APPLICATION NOTE

# ST7 TIMER PWM DUTY CYCLE SWITCH FOR TRUE 0% or 100% DUTY CYCLE

by Microcontroller Division Applications

## INTRODUCTION

This application note presents a program that uses the 16-bit timer of the ST7 in PWM output mode. The program can be used to perform a hot switch from one duty cycle to another and obtain a true fixed period and true duty cycle percentage values between 0% and 100%.

The example program in this application note has been developed for the ST7GP family (ST72251G1 and G2).

You have to choose your ST7 device at the beginning of the program using one of the several "#define" statements provided. In this application, we chose to use a ST72251G2.

## 1 PWM DUTY CYCLE

### 1.1 DEFINITIONS

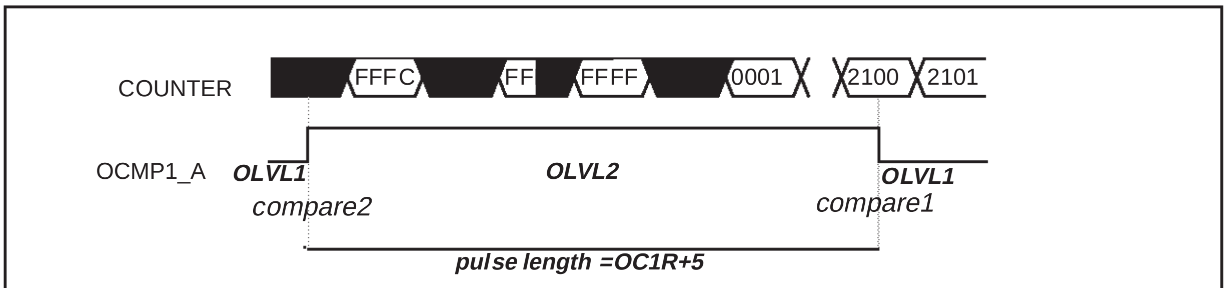
- PWM means Pulse Width Modulation. PWM mode enables the generation of a signal with a frequency and pulse length determined by the value of the timer A output compare 1 and timer A output compare 2 registers (TAOC1R and TAOC2R) of the ST7 16 bit timer. The PWM output pin is OCPM1_A (timer A output compare 1).
- x% Duty cycle means that the pulse length represents x% of the period.
- fcpu means cpu frequency (in this case 8 MHz)
- A timer tick is a timer clock cycle (here 1 timer tick is 1 µs because fcpu/8=1 MHz).
- Output level 1 (OLVL1) is a timer A control register 1 (TACR1) bit which is copied to the OCMP1 pin whenever a successful comparison occurs with the TAOC1R register. Then the TAOC1E (Timer A Output Compare 1 Enable) bit is set in the TACR2 register.
- With the OLVL1 bit, we select the level to be applied to the OCMP1 pin after a successful comparison with the TAOC1R register. With the OLVL2 bit in the TACR1 register the comparison is with the TAOC2R register.

For more precise details on how the PWM runs, please refer to the PWM chapter in the 16-BIT TIMER section of the datasheet.

## 1.2 16 BIT TIMER OPERATION IN PWM MODE

The PWM mode uses the complete Timer A Output Compare 1 function plus the TAOC2R register. An Output Compare 2 event causes the counter to be initialized to FFFCh which is the counter reset value. So after each period, the counter is reset to FFFCh. As shown in Figure 1 the pulse length and the period that you can see are 5 timer ticks longer than the values you have loaded in the TAOC1R and in TAOC2R registers.

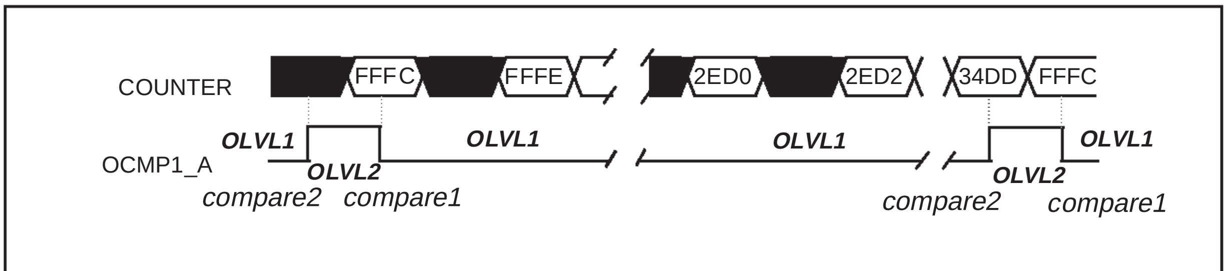
**Figure 1. General frame without correction of the 5 time tick shift**



