תיאור כללי

הפסיקה היא קטע תוכנית אשר רץ רק כאשר אירועים מוגדרים מראש קורים. הייחודיות של הפסיקה היא בכך שהמעבד מפסיק את כל הפעולות האחרות שהוא מבצע וניגש להריץ את קטע התוכנית המוגדר כפסיקה. לאחר סיום הקטע המעבד חוזר לנקודה בה הפסיק.
הפסיקות במעבד 68hc12 מתחלקות לשני חלקים עיקריים. האחד, פסיקה הרצה כל פרק זמן קבוע (8.192 מילי שניות), והשני, פסיקה הרצה בהתאם לקלט או פלט מסוים המתרחשים במעבד והוגדרו מראש על ידי המתכנת.

לפני שרוצים להגדיר פסיקות בתוכנה, מומלץ להכיר את האוגרים הבאים:

TMSK1 – מסכה שבאמצעותה ניתן להגדיר איזה כניסות מPortT יוגדרו כInterruptible.
TMSK2 – ניתן להעביר 1 לביט השמאלי ביותר של אוגר זה ואז תתבצע פסיקה כל גלישה של השעון הפנימי של המעבד, כלומר כל 8 מיקרו שניות.
TFLG2 – זהו דגל שהביט השמאלי שלו "עולה" כאשר מתקיימת גלישה בשעון המעבד.

יחד עם זאת, קיים דגל במעבד המסומן באות "I" שכאשר הא מוגדר כ-1 אין פסיקות וכאשר הוא מוגדר כ-0 הפסיקות מופעלות. שינויי המצב בדגל זה מבוצע על ידי שתי הפקודות והן CLI וSEI.

CLI – Clear Interrupt Mask Bit – פקודה זו מאפסת את דגל הפסיקה כלומר מפעילה את כל הפסיקות.

SEI – Set Interrupt Mask Bit – פקודה זו מעלה את דגל הפסיקה כלומר מפסיקה את כל הפסיקות.

כדי שהמעבד ידע איזה קטע תוכנית צריך להריץ יש לסמן את קטע זה בתווית ובסופו לאפס את הגדל המתאים לפסיקה שבוצעה (במקרה וזוהי פסיקה PortT יש לאפס את TFLG1 במקום המתאים ואם זוהי פסיקת גלישה יש לאפס את TFLG2 במקום המתאים) ולהוסיף את הפקודה RTI (Return From Interrupt) כדי שהמעבד ידע שפעולת הפסיקה הסתיימה.
למעבד יש כתובות מוגדרות מראש בזיכרון בהן יש לשים את כתובת הקטע אליו אנו רוצים שהמעבד יקפוץ ברגע הפסיקה (וקטור הפסיקה).

