer_B is not in use conserves power. 00 Stop mode: the timer is halted 01 Up mode: the timer counts up to TBCL0 10 Continuous mode: the timer counts up to the value set by TBCNTLx 11 Up/down mode: the timer counts up to TBCL0 and down to 0000h
Unused	Bit 3	Unused
TBCLR	Bit 2	Timer_B clear. Setting this bit resets TBR, the clock divider, and the count direction. The TBCLR bit is automatically reset and is always read as zero.
TBIE	Bit 1	Timer_B interrupt enable. This bit enables the TBIFG interrupt request. 0 Interrupt disabled 1 Interrupt enabled
TBIFG	Bit 0	Timer_B interrupt flag. 0 No interrupt pending 1 Interrupt pending

TBCCTLx, Capture/Compare Control Register

15	14	13	12	11	10	9	8
CMx		CCISx		SCS	CLLDx		CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

CMx	Bit 15-14	Capture mode 00 No capture 01 Capture on rising edge 10 Capture on falling edge 11 Capture on both rising and falling edges
CCISx	Bit 13-12	Capture/compare input select. These bits select the TBCCRx input signal. See the device-specific data sheet for specific signal connections. 00 CCIxA 01 CCIxB 10 GND 11 V_{CC}
SCS	Bit 11	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0 Asynchronous capture 1 Synchronous capture
CLLDx	Bit 10-9	Compare latch load. These bits select the compare latch load event. 00 TBCLx loads on write to TBCCRx 01 TBCLx loads when TBR *counts* to 0 10 TBCLx loads when TBR *counts* to 0 (up or continuous mode) TBCLx loads when TBR *counts* to TBCL0 or to 0 (up/down mode) 11 TBCLx loads when TBR *counts* to TBCLx
CAP	Bit 8	Capture mode 0 Compare mode 1 Capture mode
OUTMODx	Bits 7-5	Output mode. Modes 2, 3, 6, and 7 are not useful for TBCL0, because EQUx = EQU0. 000 OUT bit value 001 Set 010 Toggle/reset 011 Set/reset 100 Toggle 101 Reset 110 Toggle/set 111 Reset/set
CCIE	Bit 4	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0 Interrupt disabled 1 Interrupt enabled
CCI	Bit 3	Capture/compare input. The selected input signal can be read by this bit.
OUT	Bit 2	Output. For output mode 0, this bit directly controls the state of the output. 0 Output low 1 Output high
COV	Bit 1	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0 No capture overflow occurred 1 Capture overflow occurred
CCIFG	Bit 0	Capture/compare interrupt flag 0 No interrupt pending 1 Interrupt pending

b) <u>רגיסטרי מידע</u>: ישנו רגיסטר אחד הנקרא **TBR** המכיל את ערך הטיימר של ליבת המודול.
ישנו רגיסטר נוסף, אחד לכל מעטפת (ישנן 7 מעטפות) הנקרא **TBCCRx** (כאשר x = 0,…,6). רגיסטר זה מכיל את הערך המידע עבור אופני פעולה Capture/Compare. סה"כ ישנם 8 רגיסטרי מידע.

TBR (Core Timer Register):

15	14	13	12	11	10	9	8
TBRx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
TBRx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

TBRx	Bits 15-0	Timer_B register. The TBR register is the count of Timer_B.

TBCCRx (Capture/Compare Register x):

15	14	13	12	11	10	9	8
TBCCRx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
TBCCRx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

TBCCRx	Bits 15-0	Timer_B capture/compare register Compare mode: Compare data is written to each TBCCRx and automatically transferred to TBCLx. TBCLx holds the data for the comparison to the timer value in the Timer_B Register, TBR. Capture mode: The Timer_B Register, TBR, is copied into the TBCCRx register when a capture is performed.

c) <u>רגיסטר טבלת פסיקות – הגעה מהירה לקטע קוד המטפל במקור פסיקה ספציפי לפי סדר העדיפות</u>:

TBIV (Interrupt Vector Register):

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

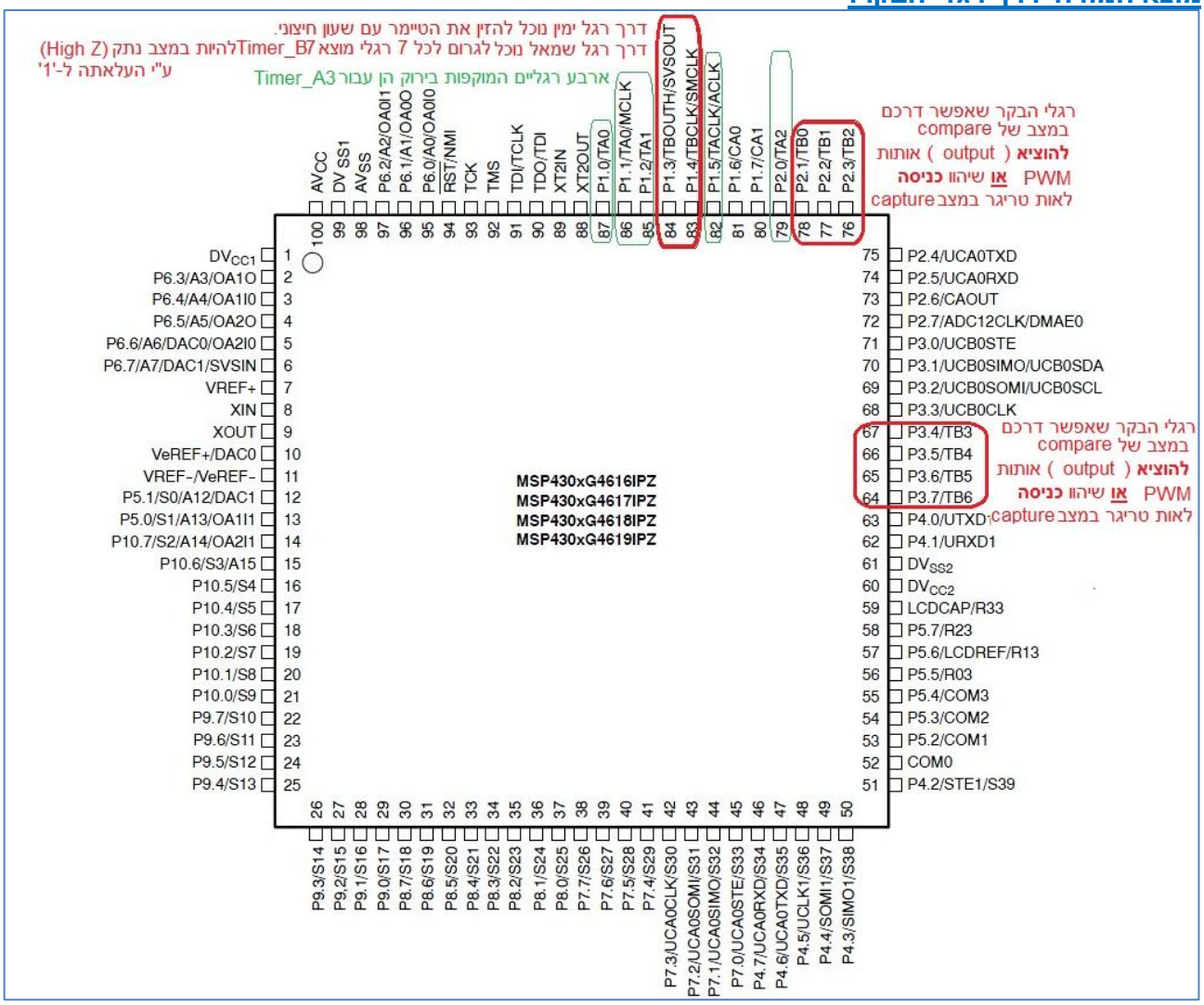
7	6	5	4	3	2	1	0
0	0	0	0	TBIVx			0
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

TBIVx	Bits 15-0	Timer_B interrupt vector value

TBIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	–	
02h	Capture/compare 1	TBCCR1 CCIFG	Highest
04h	Capture/compare 2	TBCCR2 CCIFG	
06h	Capture/compare 3†	TBCCR3 CCIFG	
08h	Capture/compare 4†	TBCCR4 CCIFG	
0Ah	Capture/compare 5†	TBCCR5 CCIFG	
0Ch	Capture/compare 6†	TBCCR6 CCIFG	
0Eh	Timer overflow	TBIFG	Lowest