**Note:** TAOC1R=2100h, TAOC2R=34DDh, OLVL1=0, OLVL2= 1

The minimum value available for the pulse length is programmed when you load the counter reset value in TAOC1R (FFFCh). Instead of having 0% duty cycle, the PWM mode feature produces a 1 timer tick pulse as shown in Figure 2.

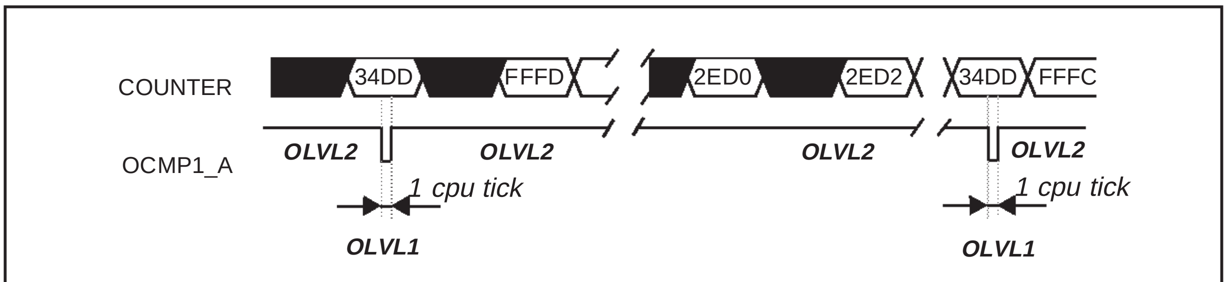
**Figure 2. 0% duty cycle minimum value with OC1R=FFFCh=Counter Reset Value**



**Note:** TAOC1R=FFFCh, TAOC2R=34DDh, OLVL1=0, OLVL2= 1

When you set the pulse length equal to the period in order to have a 100% duty cycle, you have a pulse of one CPU tick as shown in Figure 3. This could be useful for performing an external synchronization. However if you want a really 100% duty cycle you can use the TRUE routine described below.

**Figure 3. 100% duty cycle with TAOC1R=TAOC2R=Period**



**Note:** TAOC1R=34DDh, TAOC2R=34DDh, OLVL1=0, OLVL2= 1

## 1.3 PROGRAM OBJECTIVES

The three following objectives are achieved by our TRUE routine. The routine is called in the “ext0_rt” interrupt routine which changes the duty cycle value. The “init_prog“ routine loads the desired period value minus five in the TAOC2R register to obtain the correct timer tick value for the period.

### 1.3.1 Decrement the pulse length by 5 timer ticks

To obtain the correct timer tick value for the pulse length, we subtract 5 timer ticks before loading them in the TAOC1R register.

### 1.3.2 Obtain a true 0% duty cycle

On the following program the “true” routine rectifies the 1 timer tick glitch by resetting the OLVL2 bit (OLVL2=OLVL1=0) of the TACR1 register in order to have a true 0% duty cycle. This is done by comparing the pulse length value with 0000h.

### 1.3.3 Obtain a true 100% duty cycle

On the following program the “true” routine produces a true 100% duty cycle by setting OLVL1 bit (OLVL1=OLVL2=1) of the TACR1 register in order to suppress the pulse of one CPU tick. This is done by comparing the pulse length value with the period length (constant defined at the beginning of the program).

