

## Appendix A. Assembly Source Code

```

;-----
;
; LED Gamma Correction Application
;
; DESCRIPTION:
;
; In this application, the MAXQ2000 is used to linearly increase
; the brightness of an LED (LED3 on U11) using PWM. Timer 0
; generates the PWM signal with a pulse width of 1mS. Timer 1
; triggers an interrupt every 50mS that increases/decreases the
; brightness of the LED by increasing/decreasing the PWM duty cycle.
; To achieve the illusion of a linear change in brightness, the
; PWM duty cycle values loaded by Timer 1 have been gamma corrected
; (values are stored in a look-up table located in program memory).
;
; This program compiles using the free MAX-IDE development tool
; and runs on the MAXQ2000-KIT evaluation kit (Y1=16MHz) with the
; jumpers and switches as follows:
;
;     JU1-JU2 : shunt on pins 1-2
;     JU3-JU10: no shunt
;     JU11    : shunt on pins 1-2 (to power from MAXQJTAG board)
;     SW1,SW3 : all circuits off
;     SW6     : circuits 1-7 off, circuit 8 on
;
;
; VARIABLES:
;
;     A[3] - Index of the current PWM duty cycle in the Look-Up
;           Table (LUT). The value 08000h is always added to
;           the index because program memory is mapped to 08000h
;           when executing from Utility ROM.
;     A[4] - A ptr to the moveDPl function in Utility ROM
;
;
; Copyright (C) 2005 Maxim/Dallas Semiconductor Corporation,
; All Rights Reserved.
;
; Permission is hereby granted, free of charge, to any person obtaining a
; copy of this software and associated documentation files (the "Software"),
; to deal in the Software without restriction, including without limitation
; the rights to use, copy, modify, merge, publish, distribute, sublicense,
; and/or sell copies of the Software, and to permit persons to whom the
; Software is furnished to do so, subject to the following conditions:
;
; The above copyright notice and this permission notice shall be included
; in all copies or substantial portions of the Software.
;
; THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
; OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
; MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
; IN NO EVENT SHALL MAXIM/DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM,
; DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
; OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
; THE USE OR OTHER DEALINGS IN THE SOFTWARE.
;
; Except as contained in this notice, the name of Maxim/Dallas Semiconductor
; shall not be used except as stated in the Maxim/Dallas Semiconductor
; Branding Policy.
;-----

; INCLUDES -----
#include(maxq2000.inc)

; EQUATES -----

```

```

; CODE -----
org 0000h

; *****
; * main
; *****
main:
; 0. Init DP[0] and DP[1] for word transfer mode
move DPC,#0000Ch          ; DP[0], DP[1] word transfer mode

; 1. Init P0.0, output pin used for LED PWM
move PD0.0 ,#1h          ; Set P0.0 as push-pull output
move PO0.0 ,#1h          ; Set P0.0

; 2. Load A[4] with a ptr to the moveDP1 function in Utility ROM
move DP[0] ,#0800Dh      ; DP[0] = ptr to Utility ROM function tbl
move ACC ,@DP[0]         ; ACC = 1st addr in Utility ROM function tbl
add #6                   ; Incr ACC by 6 to select ptr to moveDP1
move DP[0] ,ACC          ; DP[0] = ptr to moveDP1 entry
move A[4] ,@DP[0]        ; A[4] = moveDP1 entry

; 3. Init the PWM duty cycle look-up table index to #0
move A[3] ,#08000h       ; A[3] = 08000h (because program memory is
                        ; mapped to 08000h when in Utility ROM

; 4. Init Timer 0 to 16-bit Auto-reload/Compare timer. Timer 0
; generates the PWM signal for the LED. The overflow is
; (1/16MHz)(0xFFFF-T2R0)=1mS. T2C0 is loaded with the first
; value in the LUT
move T2CFG0 ,#000h       ; TM0 as 16-bit auto-reload at SysClk/1
move T2CNA0 ,#080h       ; TM0 interrupt enabled
move T2R0 ,#0C000h       ; TM0 reload value
move AP ,#0              ; Set AP so ACC points to A[0]
move A[0] ,A[3]          ; A[0] = index of PWM duty cycle in LUT
add #LUT                 ; A[0] = A[0] + LUT address
move DP[1] ,Acc          ; DP[1] = ptr to PWM duty cycle (A[0])
call A[4]                ; Call moveDP1 Utility ROM function
                        ; GR = new PWM duty cycle
move T2C0 ,GR            ; Timer 0 capt/comp = GR

; 5. Init Timer 1 to 16-bit Auto-reload/Compare timer. Timer 1
; increases/decreases the PWM duty cycle used by Timer 0 on every
; overflow. Overflow occurs every (128/16MHz)(0xFFFF-T2R0)=50mS
move T2CFG1 ,#070h       ; TM1 as 16-bit auto-reload at SysClk/128
move T2CNA1 ,#080h       ; TM1 interrupt enabled
move T2R1 ,#0E795h       ; TM1 reload value
move T2C1 ,#00000h       ; TM1 capt/comp register

; 6. Init Interrupts
move IV ,#IntHandler     ; Load interrupt handler
move IC.0 ,#1            ; Enable global interrupts
move IMR.3 ,#1           ; Enable module 3 interrupts (Timer 0)
move IMR.4 ,#1           ; Enable module 4 interrupts (Timer 1)

; 7. Start timers
move T2CNA0.3,#1         ; Run Timer 0
move T2CNA1.3,#1         ; Run Timer 1

; 8. Program loop
loop:
jump loop                ; Repeat ...

; *****
; * IntHandler
; *
; * Description: Handles the following IRQs:
; * - Timer 0 Overflow (Controls PWM pulse width from)
; * - Timer 0 Capture/Compare (Controls PWM duty cycle)
; * - Timer 1 Overflow (Used to change Timer 0 PWM duty cycle)