MSP430x4xx devices only

4) מוצא המודול דרך רגלי הבקר:



כדי לקנפג את רגלי מוצא הבקר לשמש בתור כניסה עבור מצב input capture (מוקף בכחול) או לשמש בתור מוצא אות PWM עבור מצב output compare (מוקף באדום) נפנה לקובץ MSP430xG461x data sheet כמו שעשינו בתרגולי הכנה הקודמים. בעמוד 67,70

PIN NAME (P2.X)	X	FUNCTION	CONTROL BITS / SIGNALS	
			P2DIR.x	P2SEL.x
P2.1/TB0	1	P2.1 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI0A and Timer_B7.CCI0B	0	1
		Timer_B7.TB0 (see Note 1)	1	1
P2.2/TB1	2	P2.2 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI1A and Timer_B7.CCI1B	0	1
		Timer_B7.TB1 (see Note 1)	1	1
P2.3/TB2	3	P2.3 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI2A and Timer_B7.CCI2B	0	1
		Timer_B7.TB2 (see Note 1)	1	1

PIN NAME (P3.X)	X	FUNCTION	CONTROL BITS / SIGNALS	
			P3DIR.x	P3SEL.x
P3.4/TB3	4	P3.4 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI3A and Timer_B7.CCI3B	0	1
		Timer_B7.TB3 (see Note 1)	1	1
P3.5/TB4	5	P3.5 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI4A and Timer_B7.CCI4B	0	1
		Timer_B7.TB4 (see Note 1)	1	1
P3.6/TB5	6	P3.6 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI5A and Timer_B7.CCI5B	0	1
		Timer_B7.TB5 (see Note 1)	1	1
P3.7/TB6	7	P3.7 (I/O)	I: 0; O: 1	0
		Timer_B7.CCI6A and Timer_B7.CCI6B	0	1
		Timer_B7.TB6 (see Note 1)	1	1

NOTE 1: Setting TBOUTH causes all Timer_B outputs to be set to high impedance.

5) קינפוג מודול Timer_B:

- קינפוג הטיימר נבצע לפני הפעלתו. כאשר נרצה לשנות את קינפוגו ההתחלתי (חוץ מהביטים **TBCLR** **TBIE, TBIFG**) מומלץ מאוד לעצור את הטיימר (**MCx = 0x00**).
- כאשר מחברים שעון חיצוני לטיימר דרך רגל P1.4 יש לדאוג לסנכרונו עם שעון ה- CPU (MCLK) אחרת בכל קריאה מרגיסטר TBR נצטרך לעצור את הטיימר (ישנה שיטה שלא מצריכה לעצור את הטיימר הנקראת majority vote האומרת לקרוא את ערך TBR יותר מפעמיים בתוך זמן מחזור וערך הקריאה הנבחר יהיה לפי הרוב).
- רגיסטר TBR ניתן לקריאה וכתיבה. כאשר כותבים אליו ההשפעה קוראת מיד.

6) אופניי מנייה של Timer_B (Counting Mode):

הטיימר מתחיל לעבוד כאשר ערך **MCx** > 0 ושעון ההזנה של הטיימר פעיל.

עצירת הטיימר מתאפשרת ע"י איפוס 2 ביטי **MCx** או ע"י איפוס רגיסטר **TBCCR0** (במצב זה הפעלה מחדש של הטיימר תתבצע ע"י כתיבת מס' שונה מ-0 לרגיסטר **TBCCR0** והטיימר יתחיל במנייה מעלה).

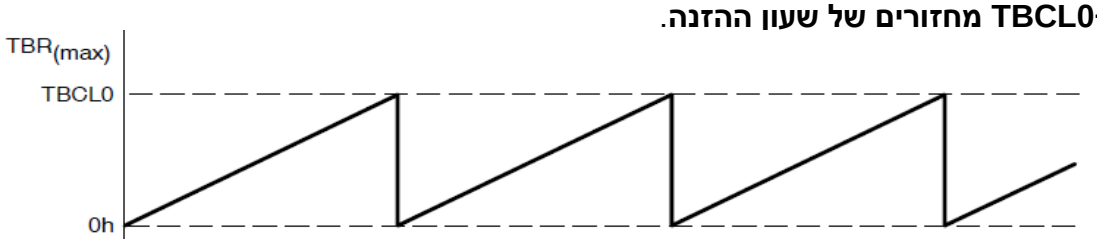
בטבלה הבאה מפורטות 4 אפשרויות מנייה של הטיימר בכתיבה ל-2 ביטים **MCx.**

MCx	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of compare register TBCL0.
10	Continuous	The timer repeatedly counts from zero to the value TBR_{MAX} (selected by the **CNTLx** bits).
11	Up/down	The timer repeatedly counts from zero up to the value of TBCL0 and then back down to zero.

אופני פעולה של Timer_B:

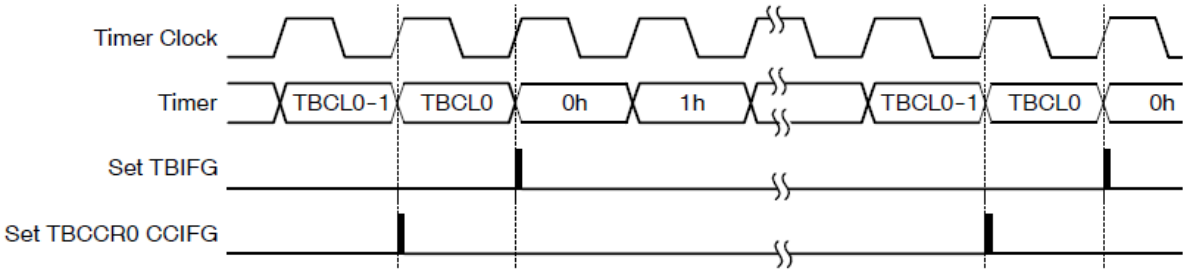
a) Up Mode:

הטיימר מבצע מנייה (ערך TBR בכל מחזור שעון מקודם ב-1) מערך 0 עד לערך הכתוב ברגיסטר TBCL0 (אנו כותבים לרגיסטר TBCCR0). הערך שאנו כותבים ב- TBCCR0 צריך להיות קטן מערך TBR_{MAX} (אחרת האופן המתאים הוא Continuous_Mode). אפשר לראות בדיאגרמה שמתבצעת השוואה בין ערך TBR לבין הערך שברגיסטר TBCL0, המנייה ממשיכה עד הגעת TBR לערך TBCL0 ואז מתבצע איפוס לטיימר (רגיסטר TBR) והמנייה מתחילה שוב מאפס. **במצב זה הטיימר סופר TBCL0+1 מחזורים של שעון ההזנה.**



★ **לרגיסטר TBCCR0** יש דגל (ביט) פסיקה CCIFG הנמצא ברגיסטר TBCCTL0 העולה ל-'1' כאשר הטיימר מגיע לערך TBCL0.

★ **לרגיסטר TBR** (לטיימר עצמו) יש דגל (ביט) פסיקה TBIFG הנמצא ברגיסטר TBCTL העולה ל-'1' כאשר הטיימר מתחיל למנות מההתחלה (סופר מ- TBCL0 לאפס).