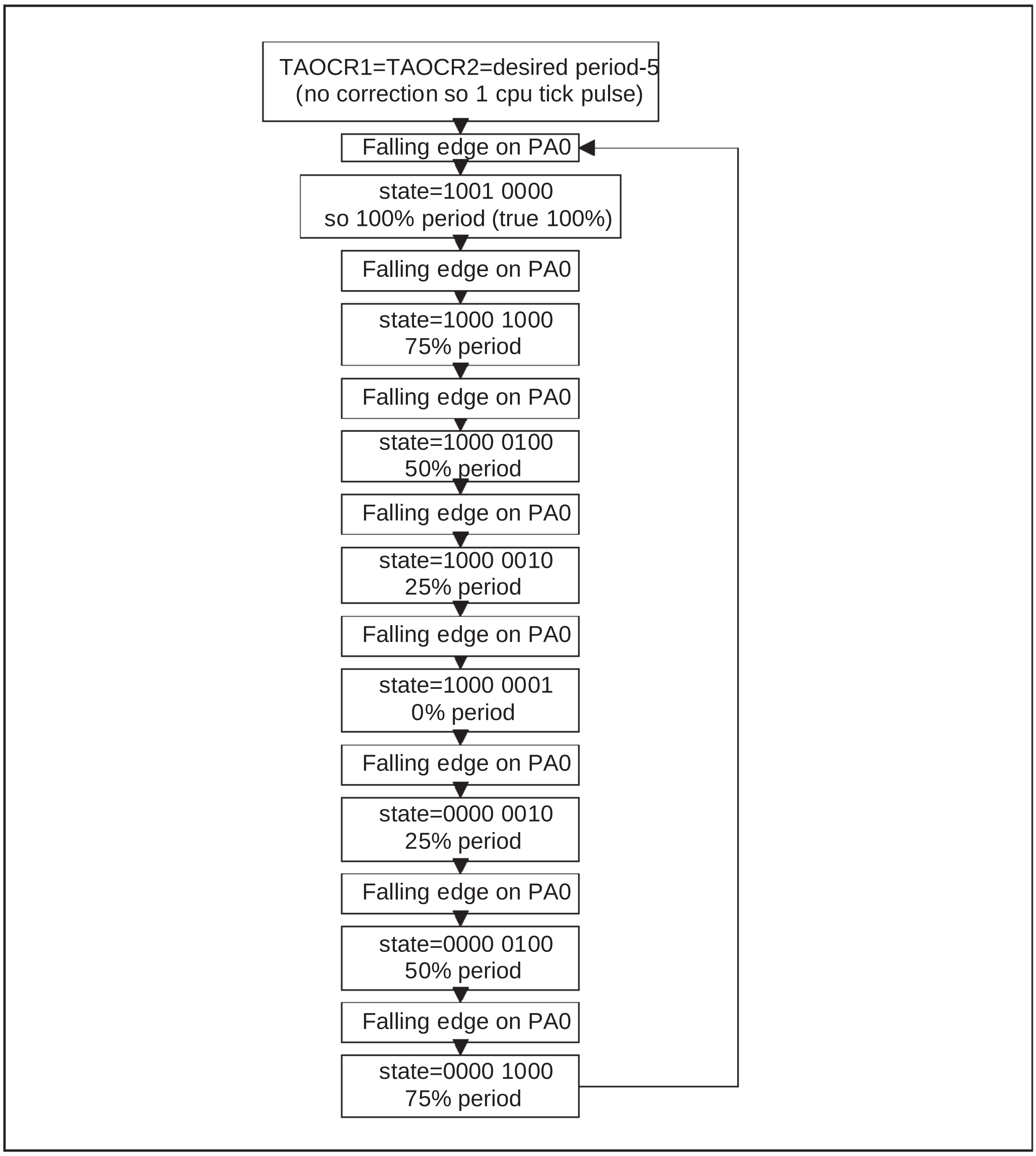
**Note**: You can use the program to switch to different duty cycles by looping through the following series of constants using the “ext0_rt” interrupt routine: 100%, 75%, 50%, 25%, 0%, 25%...

The switch is caused by a falling edge on PA0. PA0 is configured as an input with pull up and interrupt, a low level applied on this pin causes an external interrupt.

By default when you run the program you have a 100% duty cycle with the one timer tick pulse as explained above.