```

```

; *****
IntHandler:
    move c,T2CNB0.3          ; Load c = Timer 0 overflow flag
    jump c,IntHandlerTF0    ; If Timer 0 overflow ...
    move c,T2CNB0.1          ; Load c = Timer 0 capt/comp flag
    jump c,IntHandlerTCC0   ; If Timer 0 capt/comp flag ...
    move c,T2CNB1.3          ; Load c = Timer 1 overflow flag
    jump c,IntHandlerTF1    ; If Timer 1 overflow ...
    reti                     ; Else, return from IRQ

; If Timer 0 Overflow IRQ
IntHandlerTf0:
    move P00.0 ,#1          ; Set P0.0
    move T2CNB0.3,#0        ; Clear Timer 0 overflow flag
    reti                     ; Return from IRQ

; If Timer 0 capt/comp IRQ
IntHandlerTcc0:
    move P00.0 ,#0          ; Clear P0.0
    move T2CNB0.1,#0        ; Clear Timer 0 capt/comp flag
    reti                     ; Return from IRQ

; If Timer 1 Overflow IRQ
IntHandlerTf1:
    move AP ,#0              ; Set AP so ACC points to A[0]
    move A[0] ,A[3]          ; A[0] = index of PWM duty cycle in LUT
    add #1                   ; A[0] = A[0] + 1
    and #0803Fh              ; A[0] = A[0] modulus 64 + 08000h (because
                                ; there are 64 elements in the LUT)
    move A[3] ,Acc           ; Save A[0] as next PWM duty cycle index, A[3]
    add #LUT                 ; A[0] = A[0] + LUT address
    move DP[1] ,Acc          ; DP[1] = ptr to PWM duty cycle (A[0])
    call A[4]                ; Call moveDP1 Utility ROM function
                                ; GR = new PWM duty cycle
    move T2C0 ,GR            ; Timer 0 capt/comp = GR
    move T2CNB1.3,#0        ; Clear Timer 1 overflow flag
    reti                     ; Return from IRQ

; LOOK-UP TABLES -----
LUT:
    dw 0C0F8h ; 0    -- PWM duty cycles (gamma corrected) used to increase
    dw 0C158h ; 1    the brightness of an LED
    dw 0C1CCh ; 2
    dw 0C254h ; 3
    dw 0C2F2h ; 4
    dw 0C3A7h ; 5
    dw 0C474h ; 6
    dw 0C559h ; 7
    dw 0C658h ; 8
    dw 0C771h ; 9
    dw 0C8A6h ; 10
    dw 0C9F7h ; 11
    dw 0CB65h ; 12
    dw 0CCF1h ; 13
    dw 0CE9Bh ; 14
    dw 0D065h ; 15
    dw 0D24Fh ; 16
    dw 0D45Ah ; 17
    dw 0D686h ; 18
    dw 0D8D5h ; 19
    dw 0DB46h ; 20
    dw 0DDDBh ; 21
    dw 0E094h ; 22
    dw 0E371h ; 23
    dw 0E674h ; 24
    dw 0E99Eh ; 25
    dw 0ECEDh ; 26
    dw 0F065h ; 27
    dw 0F403h ; 28
    dw 0F7CBh ; 29
    dw 0FB8Bh ; 30

```

```
dw 0FFD5h ; 31
dw 0FFD5h ; 32
dw 0FBBBh ; 33
dw 0F7CBh ; 34
dw 0F403h ; 35
dw 0F065h ; 36
dw 0ECEDh ; 37
dw 0E99Eh ; 38
dw 0E674h ; 39
dw 0E371h ; 40
dw 0E094h ; 41
dw 0DDDBh ; 42
dw 0DB46h ; 43
dw 0D8D5h ; 44
dw 0D686h ; 45
dw 0D45Ah ; 46
dw 0D24Fh ; 47
dw 0D065h ; 48
dw 0CE9Bh ; 49
dw 0CCF1h ; 50
dw 0CB65h ; 51
dw 0C9F7h ; 52
dw 0C8A6h ; 53
dw 0C771h ; 54
dw 0C658h ; 55
dw 0C559h ; 56
dw 0C474h ; 57
dw 0C3A7h ; 58
dw 0C2F2h ; 59
dw 0C254h ; 60
dw 0C1CCh ; 61
dw 0C158h ; 62
dw 0C0F8h ; 63
```

end

; END!