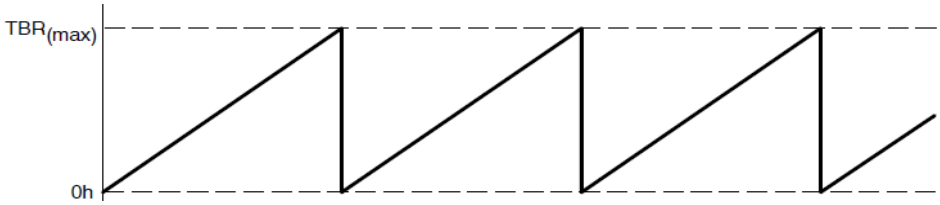


הערות:

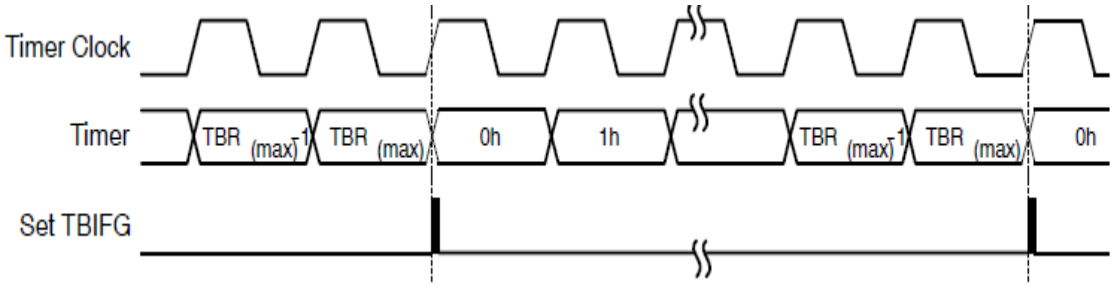
- ★ במהלך ריצת הטיימר, כאשר נבחר באופן Up Mode כאשר ערך הטיימר מעל ערך TBCL0, הטיימר מתחיל מיד למנות מאפס.
- ★ במהלך ריצת הטיימר, כאשר נשנה את ערך TBCL0 (כאשר הטעינה מ- TBCCR0 ל- TBCL0 מידית, ע"י CLLD0 = 00) אם ערכו החדש גדול מהערך הנוכחי של TBR המנייה תתבצע עד ערכו החדש. אחרת, המנייה תתחיל מאפס.

b) Continuous Mode:

הטיימר מבצע מנייה עד לערך TBR_{MAX} ואז מתחיל מההתחלה.



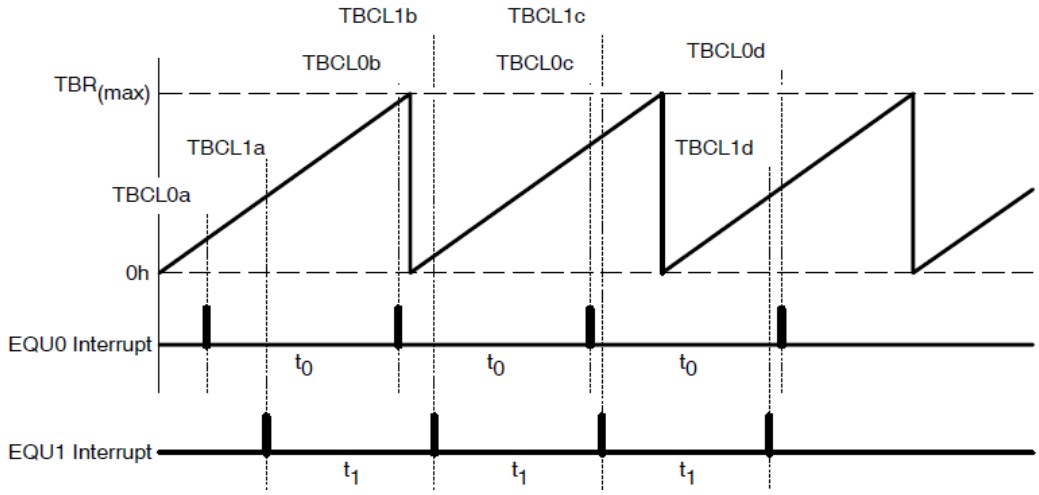
לרגיסטר TBR (לטיימר עצמו) יש דגל (ביט) פסיקה TBIFG הנמצא ברגיסטר TBCTL העולה ל-'1' כאשר הטיימר מתחיל למנות מההתחלה (סופר מ- TBR_{MAX} לאפס).



שימושים של אופן Continuous Mode:

בנוסף לשימוש המתואר לעיל, אופן זה יכול לשמש ליצירת פסיקות במרווחי זמן בלתי-תלויים (בהתבסס על התכונה שהמנייה הינה מחזורית מאפס עד לערך TBR_{MAX} ולכן רגיסטר TBCCR0 פנוי לשימוש כשאר רגיסטרי TBCCRx). לדוגמה, נקנפג את Timer_B כך שנבחר לכתוב 2 ערכים שונים לרגיסטרים TBCCR0 ו- TBCCR1 (כאשר נבחר TBCCR1 > TBCCR0). כאשר ערך TBR יגיע לערך TBCCR0 תבוצע בקשת פסיקה וב- ISR נוסיף את ערך TBCCR0 לעצמו (כמובן שיש להתחשב בערך הגלישה). מרווח הזמן שנוצר הוא t_0, אותו הדבר נבצע כאשר ערך TBR יגיע לערך TBCCR1, מרווח הזמן שנוצר הוא t_1. במצב זה מרווחי הזמן t_0,t_1 נשלטים בחומרה (עקב העבודה במקביל של רכיב פריפריאלי ל- CPU, במקרה זה Timer_B) ולא מושפעים מהשהיות של כניסה ויציאה מ-ISR.

בבקר שלנו המודול נקרא Timer_B7 מאחר וישנם 7 רגיסטרי TBCCR0 - TBCCR6, בעזרתם נוכל ליצור מקסימום 7 פסיקות במרווחי זמן בלתי-תלויים.

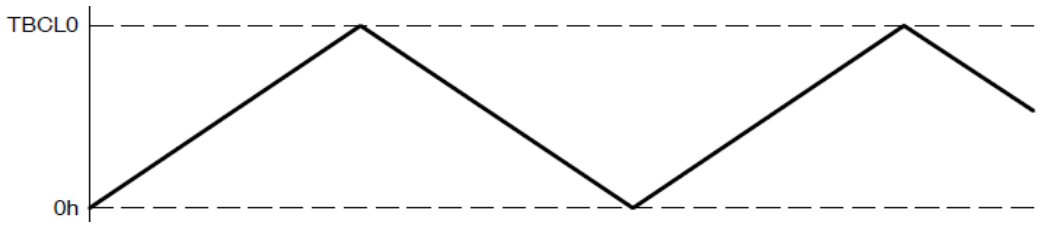


הערה: שימוש נוכל לבצע גם באופנים אחרים כאשר ערך TBCCR0 מהווה זמן מחזור לשאר הרגיסטרים TBCCR1 – TBCCR6. יש לדאוג שכאשר תוצאת חיבור (בתוך ה- ISR) ערך TBCCRx החדש לישן גדולה מ- TBCCR0, נחסר ממנה את ערך TBCCR0.

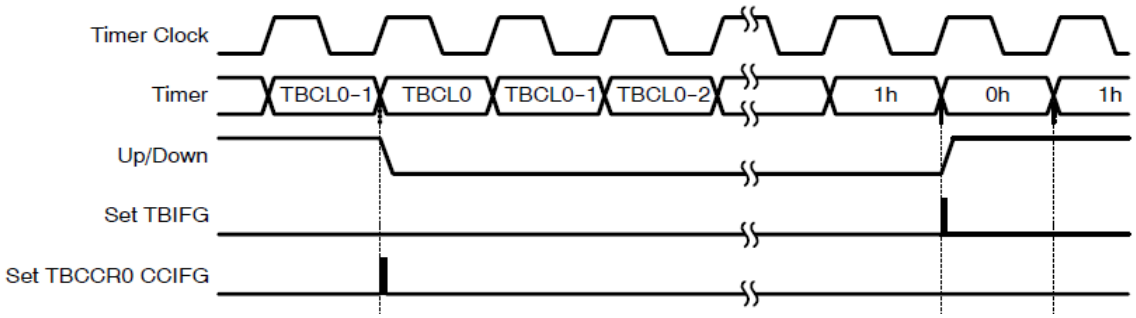
c) Up/Down Mode:

הטיימר מבצע מנייה **מעלה** מאפס עד לערך שכתבנו ברגיסטר TBCCR0 ואז מבצע מנייה מטה עד לאפס וחוזר חלילה. במצב זה מחזור מנייה הוא $2 \cdot TBCCR0$ (מס' מחזורי שעון ההזנה של הטיימר).

הערה: כאשר $TBR_{max} < TBCCR0$ הטיימר פועל כמו באופן Continuous Mode ולא תתבצע מנייה מטה TBR_{max}.