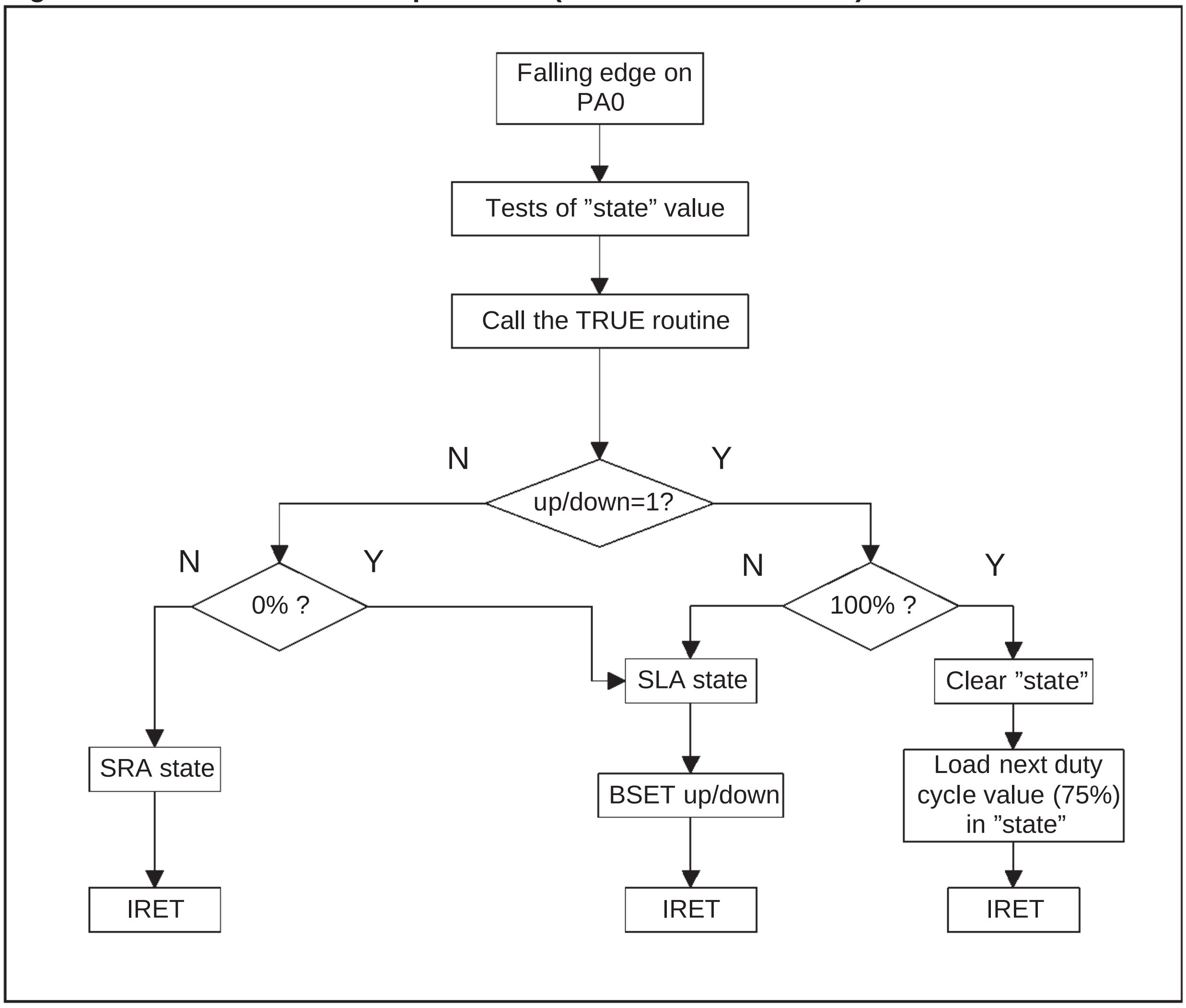
## 2 FLOWCHARTS

**Figure 4. Main routine (for ST72251G2)**



The falling edges on PA0 allow you to switch from one duty cycle value to another by executing the “ext0_rt” interrupt routine.