Vector Address	Interrupt Source	CCR Mask	Local Enable		HPRIO Value to Elevate to Highest I Bit
			Register	Bit(s)	
$FFFE, $FFFF	Reset	None	None	None	—
$FFFC, $FFFD	COP clock monitor fail reset	None	COPCTL	CME, FCME	—
$FFFA, $FFFB	COP failure reset	None	None	COP rate selected	—
$FFF8, $FFF9	Unimplemented instruction trap	None	None	None	—
$FFF6, $FFF7	SWI	None	None	None	—
$FFF4, $FFF5	XIRQ	X bit	None	None	—
$FFF2, $FFF3	IRQ	I bit	INTCR	IRQEN	$F2
$FFF0, $FFF1	Real-time interrupt	I bit	RTICTL	RTIE	$F0
$FFEE, $FFEF	Timer channel 0	I bit	TMSK1	C0I	$EE
$FFEC, $FFED	Timer channel 1	I bit	TMSK1	C1I	$EC
$FFEA, $FFEB	Timer channel 2	I bit	TMSK1	C2I	$EA
$FFE8, $FFE9	Timer channel 3	I bit	TMSK1	C3I	$E8
$FFE6, $FFE7	Timer channel 4	I bit	TMSK1	C4I	$E6
$FFE4, $FFE5	Timer channel 5	I bit	TMSK1	C5I	$E4
$FFE2, $FFE3	Timer channel 6	I bit	TMSK1	C6I	$E2
$FFE0, $FFE1	Timer channel 7	I bit	TMSK1	C7I	$E0
$FFDE, $FFDF	Timer overflow	I bit	TMSK2	TOI	$DE
$FFDC, $FFDD	Pulse accumulator overflow	I bit	PACTL	PAOVI	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I bit	PACTL	PAI	$DA
$FFD8, $FFD9	SPI serial transfer complete	I bit	SP0CR1	SPIE	$D8
$FFD6, $FFD7	SCI 0	I bit	SC0CR2	TIE, TCIE, RIE, ILIE	$D6
$FFD4, $FFD5	Reserved	I bit	—	—	$D4
$FFD2, $FFD3	ATD	I bit	ATDCTL2	ASCIE	$D2
$FFD0, $FFD1	BDLC	I bit	BCR1	IE	$D0
$FF80–$FFC1	Reserved (not implemented)	I bit	—	—	$80–$C0
$FFC2–$FFC9	Reserved (implemented)	I bit	—	—	$C2–$C8
$FFCA, $FFCB	Pulse accumulator B overflow	I bit	PBCTL	PBOVI	$CA
$FFCC, $FFCD	Modulus down counter underflow	I bit	MCCTL	MCZI	$CC
$FFCE, $FFCF	Reserved (implemented)	I bit	—	—	$CE

טבלת וקטורי הפסיקות של המעבד 68HC12B32

כעת נדגים שימוש בפסיקה בה נרצה להגדיל מונה כל קליטת גל עולה בPortT0 (PT0). לכן נריץ את קטע הקוד המתחיל בכתובית INT_DEMO :

```
INT_INIT:
        CLR   MONE                ; Clear the counter
        MOVB #$01,TCTL4           ; Set the input capture parameters
        MOVB #$01,TFLG1           ; Clear the flag
        LDX   #INT_DEMO           ; Load the label address into x
        STX   CH0_INT             ; Store the memory address in the
                                  ; specified address
        MOVB #$01,TMSK1           ; Set the interrupt mask to PT0
        CLI                       ; Enable interrupts
        …
        …

INT_DEMO:
        INC    MONE               ;  Increase the counter by 1
        MOVB   #$01,TFLG1         ;  Clear the appropriate flag
        RTI                       ;  Return from the interrupt
```

פסיקות בתוכנה

בקטע זה נתאר כיצד השתמשנו בפסיקות בתוכנה. קודם כל נעבור על המקומות בהם השתמשנו בפסיקות:

ניווט – בדיקת קיר קדמי.

ניווט – בדיקת חיישן Uvtron למציאת נר.

כיבוי נר – בדיקת פס לבן וחיישן Pyro.

קטע בדיקת הקיר הקדמי מוגדר כבר בתחילת התוכנית בקובץ Inits:

```
INT_INIT:
            MOVB #$00,TMSK1     ; Clear interrupts for PortT
            MOVB #$80,TMSK2     ; Set overflow interrupt
            LDX  #UV_INT        ; Load Uvtron interrupt mem. location
            STX  INT_CH2        ; Put memory location in channel 2
            LDX  #UV_INT        ; Load Uvtron interrupt mem. location
            STX  INT_CH3        ; Put memory location in channel 3
            LDX  #T_OVF         ; Load timer overflow subroutine adr.
                  STX  INT_OVF        ; Put timer ovf. Subroutine in
      debug12                   ;                                vector
            MOVB #$80,TFLG2     ; Clear overflow flag
            SEI                 ; Stop all interrupts
            RTS
```

כפי שניתן לראות, אנו מאפשרים פסיקת גלישה ובנוסף גם מגדירים את פסיקות הUvtron, אך לא מפעילים אותן עדיין. יחד עם זאת אנו מפסיקים את כל הפסיקות ומאפשרים אותן רק בתחילת הניווט, זאת כדי למנוע בעיות מיותרות.