- ★ **לרגיסטר TBCCR0** יש דגל (ביט) פסיקה CCIFG הנמצא ברגיסטר TBCCTL0 העולה ל-'1' כאשר הטיימר עובר מערך TBCL0-1 לערך TBCL0.
- ★ **לרגיסטר TBR** (לטיימר עצמו) יש דגל (ביט) פסיקה TBIFG הנמצא ברגיסטר TBCTL העולה ל-'1' כאשר הטיימר עובר מערך 0x0001 לערך לאפס.



הערה:

במהלך ריצת הטיימר, כאשר נשנה את ערך TBCL0 (כאשר הטעינה מ- TBCCR0 ל- TBCL0 מידית, ע"י CLLD0 = 00) **כאשר הטיימר במנייה מטה,** הוא ממשיך עד לאפס ובמנייה מעלה מונה עד הערך החדש. **כאשר הטיימר במנייה מעלה,** אם ערכו החדש גדול מהערך הנוכחי של TBR המנייה תתבצע עד ערכו החדש ומשם מטה. אחרת, מיד המנייה תתחיל מטה.

שימושים של אופן Up/Down Mode:

בנוסף לשימוש המתואר לעיל, אופן זה תומך ביישומים הדורשים " **dead times** ", זמן נדרש שבו כל אותות PWM המיוצרים במקביל ממוצא הטיימר המוצא שלהם 0v. (הסבר בהרחבה עמ' 11).

7) Capture/Compare Modes:

במודול Timer_B7 (קיים בבקר שלנו) ישנם 7 רגיסטרי capture/compare (TBCCR0 - TBCCR6). כל רגיסטר הוא חלק מבלוק חומרה המחובר במקביל לטיימר TBR (במתואר בדיאגרמה בעמ' 5) ולו 2 שימושים. **האחד**, לכידת ערך הטיימר באחד מ-7 הרגיסטרים (לצורך קריאה בתוכנה) בהינתן טריגר של אות חיצוני ברגל הבקר המתאימה לרגיסטר שבחרנו (3 אפשרויות - עלייה / ירידה / עלייה וירידה) ובקשת פסיקה. תכונה זאת נקראת **capture**, ניתן לשימוש למדידת זמני חישוב ומשך זמן בין טריגרים וכו'. **השני,** בקשת פסיקה כאשר הטיימר יגיע לערך שכתבנו באחד מ-7 הרגיסטרים, תכונה זאת נקראת **compare.**

a) Capture Mode:

תכונה זו נבחר ע"י העלאת ביט CAP ל-'1'. לכל אחד מ-7 הבלוקים יש כניסות (capture inputs) הנקראות CCIxA ו- CCIxB (בחירתם ע"י 2 הביטים CCISx) המחוברים **לרגלי הבקר** <u>או</u> **לאותות פנימיים בבקר** (ראה datasheet) כך שנוכל להשתמש באותות דרכם כטריגר ללכידת ערך הטיימר.

צורת הטריגר (3 אפשרויות - עלייה / ירידה / עלייה וירידה) נקבעת ע"י 2 הביטים CMx.

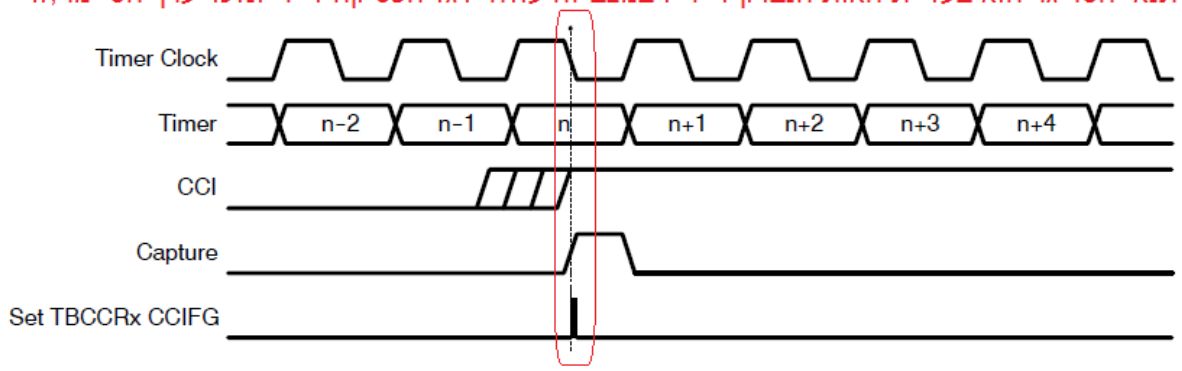
כאשר מתבצע הטריגר ערך טיימר TBR מועתק לרגיסטר TBCCRx ודגל (ביט) CCIFG עולה ל-'1'.

ניתן לקרוא בתוכנה את ערך אות הכניסה ע"י ביט CCI. מומלץ מאוד במצב של **capture** לסנכרן את אות הכניסה לשעון המעבד (MCLK) ע"י העלאת ביט SCS ל-'1'.

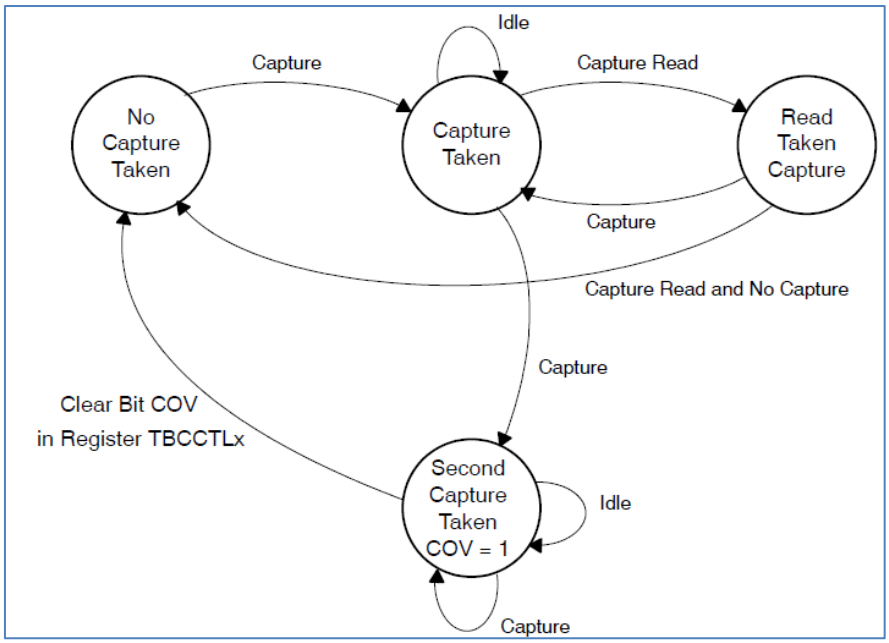
לכל אחד מרגיסטרי TBCCTLx ישנו ביט COV העולה ל-'1' כאשר יש Overflow המתריע על מצב שבו קרה טריגר חדש לפני שקראנו את ערך הטיימר מרגיסטר TBCCRx כתוצאה מהטריגר הקודם.

ביט COV מתאפס (לצורך reset) בתוכנה **בלבד**.

תנאי הטריגר הוא בעליית האות הנבדק ל-'1' . במצב זה עולה דגל הפסיקה ל-'1' ונלכד ערך הטיימר,n



להלן דיאגרמת מצבים של מחזור capture,



הערה:

ישנה אפשרות של יצירת טריגר בתוכנה (ניתן לשימוש למדידת זמן חישוב בזמן-אמת של קטע קוד או כל מודול שנבחר). ע"י שימוש ב- CCISx = 2 (חיבור לאדמה = GND = 0v) וב- CCISx = 3 (חיבור למתח VCC = 3.3v = 1 לוגי). נכתוב CCISx = 2 נבחר capture בטריגר עלייה וירידה יחד, בתחילת המיקום בקוד שממנו נרצה למדוד נכתוב CCISx = 3, ערך הטיימר שנלכך זה זמן t_0. במיקום בקוד שעד אליו נרצה למדוד נכתוב CCISx = 2 ערך הטיימר שנלכך זה זמן t_1. זמן הריצה הנמדד הוא t_1-t_0.

b) Compare Mode:

תכונה זו נבחר ע"י העלאת ביט CAP ל-'0'. בתכונה זו נשתמש לצורך יצירת אותות PWM במוצא רגלי הבקר וליצירת פסיקות במרווחי זמן נדרשים. כאשר ערך הטיימר (רגיסטר TBR) מגיע לערך ברגיסטר TBCCRx (אליו אנו כותבים), אות פנימי EQUx (במוצא ה- latch, TBCLx) עולה ל-'1', גורם לדגל הפסיקה CCIFG לעלות ל-'1' מה שגורם לאות המוצא OUTx signal להוציא אות PWM לפי אופן שנבחר (הסבר בהרחבה בעמ' 33). בקוד אנו כותבים לרגיסטר TBCCRx, ערך זה עובר בפעולת חומרה (אין גישה אליו בתוכנה) לרגיסטר TBCLx (המשמש Buffer). זמן המעבר (**העדכון**) נקבע ע"י ביטים CLLDx, לפי הטבלה הבאה:

נזכור כי ההשוואה מתבצעת בין ערך הטיימר לערך ברגיסטר TBCLx.