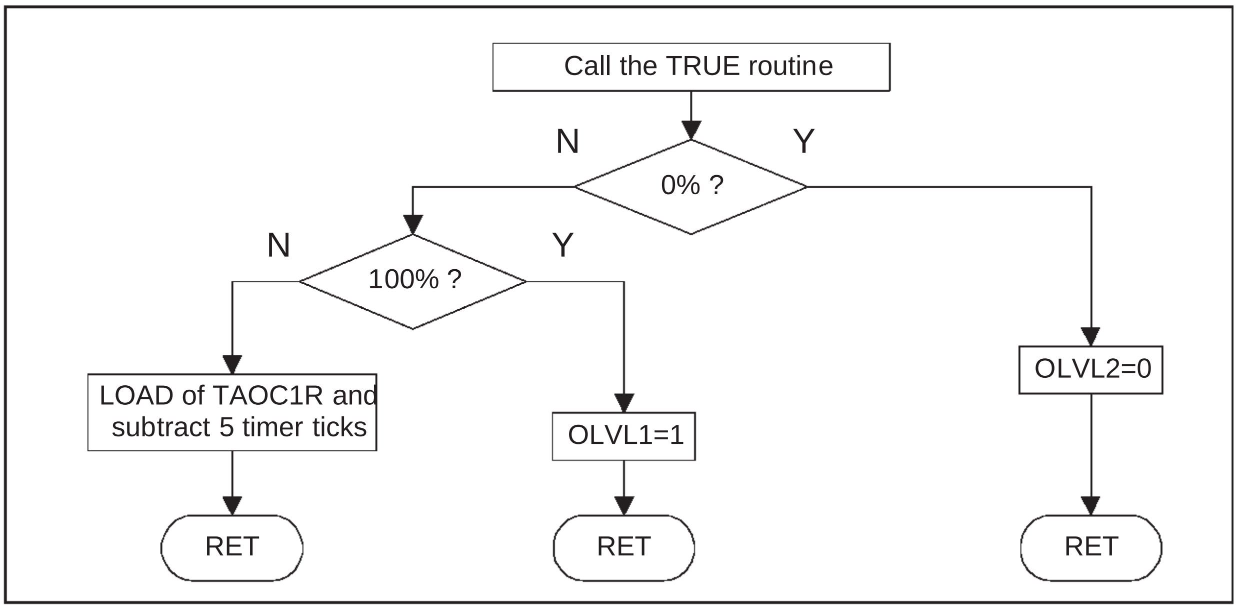
**Figure 5. The “ext0_rt” interrupt routine (here for a ST72251G2)**



To use the duty cycle values in increasing or decreasing order, we use the up/down bit of the “state” 8-bit variable. The up/down bit is the seventh bit of “state”. The others are used to code the duty cycle value. Look at the following code meaning with x =up/down:

- 100% duty cycle -> x001 0000
- 75% duty cycle -> x000 1000
- 50% duty cycle -> x000 0100
- 25% duty cycle -> x000 0010
- 0% duty cycle -> x000 0001

**Figure 6. The “true” routine flowchart**



This routine performs the 5 timer tick shift, and generates a true 0% or 100% duty cycle using the OLVL1 and OLVL2 bits.