ועכשיו נראה את הפונקציה T_OVF שרצה בכל גלישת שעון מעבד:

```
T_OVF:
            PSHD                    ; Push registers and accumulators
            PSHX
            PSHY
            PSHC

SIUM:
            LDX  #$28               ; Times to check IR Sensors
CHK_WALL:                           ; Check for front wall
            JSR    READ_ATD         ; Read from left forward IR
            LDAB   ADR0H
            CMPB   WALL             ; Compare to "near wall" value
            BMI    SIOM
            JSR    READ_ATD         ; Read from right forward IR
            LDAA   ADR1H
            CMPA   WALL             ; Compare to "near wall" value
            BMI    SIOM
            DEX
            BNE    CHK_WALL

NEARWALL:
            MOVB   #$FF,PLZ_TURN    ; Set the "Wall is near" flag
SIOM:
            MOVB   #$80,TFLG2       ; Clear the overflow flag
            PULC                    ; Pull registers and accumulators
            PULY
            PULX
            PULD
            RTI                     ; Return from interrupt
```

נסביר בקצרה על הפרוצדורה שלהלן. הפרוצדורה מקבלת משתנה X שבמקרה זה ערכו הוא 28, והוא קובע את מספר הפעמים שבהן חיישני האינפרא אדום יבדקו את הקיר (בבסיס 16). לאחר מכן אנו עוברים לקריאות חיישנים כאשר קוראים כל פעם מחיישן נפרד. כפי שניתן לראות החיישנים מחוברים לכניסת ה-a/d במקומות Adr0 ו- Adr1. על מנת שנזהה קיר נצטרך לעבור על כל חיישן 28 פעמים ולבדוק שהערך שהוא נותן באמת מראה שהקיר קרוב יותר או שווה לערך שנתנו בתוך תוכנת הניווט (זאת מכיוון שזמן התעדכנות החיישן הוא 33 מילי שניות ורצינו לוודא שנקבל לפחות 2 קריאות שונות). במידה והדבר קרה, נעביר לדגל (משתנה בזיכרון) שנקרא בשם PLZ_TURN (Please Turn) סימן לכך שצריך לפנות ובתוכנת הניווט נבדוק כל הזמן את דגל זה על מנת לוודא שאנו רחוקים מהקיר וניתן להמשיך בתוכנית הראשית ואם אנו קרובים לקיר נפנה.

חשוב! – יש לשים דגש על הדחיפות והמשיכות מהמחסנית בתחילת ובסוף כל פסיקה שרצה! אנו עושים זאת מכיוון שהפסיקה לא דוחפת ומושכת את האוגרים A,B,X,Y ואת הCCR (Condition Code Register) אוטומטית בכל פסיקה! דבר זה אומר שאם לדוגמה אנו מריצים לולאה שבודקת את האוגר X ובאותו הזמן רצה פסיקה שמשנה את האוגר X, התוכנית הראשית תיתקע או תיתן תוצאות שגויות מכיוון שתוכן האוגר X ישתנה!

נמשיך לפונקצית חיישן הUvtron. הכנו שתי פונקציות, UV_OV ו-UV_OFF שתפקידן הוא להפעיל את שני חיישני הUvtron ולכבותם בהתאם (הפעלה – אפשור פסיקת UvTron). בנוסף, גילינו לאחר מכן שאנו זקוקים גם לעוד שתי פונקציות: UV_ON_L ו-UV_ON_R שתפקידן הוא להדליק את החישן השמאלי או הימני בלבד בהתאמה. פונקצית הכיבוי של החיישנים זהה כמובן.

כעת נסתכל על פונקציות ההדלקה והכיבוי:

```
UV_ON:
        MOVB   #$0C,TFLG1       ; Clear UV Reads from TFLG1
        LDAA   TMSK1            ; Enable UV interrupts
        ORAA   #$0C             ; Enable UV interrupts
        STAA   TMSK1            ; Enable UV interrupts
        RTS
```

```
UV_OFF:
        MOVB   #$0C,TFLG1
        LDAA   TMSK1
        ANDA   #$F3
        STAA   TMSK1
        RTS
```