CLLDx	Description
00	New data is transferred from TBCCRx to TBCLx immediately when TBCCRx is written to.
01	New data is transferred from TBCCRx to TBCLx when TBR *counts* to 0.
10	New data is transferred from TBCCRx to TBCLx when TBR *counts* to 0 for **up and continuous modes**. New data is transferred to from TBCCRx to TBCLx when TBR *counts* to the old TBCL0 value or to 0 for **up/down mode.**
11	New data is transferred from TBCCRx to TBCLx when TBR *counts* to the old TBCLx value.

ישנה אפשרות שעדכון רגיסטרי **TBCLx** יתבצע בזמן מקביל (סימולטנית) בקבוצות בעזרת 2 ביטים TBCLGRPx. במצב זה זמן המעבר ייקבע ע"י ערך 2 ביטים CLLDx (חוץ ערך 0. אחרת, כל אחד יתעדכן בנפרד והעדכון לא יהיה סימולטני) של רגיסטר TBCCRx בעל האינדקס הנמוך ביותר ($x = 0,1,2,3,4,5,6$) חוץ ממצב 3, ראה פירוט בטבלה הבאה:

TBCLGRPx	Grouping	Update Control
00	None	Individual
01	TBCL1+TBCL2	TBCCR1
	TBCL3+TBCL4	TBCCR3
	TBCL5+TBCL6	TBCCR5
10	TBCL1+TBCL2+TBCL3	TBCCR1
	TBCL4+TBCL5+TBCL6	TBCCR4
11	TBCL0+TBCL1+TBCL2+TBCL3+TBCL4+TBCL5+TBCL6	TBCCR1

8) יחידת המוצא (נמצאת בבלוק capture/compare) Output Modes:

כל בלוק capture/compare מתוך 7 הקיימים ב- Timer_B7 ישנה יחידת מוצא (output_unit_0 – output_unit_6) ליצירת אותות PWM דרך אות בשם **OUTx signal** כאשר ($x = 0,1,2,3,4,5,6$) ב-8 אופני פעולה שונים המבוססים על אותות **EQU0** ו- **EQUx** כאשר $x = 1,2,3,4,5,6$ (ללא אינדקס **0**, מאחר ובמצב זה TBCCR0 משמש זמן מחזור לשאר ששת רגיסטרי TBCCRx). ישנה אפשרות לגרום ל-7 רגלי מוצא הטיימר המייצרים אותות PWM להיות במצב נתק (High Z) ע"י העלאת המתח ברגל **TBOUTH** (רגל מספר 84) ל-'1'.

שמונה אופני פעולה:

ישנם 8 אופני פעולה הנקבעים ע"י 3 ביטים **OUTMODx**. בכל האופנים (למעט אופן 0 שינוי רמה במוצא תהיה בעליית רמה של שעון המזין את הטיימר).

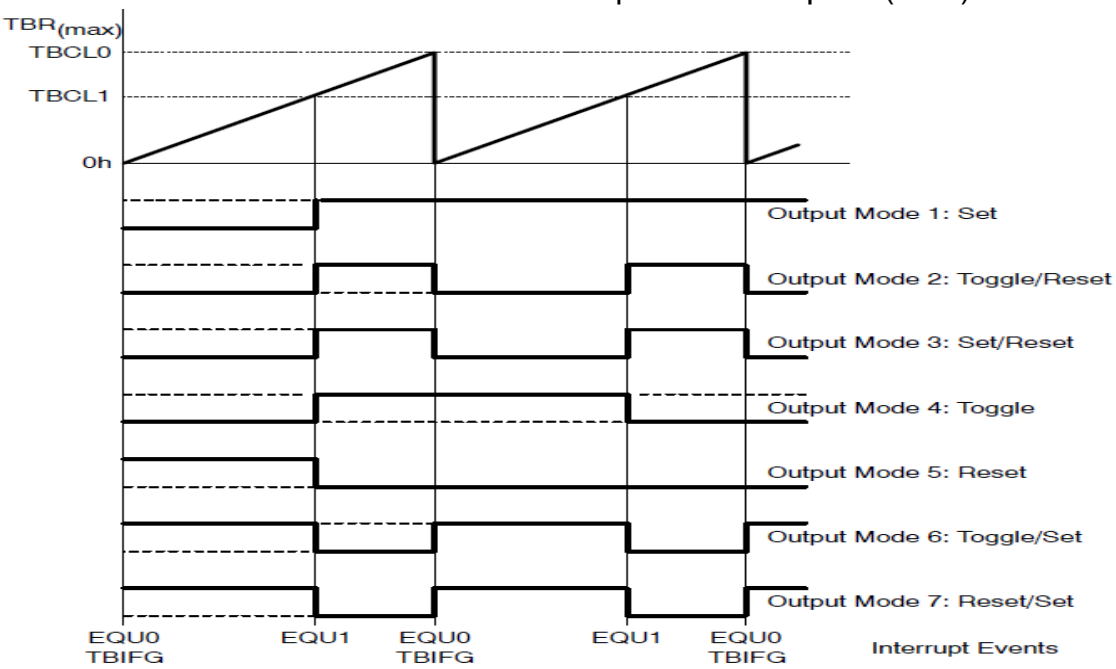
OUTMODx	Mode	Description
000	Output	The output signal OUTx is defined by the OUTx bit. The OUTx signal updates immediately when OUTx is updated.
001	Set	The output is set when the timer *counts* to the TBCLx value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer *counts* to the TBCLx value. It is reset when the timer *counts* to the TBCL0 value.
011	Set/Reset	The output is set when the timer *counts* to the TBCLx value. It is reset when the timer *counts* to the TBCL0 value.
100	Toggle	The output is toggled when the timer *counts* to the TBCLx value. The output period is double the timer period.
101	Reset	The output is reset when the timer *counts* to the TBCLx value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer *counts* to the TBCLx value. It is set when the timer *counts* to the TBCL0 value.
111	Reset/Set	The output is reset when the timer *counts* to the TBCLx value. It is set when the timer *counts* to the TBCL0 value.

הערה: אופני פעולה **2,3,6,7** אינם שימושיים עבור output_unit_0 מאחר ומתקיים בהם EQUx = EQU0 והרי TBCCR0 משמש זמן מחזור לשאר ששת רגיסטרי TBCCRx.