## 3 SOFTWARE

The assembly code given below is guidance only. The complete software with all the files can be found in the software library.

```
st7/                    ; the first line is reserved
                        ; for specifying the instruction set
                        ; of the target processor

;***********************************************************************
; TITLE:                PWM.ASM
; AUTHOR:               PPG Microcontroller Application Team
; DESCRIPTION:          Demonstration Program
;                       for the Timer peripheral used in PWM configuration.
;                       This program illustrates the timer's behaviour when it is
;                       configured in PWM mode.
;                       A PWM signal will be generated on OCMP1_A pin. The length
;                       of the pulse is located in the OCR1 register while the
;                       signal's period is in the OCR2 register.
;                       We begin by a normal 100% Duty Cycle which has a one cpu tick
;                       pulse by default.
;                       At each falling edge on PA0,(cf ST72 training board
;                       button) you change the Duty Cycle by looping through this
;                       sequence of values:true 100% 75% 50% 25% 0% 25% 50% 75%
;                                                    \_______________________|
;                       The interrupt produced by the ICF1 bit at the end of each
;                       period is used to toggle PB7.
;***********************************************************************
        TITLE  "PWM.ASM"    ; this title will appear on each page of the listing file
        MOTOROLA            ; this directive forces the Motorola
                            ; format for the assembly (default)
        #INCLUDE "st72251.inc"; include st72251 registers and memory

;***********************************************************************
;    Variables, constants defined and referenced locally
;    You can define your own values for a local reference here
;***********************************************************************
        #define         up/down 7   ;To use sequence in increasing or decrasing order
        #define         OLVL1 0     ; OLVL1 bit of TACR1 register
        #define         OLVL2 2     ; OLVL2 bit of TACR1 register
        WORDS
period equ $34E2        ; For the period
duty0  equ 0000
duty25 equ {period/4}
duty50 equ {period/2}
duty75 equ {duty25+duty50}
```

```
;*****************************************************************************
;   Program code
;*****************************************************************************

        segment 'ram0'
        BYTES
.state  DS.B 1                    ; Conversion step flag

        segment 'rom'
         WORDS                    ; define subsequent addresses as words
                                  ; meaning that all instructions are located
                                  ; in the address field after 0FFh in the ST72251
                                  ; memory mapping

;-----------------------------------------------------------------------------
; ROUTINE NAME : true
; INPUT/OUTPUT : X for PWM pulse length MSB, A for PWM pulse length LSB
;        / TAOC1R,TACR1 bit 2 (output level 2)
; DESCRIPTION : Generation of true 100%, true 0% Duty Cycle and 5 timer-tick shift
;-----------------------------------------------------------------------------
.true   BRES TACR1,#OLVL1         ; Reload the default value of OLVL1
        BSET TACR1,#OLVL2         ; Reload the default value of OLVL2
        TNZ A                     ; Test if PWM pulse length LSB=0 ?
        JRNE go1
.go     TNZ X                     ; Test if PWM pulse length MSB=0 ?
        JRNE  go1
        BRES  TACR1,#OLVL2        ; To have a true 0% Duty Cycle
        RET
.go1    CP  A,#period.l           ; Test if PWM pulse length LSB=Period LSB ?
        JRNE  cont1
        CP  X,#period.h           ; Test if PWM pulse length MSB=Period MSB ?
        JRNE  cont1
        BSET  TACR1,#OLVL1        ; To obtain true 100% duty cycle
        RET

.cont1  SUB A,#5                  ; Subtraction of 5 timer ticks
        JRNC cont                 ; to have the true pulse length
        DEC X
.cont   LD TAOC1HR,X
        LD TAOC1LR,A
        RET


;-----------------------------------------------------------------------------
```

```
; ROUTINE NAME : init_PWM
; INPUT/OUTPUT : / TACR1, TACR2, PADDR, PAOR, MISCR
; DESCRIPTION :  Timer configuration for PWM generation with overflow
; interrupt turned on, and no other interrupts allowed (input capture, output
;compare)
;-----------------------------------------------------------------------
.init_PWMLD A,#$04             ; OLVL2 = 1, OLVL1 = 0
      LD TACR1,A               ; No other interrupt enabled.
      LD A,#$98          ; output on OCMP1 pin enabled and PWM mode selected.