חיישני הUvtron מחוברים לכניסות pt2 וpt3 על PortT באמצעות כרטיס הInterface.
בשתי הפונקציות אנו בהתחלה מאפסים את קריאות הUV במידה והיו כאלו לפני שהחלטנו להגדיר פסיקה לחיישן. בפונקציה הראשונה אנו מבצעים פעולת Or של תוכן הזיכרון TMSK1 עם הערך 0C בבסיס 16, וזאת במקום להעביר את הערך ישירות לתא הזיכרון. ביצוע דבר זה ארוך יותר אך הוא בטוח ומונע בעיות ואף מקל על העבודה בתוכנה, מכיוון שכך אפשר בטעות להפעיל/לכבות פסיקות אחרות.
על מנת להראות את היתרון ניקח דוגמא שרירותית של המצוי בתוך האוגר TMSK1:

TMSK1	0	1	1	0	0	0	0	0

כעת נבצע עליו פעולת Or עם הערך $0C:

$0C	0	0	0	0	1	1	0	0
TMSK1+$0C	0	1	1	0	1	1	0	0

יש לשים לב שהפקודה לא שינתה את הערך הקיים בtmsk1, וזאת בניגוד לפקודה movb הרגילה (Move Bytes) אשר מוחקת את מה שהיה בtmsk1, והרי כאשר אנו מכניסים ערך חדש לtmsk1 אנו לא יודעים מה היה שם קודם.

בפונקציה השנייה אנו מבצעים פעולת And של תוכן הזיכרון שבTMSK1 עם הערך F3 בבסיס 16.

נדגים גם את ביצוע פעולה זו:

TMSK1	0	1	1	0	1	1	0	0

כעת נבצע עליו פעולת And עם הערך $F3:

$F3	1	1	1	1	0	0	1	1
TMSK1*$F3	0	1	1	0	0	0	0	0

עוד מידע ניתן לקבל בפרק על שערים לוגיים.

נמשיך ונראה את הפונקציה שרצה כאשר חיישן הUvtron קולט נר:

```
UV_INIT:
        MOVB   #$FF,UV_FLAG
        MOVB   #$0C,TFLG1
        RTI
```

בפונקציה זו אנו מאפסים או הדגל של קריאת הנר ומכניסים לתוך דגל שיצרנו (Uv_Flag) ערך שמציג זיהוי נר. לאחר מכן אנו חוזרים לתוכנית הראשית. בתוכנית אנו נבדוק את דגל זה כדי לדעת אם יש נר בחדר.

נעבור כעת לפסיקה הרצה בתוך חדר כאשר מחפשים את הנר. הפסיקה כוללת חיפוש פס לבן ו-Pyro בהתאמה:

```
AKOV_INT:
            PSHX                        ; Push registers and accumulators
            PSHD
            PSHC
            CLR   PAS_FLAG              ; Clear pas lavan flag
            LDAA  PDLC                  ; Read from pas lavan
            ORAA  #$DF                  ; Read from pas lavan
            INCA                        ; Read from pas lavan
            LBNE  TURN_OFF_I            ; If found, jump to fan function
            LDAA  PDLC                  ; Read from pas lavan
            ORAA  #$EF                  ; Read from pas lavan
            INCA                        ; Read from pas lavan
            LBNE  TURN_OFF_I            ; If found, jump to fan function
            TST   SML_FLG               ; Flag to prevent pyro chk on
                                        ; small rooms
            LBNE   PYRO_CHK             ; Go to Pyro check

END_INT_F:
            PULC                        ; Pull registers and accumulators
            PULD
            PULX
            MOVB  #$80,TFLG2            ; Clear overflow interrupt
            RTI                         ; Return from interrupt
```

כצפוי, הפונקציה דוחפת ומושכת דברים מהמחסנית (הסבר מצוי קודם בפרק זה), בודקת פס לבן וחיישן Pyro שאת הפונקציה שלו נראה בהמשך (הפונקציה רצה רק בחדרים גדולים מאחר וגילינו שאין טעם להריץ אותה בחדרים קטנים). במידה והתגלה פס לבן באחד משני החיישנים הרובוט עובר למצב כיבוי והמאוורר מופעל.