יצירת אות PWM כתלות בביטים **OUTMODx** ובבחירת אופן מניית הטיימר (ביטים **MCx**):

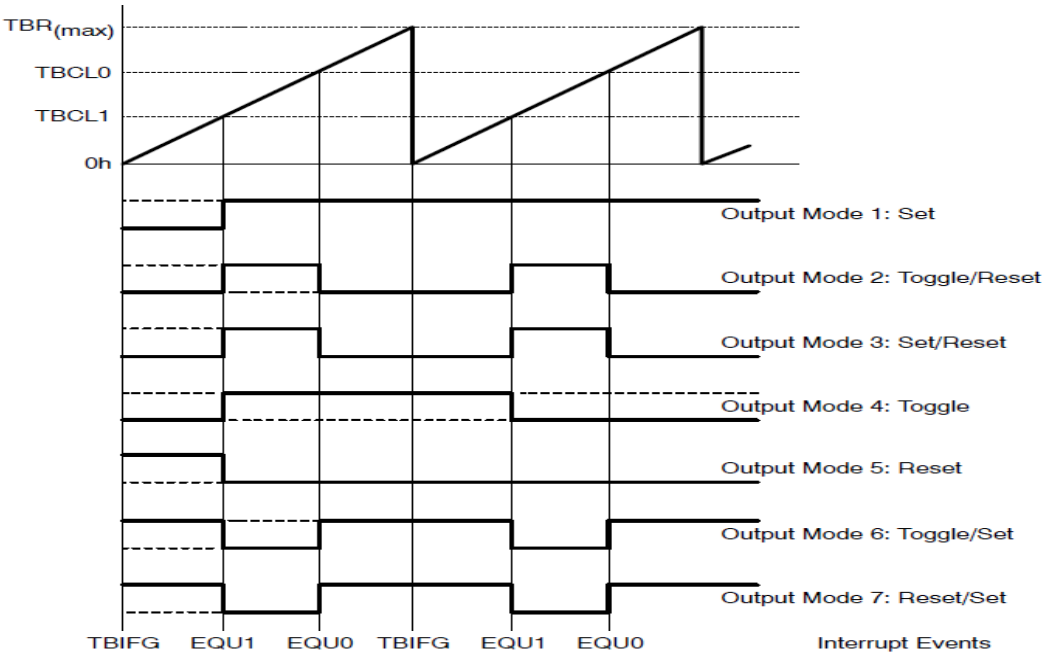
a) אופן מנייה Up Mode:

במצב זה אות המוצא של המודול OUTx signal (ראה דיאגרמת בלוקים) המחובר לרגל TBx משנה רמה לוגית **כאשר ערך TBR מגיע לערך TBCLx** (באופנים 2,3,4,6,7) ומשנה רמה שוב לוגית שוב (באופנים 2,3,6,7) **כאשר ערך TBR עובר מערך TBCL0 לאפס**. בדוגמה הבאה מופיעים שמונה סוגי האותות במוצא רגל TB1(P2.2) באופן מנייה Up Mode.



b) אופן מנייה Continuous Mode:

במצב זה אות המוצא של המודול OUTx signal (ראה דיאגרמת בלוקים) המחובר לרגל TBx משנה רמה לוגית **כאשר ערך TBR מגיע לערך TBCLx** (באופנים 2,3,4,6,7) ומשנה רמה שוב לוגית שוב (באופנים 2,3,6,7) **כאשר ערך TBR מגיע לערך TBCL0**. בדוגמה הבאה מופיעים שמונה סוגי האותות במוצא רגל TB1(P2.2) באופן מנייה Continuous Mode.

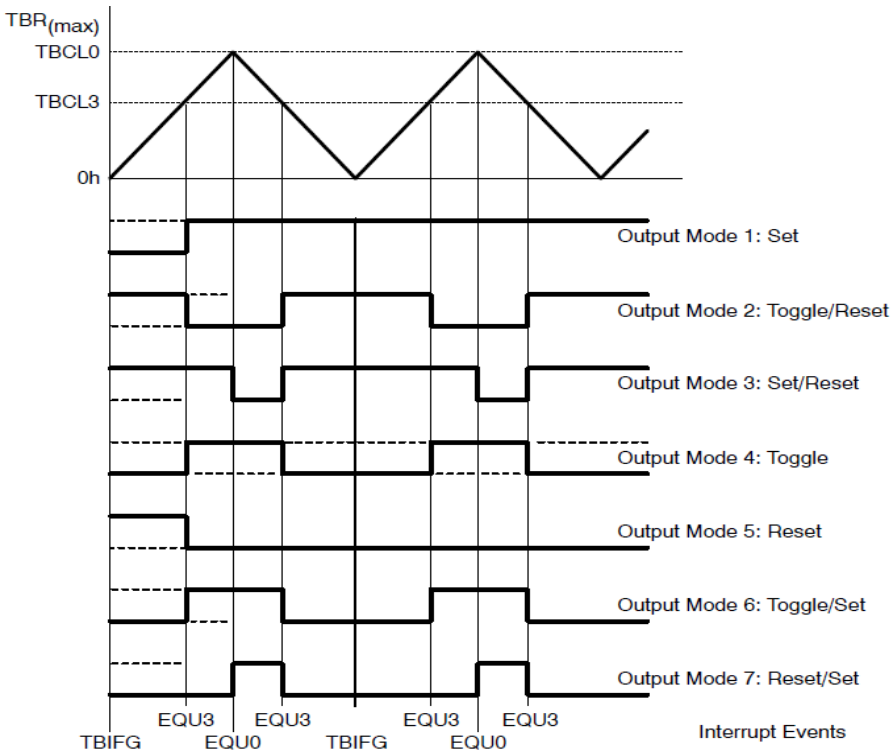


c) אופן מנייה Up/Down Mode:

במצב זה אות המוצא OUTx signal (ראה דיאגרמת בלוקים) המחובר לרגל TBx

- משנה רמה לוגית כאשר ערך TBR מגיע לערך TBCLx במנייה מעלה ובמנייה מטה (אופנים 2,4,6)
- משנה רמה לוגית כאשר ערך TBR מגיע לערך TBCL0 במנייה מעלה ומשנה שוב רמה כאשר ערך TBR מגיע לערך TBCLx במנייה מטה (אופנים 3,7).

בדוגמה הבאה מופיעים שמונה סוגי האותות במוצא רגל TB3(P2.3) באופן מנייה Up/Down Mode



לאופן מנייה Up/Down Mode ישנו יישום שימושי כאשר בין מספר מוצאי PWM נדרש **"dead times"** , זמן שכל אותות מוצאי PWM הם במצב '0', <u>כמתואר בדוגמה הבאה</u>:

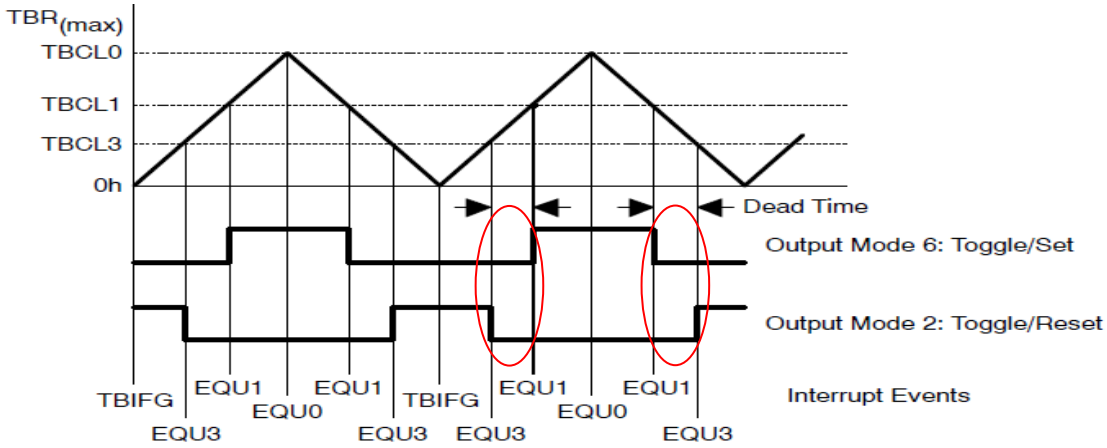
t_{dead} = Time during which both outputs need to be inactive.

t_{timer} = Cycle time of the timer clock.

TBCLx = Content of compare latch x.

$t_{dead} = t_{timer} \times (TBCL1 - TBCL3)$

The ability to simultaneously load grouped compare latches assures the dead times.



הערה כללית: במהלך פעולת הטיימר בייצור אות מוצא, כאשר נרצה להחליף את אופן אות המוצא של הטיימר. בכתיבה לביטים OUTMODx נוודא שביט אחד מהמצב הקודם נשאר ב- '1' (חוץ מהמקרה שנבחר באופן 0). על כן מומלץ לבחור אופן 7 (כמצב ביניים) ולאחר מכן לבחור במצב שנבחר.

9) פסיקות מודול Timer B:

ישנם 2 וקטורי פסיקה הקשורים למודול זה (כמובן שיש לבצע אפשור מקומי וגלובלי, כבשאר המודולים).