      LD TACR2,A               ; timer clock = fcpu/8 (1us if fcpu=8MHz)
                               ; $94 for fcpu/2 and $90 for fcpu/4
      BRES PADDR,#0            ; Configuration of PA0 in Floating input with
      BSET PAOR,#0             ; interrupt

      LD A,#$10                ; Miscellaneous Register Interrupt generation
      LD MISCR,A               ; on falling edge only
      RET

;-----------------------------------------------------------------------
; ROUTINE NAME : init_prog
; INPUT/OUTPUT : period/TAOC1R, TAOC2R, oldx, oldy
; DESCRIPTION :  to begin the prog with the default 100% Duty cycle
; Period=Pulse length -> one timer tick pulse
; COMMENTS :  The Timer reset value is FFFCh so to have the true values
; we subtract 5 timer ticks
;-----------------------------------------------------------------------
.init_prog
; To get right at the first ext0_IT
      CLR state
      BSET state,#up/down; Duty cycle sequence in decreasing order at first
      BSET state,#4      ; To obtain true 100% after the first IT
;To begin with the default 100% Duty Cycle (Period=Pulse -> 1 cpu pulse)
      LD A,#period.l
      LD Y,#period.h
      SUB A,#5                 ;because of the timer reset value which is FFFCh
      JRNC sup
      DEC Y
.sup  LD TAOC2HR,Y             ; load the MSB of the PWM period
      LD TAOC2LR,A             ; load the LSB of the PWM period
      LD TAOC1HR,Y             ; load the MSB of the PWM period
      LD TAOC1LR,A             ; load the LSB of the PWM period
      RET


;************************************************************************
```

```
;*                                                                         *
;*                      MAIN-ROUTINE SECTION                               *
;*                                                                         *
;***************************************************************************
.main
      call init_PWM
      call init_prog
      RIM                     ; enable interrupt (I=0 in CCR)
.loop JRA loop                ; Wait for timer interrupt

;    *******************************************
;    *                                         *
;    * INTERRUPT SUB-ROUTINES LIBRARY SECTION  *
;    *                                         *
;    *******************************************
.dummy  IRET

.sw_rt  IRET

.ext0_rt  BTJF state,#4,bit3  ; state x001 0000->100%
      LD X,#period.h          ; copy of the MSB of the period
      LD A,#period.l          ; copy of the MSB of the period
      JRA start
.bit3 BTJF state,#3,bit2      ; state LSB 1000-> 75%
      LD X,#duty75.h          ; copy of duty cycle 75% MSB
      LD A,#duty75.l          ; copy of duty cycle 75% LSB
      JRA start
.bit2 BTJF state,#2,bit1      ; state LSB 0100-> 50%
      LD X,#duty50.h          ; copy of duty cycle 50% MSB
      LD A,#duty50.l          ; copy of duty cycle 50% LSB
      JRA start
.bit1 BTJF state,#1,bit0      ; state LSB 0010-> 25%
      LD X,#duty25.h          ; copy of duty cycle 25% MSB
      LD A,#duty25.l          ; copy of duty cycle 25% LSB
      JRA start
.bit0 BTJF state,#0,else      ; state LSB 0001->  0%
      LD X,#duty0.h
      LD A,#duty0.l
      JRA start
.else LD A,#%10010000         ; Reload state with its default value
      LD state,A
      IRET
.start  CALL true

      BTJF state,#up/down,down ; up/down=1 to increase up/down=0 to decrease
      BTJF state,#4,out
      CLR state                ; To decrease at the next IT
      BSET state,#3            ; To have 75% at the next IT
      IRET
.out  SLA state                ; To obtain the good value at the next IT
      BSET state,#up/down      ; To keep the up/down bit
      IRET
.down BTJT state,#0,out
      SRA state                ; To obtain the good value at the next IT
      IRET

.ext1_rt  IRET

.spi_rt  IRET

.tima_rt  IRET

.timb_rt  IRET

.i2c_rt  IRET

      segment 'vectit'

          DC.W dummy           ;FFE0-FFE1h location
          DC.W dummy           ;FFE2-FFE3h location
.i2c_it   DC.W i2c_rt          ;FFE4-FFE5h location
          DC.W dummy           ;FFE6-FFE7h location
          DC.W dummy           ;FFE8-FFE9h location
          DC.W dummy           ;FFEA-FFEBh location
          DC.W dummy           ;FFEC-FFEDh location
.timb_it  DC.W timb_rt         ;FFEE-FFEFh location
          DC.W dummy           ;FFF0-FFF1h location
.tima_it  DC.W tima_rt         ;FFF2-FFF3h location
.spi_it   DC.W spi_rt          ;FFF4-FFF5h location
          DC.W dummy           ;FFF6-FFF7h location
.ext1_it  DC.W ext1_rt         ;FFF8-FFF9h location
.ext0_it  DC.W ext0_rt         ;FFFA-FFFBh location
.softit   DC.W sw_rt           ;FFFC-FFFDh location
.reset    DC.W main            ;FFFE-FFFFh location
      ; This last line refers to the first line.
      ; It used by the compiler/linker to determine code zone

      END         ; Be aware of the fact that the END directive should not
                  ; be on the left of the page like the label names.
```