- עדיפות גבוהה – וקטור פסיקה TIMERB0_VECTOR הקשור לבקשת פסיקה עקב דגל **CCIFG** (הנמצא ב- TBCCTL0) הקשור לרגיסטר **TBCCR0**.
- עדיפות נמוכה (אחת פחות) – וקטור פסיקה TIMERB1_VECTOR הקשור לבקשות פסיקה עקב דגלי **CCIFG** (הנמצאים ב TBCCTL1 – TBCCTL6) הקשורים לרגיסטר **TBCCR6 - TBCCR1.**

בנוסף וקטור זה משמש עבור דגל **TBIFG** הקשור לפסיקה של הטיימר עצמו (רגיסטר TBR).

מאחר ווקטור פסיקה זה קשור ל-7 מקורות, הוא מנוהל ע"י רגיסטר **TBIV** המקבל ערך זוגי בין 2 ל- 14 המתאים לכל מקור פסיקה כך שבכניסה ל- ISR נוכל לחברו ל- PC לצורך חסכון בזמן ריצה ובתחזוקה יעילה של הקוד (כמתואר בתרגול ההכנה של מודול ADC12). מיסוך הפסיקות לא משפיע על עדכון הערכים ברגיסטר **TBIV.**

TBIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	–	
02h	Capture/compare 1	TBCCR1 CCIFG	Highest
04h	Capture/compare 2	TBCCR2 CCIFG	
06h	Capture/compare 3	TBCCR3 CCIFG	
08h	Capture/compare 4	TBCCR4 CCIFG	
0Ah	Capture/compare 5	TBCCR5 CCIFG	
0Ch	Capture/compare 6	TBCCR6 CCIFG	
0Eh	Timer overflow	TBIFG	Lowest

הדגלים CCIFG הקשורים לרגיסטרים TBCCR0 – TBCCR6 עולים ל-'1' עבור מצב של input capture כאשר ישנו טריגר ברגל הבקר (כמתואר בעמ' 6) או עבור מצב של output compare כאשר הטיימר מגיע לערך שברגיסטר TBCLx. דגל CCIFG הקשורים לרגיסטרים TBCCR0 מתאפס אוטומטית בכניסה ל- ISR של וקטור פסיקה TIMERB0_VECTOR . עבור שאר הדגלים, בכניסה ל- ISR של וקטור פסיקה TIMERB1_VECTOR דגל הפסיקה של העדיפות הגבוהה מתאפס. ניתן לאפס בתוכנה את כל הדגלים של 2 וקטורי הפסיקות.

בדוגמה הבאה מתואר שימוש נכון ברגיסטר TBIV.

```
//------------ Interrupt handler for TBCCR0 CCIFG.----------------------
            CCIFG_0_HND
                    ...     ; Start of handler Interrupt latency      6 Cycles
            RETI                                                      ;5 Cycles
//------------ Interrupt handler for TBIFG, TBCCR1 and TBCCR2 CCIFG.------------

TB_HND                  ...                 ; Interrupt latency     6 Cycles
              ADD    &TBIV,PC                 ; Add offset to Jump table    3 Cycles
              RETI                            ; Vector 0: No interrupt      5 Cycles
              JMP    CCIFG_1_HND              ; Vector 2: Module 1          2 Cycles
              JMP    CCIFG_2_HND              ; Vector 4: Module 2          2 Cycles
              RETI                            ; Vector 6
              RETI                            ; Vector 8
              RETI                            ; Vector 10
              RETI                            ; Vector 12

TBIFG_HND                                      ; Vector 14: TIMOV Flag
              ...                              ; Task starts here
              RETI                            ; 5 Cycles

CCIFG_2_HND                                   ; Vector 4: Module 2
                  ...                         ; Task starts here
              RETI                            ; Back to main program 5 Cycles

// The Module 1 handler shows a way to look if any other
// interrupt is pending: 5 cycles have to be spent, but
// 9 cycles may be saved if another interrupt is pending
CCIFG_1_HND                                   ; Vector 6: Module 3
                  ...                         ; Task starts here
              JMP TB_HND                      ; Look for pending ints 2
```

10)תכונות דומות ושונות בין Timer_B ל- Timer_A:

תוכלו להשתמש גם במודול Timer_A3 הזהה למודול Timer_B7 למעט ההבדלים הבאים:

- רק בטיימר B קיימת תכונת grouped (כמתואר לעיל)
- רק בטיימר B קיימת יכולת שמוצא אותות PWM יהיו בנתק בשימוש TBOUT.
- רק בטיימר A ישנה פונקציית סנכרון עבור capture/compare input ע"י ביט SCCI (עמ' 460 בקובץ msp430x4xx).

D. דוגמאות – Timer_B:

1) בדוגמה זו נקנפג את הטיימר כך שיוציא במקביל שישה אותות PWM ב- Duty Cycle שונה דרך רגלי הבקר TB1 - TB6 במצב מנייה Up Mode. שעון ההזנה של הטיימר הוא SMCLK וערך ברירת המחדל (כמובן שניתן לתכנתו - לא למדתם) הוא $f_{\mathrm{SMCLK_{default}}} = 32768 \cdot 32 = 2^{20} = 1{,}048{,}576Hz$.

רגיסטר TBCCR0 מהווה זמן מחזור לכל 6 האותות וערכו 511 המהווה תדר של $\frac{f_{\mathrm{SMCLK}}}{511} = 2052\mathrm{Hz}$

שאר הרגיסטרים TBCCR1 - TBCCR6 מהווים את חלק במחזור שהאות ב- '1' (מצב Duty). לדוגמה, עבור TBCCR1 = 383 האות במוצא יהיה אות PWM עם $\text{Duty Cycle} = \frac{383}{511} \cong 75\%$. כך עבור שאר האותות.

```
#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

SetupP2         bis.b   #0Ch,&P2SEL                 ; P2 option select – TB1,TB2
                bis.b   #0Ch,&P2DIR                 ; P2 outputs
SetupP3         bis.b   #0F0h,&P3SEL                ; P3 option select– TB3,TB4, TB5,TB6
                bis.b   #0F0h,&P3DIR                ; P3 outputs

SetupC0         mov.w   #512-1,&TBCCR0              ; PWM Period
SetupC1         mov.w   #OUTMOD_7,&TBCCTL1          ; CCR1 reset/set
                mov.w   #383,&TBCCR1                ; CCR1 PWM Duty Cycle
SetupC2         mov.w   #OUTMOD_7,&TBCCTL2          ; CCR2 reset/set
                mov.w   #128,&TBCCR2                ; CCR2 PWM duty cycle
SetupC3         mov.w   #OUTMOD_7,&TBCCTL3          ; CCR3 reset/set
                mov.w   #64,&TBCCR3                 ; CCR3 PWM duty cycle
SetupC4         mov.w   #OUTMOD_7,&TBCCTL4          ; CCR4 reset/set
                mov.w   #32,&TBCCR4                 ; CCR4 PWM duty cycle
SetupC5         mov.w   #OUTMOD_7,&TBCCTL5          ; CCR5 reset/set
                mov.w   #16,&TBCCR5                 ; CCR5 PWM duty cycle
SetupC6         mov.w   #OUTMOD_7,&TBCCTL6          ; CCR6 reset/set
                mov.w   #8,&TBCCR6                  ; CCR6 PWM duty cycle
SetupTB         mov.w   #TBSSEL_2+MC_1,&TBCTL       ; SMCLK, upmode
                        ;
Mainloop        bis.w   #CPUOFF,SR                  ; CPU off
                nop                                 ; Required only for debugger
;-------------------------------------------------------------------------------
        COMMON  INTVEC                              ; Interrupt Vectors
;-------------------------------------------------------------------------------
        ORG     RESET_VECTOR                        ; MSP430 RESET Vector
        DW      RESET
        END
```

2) בדוגמה זו נבצע Toggle (היפוך מצב) לרגל P9.1 בכניסה ל- ISR של וקטור פסיקה TIMERB1_VECTOR עקב over flow של הטיימר. שעון ההזנה של הטיימר הוא ACLK = 32768Hz. על כן הבהוב הלד יתבצע בתדר של 0.5Hz מאחר וערך הביטים **CNTLx** שברגיסטר TBCTL שווה 0, כלומר over flow של 16bit.
בדוגמה זו יש שימוש ברגיסטר TBIV ב- ISR של וקטור הפסיקה TIMERB1_VECTOR ממקור פסיקה TBR Overflow. הטיימר עובד באופן מנייה Continues Mode אולם זה לא רלוונטי במקרה זה, מאחר ואנו מתייחסים לדגל פסיקה של TBR (TBIFG) בלבד.

```
#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
OFIFGcheck      bic.b   #OFIFG,&IFG1            ; Clear OFIFG
                mov.w   #047FFh,R15             ; Wait for OFIFG to set again if
OFIFGwait       dec.w   R15                     ; not stable yet
        jnz     OFIFGwait
                bit.b   #OFIFG,&IFG1            ; Has it set again?
                jnz     OFIFGcheck              ; If so, wait some more

SetupP9         bis.b   #002h,&P9DIR            ; P9.1 output

SetupTB         mov.w   #TBSSEL_1+MC_2+TBIE,&TBCTL  ; ACLK, cont. mode, interrupt

Mainloop        bis.w   #LPM3+GIE,SR            ; Enter LPM3, interrupts enabled
                nop                             ; Required only for debugger
;-------------------------------------------------------------------------------
TBX_ISR                                         ;Common ISR for TBCCR1-4 and overflow
;-------------------------------------------------------------------------------
                add.w   &TBIV,PC                ; Add Timer_B offset vector
                reti                            ; Vector 0 - no interrupt
                reti                            ; Vector 1 - TBCCR1
                reti                            ; Vector 2 - TBCCR2
                reti                            ; Vector 3 - TBCCR3
                reti                            ; Vector 4 - TBCCR4
                reti                            ; Vector 5 - TBCCR5
                reti                            ; Vector 6 - TBCCR6
TB_over         xor.b   #002h,&P9OUT            ; Vector 7 - overflow (Toggle P9.1)
                reti                            ; Return from overflow ISR
;-------------------------------------------------------------------------------
        COMMON  INTVEC                          ; Interrupt Vectors
;-------------------------------------------------------------------------------
        ORG     RESET_VECTOR                    ; RESET Vector
        DW      RESET
        ORG     TIMERB1_VECTOR                  ; Timer_BX Vector
        DW      TBX_ISR
        END
```

קטע קוד זה בודק התייצבות מתנד התומך בשעון המזין את טיימר B, לא נדרש להבינו.

3) בדוגמה זו נבצע Toggle (היפוך מצב) לרגל P9.1 בכניסה ל- ISR של וקטור פסיקה TIMERB0_VECTOR הטיימר עובד באופן מנייה Up Mode וכאשר ערכו מגיע לערך שברגיסטר TBCCR0 מתבצעת בקשת פסיקה. שעון ההזנה של הטיימר במקרה זה הוא SMCLK.

```
#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
OFIFGcheck  bic.b  #OFIFG,&IFG1                 ; Clear OFIFG
            mov.w  #047FFh,R15                  ; Wait for OFIFG to set again if
OFIFGwait   dec.w  R15                          ; not stable yet
            jnz    OFIFGwait
            bit.b  #OFIFG,&IFG1                 ; Has it set again?
            jnz    OFIFGcheck                   ; If so, wait some more
SetupP9     bis.b  #002h,&P9DIR                 ; P9.1 output

SetupC0     mov.w  #CCIE,&TBCCTL0               ; TBCCR0 interrupt enabled
            mov.w  #20000,&TBCCR0
SetupTB     mov.w  #TBSSEL_2+MC_1,&TBCTL        ; SMCLK, up mode
                   ;
Mainloop    bis.w  #CPUOFF+GIE,SR               ; CPU off, interrupts enabled
            nop                                 ; Required for debugger
;-------------------------------------------------------------------------------
TB0_ISR
;-------------------------------------------------------------------------------
            xor.b  #002h,&P9OUT                 ; Toggle P9.1
            reti
;-------------------------------------------------------------------------------
        COMMON  INTVEC                          ; Interrupt Vectors
;-------------------------------------------------------------------------------
        ORG    RESET_VECTOR                     ; MSP430 RESET Vector
        DW     RESET
        ORG    TIMERB0_VECTOR                   ; Timer_B0 Vector
        DW     TB0_ISR
        END
```

קטע קוד זה בודק התייצבות מתנד התומך בשעון המזין את טיימר B, לא נדרש להבינו.

4) בדוגמה זו נבצע input capture לרגל הבקר P2.2 אשר תשמש כניסת TB1 . לרגל זו נחבר אות ריבועי מהמחולל המהווה טריגר בעליית רמה מ-'0' ל- '1' (rising edge). בהינתן טריגר נבצע חיסור בין ערך רגיסטר TBCCR1 הנוכחי לערכו הקודם. את ערך שמונת הביטים הנמוכים של תוצאת החיסור, נציג על גבי שמונת הלדים.

```
#include <msp430xG46x.h>
        org   0x1100
flag    DB    0
;-------------------------------------------------------------------------------
        RSEG  CSTACK          ; Define stack segment
;-------------------------------------------------------------------------------
        RSEG  CODE            ; Assemble to Flash memory
;-------------------------------------------------------------------------------
RESET       mov.w   #SFE(CSTACK),SP                 ; Initialize stack pointer
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL          ; Stop WDT
SetupFLL    bis.b   #XCAP14PF,&FLL_CTL0             ; Configure load caps

OFIFGcheck  bic.b   #OFIFG,&IFG1                    ; Clear OFIFG
            mov.w   #047FFh,R15                     ; Wait for OFIFG to set again if
OFIFGwait   dec.w   R15                             ; not stable yet
            jnz     OFIFGwait
            bit.b   #OFIFG,&IFG1                    ; Has it set again?
            jnz     OFIFGcheck                      ; If so, wait some more

SetupP9     bis.b   #0xff,&P9DIR                    ; Set P9.1 as Output
            clr     R4
            clr     R5

SetupTB     bis.b   #0x04,&P2SEL                    ; P2.2 PORT -->TB1 function
            clr     &TBR                            ; Clean Timer_B counter
            mov.w   #TBSSEL_1+MC_2,&TBCTL           ; ACLK, continuous mode up to 0xffff
            mov.w   #CM_1+SCS+CAP+CCIE,&TBCCTL1     ; Capture on rising edges
Mainloop    bis.w   #LPM0+GIE,SR                    ; Enter LPM3, enable interrupts
            nop                                     ; Required for Debugger
;-------------------------------------------------------------------------------
TimerB_ISR                                          ; Timer_B Interrupt Service Routine
;-------------------------------------------------------------------------------
            add.w &TBIV,PC          ; Add Timer_B offset vector
            reti                    ; Vector 0 - no interrupt
            jmp   ReadTBCCR1        ; Vector 1 - TBCCR1
            reti                    ; Vector 2 - TBCCR2
            reti                    ; Vector 3 - TBCCR3
            reti                    ; Vector 4 - TBCCR4
            reti                    ; Vector 5 - TBCCR5
            reti                    ; Vector 6 - TBCCR6
            reti                    ; Vector 7 - overflow (Toggle P9.1)
            reti                    ; Return from overflow ISR

;------------ handler for Vector 1 - TBCCR1 CCIFG ----------------------------------------
ReadTBCCR1  tst.b   flag
            jne     flag_1
            mov.w   &TBCCR1,R4
            mov.w   R4,R3
            sub     R5,R3
            mov.b   R3,&P9OUT
            xor.b   #0x01,flag
            reti

flag_1      mov.w   &TBCCR1,R5
            mov.w   R5,R3
            sub     R4,R3
            mov.b   R3,&P9OUT
            xor.b   #0x01,flag
            reti
;-------------------------------------------------------------------------------
            COMMON  INTVEC          ; Interrupt Vectors
;-------------------------------------------------------------------------------
            ORG     RESET_VECTOR        ; MSP430 RESET Vector
            DW      RESET
            ORG     TIMERB1_VECTOR      ; MSP430 TIMERB1 Interrupt Vector
            DW      TimerB_ISR
            END
```

קטע קוד זה בודק התייצבות מתנד התומך בשעון המדין את טיימר B, לא נדרש להבינו.