# Controlling the DCO Frequency of the MSP430x11x

*Anton Muehlhofer* *Texas Instruments Deutschland*

**ABSTRACT**

The Basic Clock Module of the MSP430x11x family allows the CPU and on-chip peripherals to be supplied with a clock generated by an internal RC-type digitally-controlled oscillator (DCO) without the need for any external components. This report describes the P-control and sliding-mode-control algorithms utilized to compensate for silicon production variations and temperature and voltage drifts. A fully executable, scalable program and a schematic of a stove's vent-hood application using this clock control approach are also included.

**Contents**



**List of Figures**



**List of Tables**

# 1 Introduction

The Basic Clock Module of the MSP430x11x family allows the CPU and on-chip peripherals to be supplied with a clock generated by the internal RC-type digitally-controlled oscillator (DCO) without using any external components such as a quartz crystal or RC combination. However, silicon production variations and temperature and voltage drifts can influence the DCO frequency. To achieve a stable, well defined frequency, the DCO must be controlled by software. A P-control algorithm adjusts the DCO frequency during the initialization phase to quickly achieve an accurate frequency. During normal program flow, a sliding-mode control algorithm continuously regulates the DCO frequency to keep a very accurate frequency output. Both control algorithms can use the line frequency or a 32-kHz quartz frequency as a reference.

This report describes the P-control and sliding-mode control algorithms, and provides a fully executable, scalable program example. It also presents a schematic of a stove's vent-hood application using this clock control approach.

Assigning the software the task of controlling the system frequency opens new possibilities to the user. This approach allows the synchronization of the timer frequency to the line frequency; the result is a system that is independent of absolute line frequency. Applications thus programmed work the same way with 50-Hz and 60-Hz power systems without any need for software modifications.

# 2 DCO Characteristics

The DCO implemented in the MSP430x11x devices consists of a ring-delay line. The length of the delay line is programmable in eight steps, resulting in eight-possible discrete frequencies. This discrete frequencies can be modulated to obtain a higher number of frequencies. The number of next-higher discrete frequency cycles (0 to 31) can be programmed in software within 32 clock cycles. Using this approach, 224-different clock frequencies can be selected. Register DCOCTL determines the discrete frequency and the modulation factor. The next-higher discrete frequency is 12% higher than the current discrete frequency. Due to production variation, however, the actual difference can range between 7% and 16%. This should be taken into account when defining the P value used in the P-control algorithm.

The current injected into the DCO can be programmed using an external resistor, or by selection of one-of-eight different internal resistors. This current determines the oscillation frequency of the DCO. When selecting internal resistors, the programmable currents available result in a nominal frequency range of 130 kHz to 4.5 MHz when using a 5-V supply voltage.

In summary, the DCO frequency range extends from approximately 100 kHz to 5 MHz when using the internal resistors and the programmable delay-line length of the DCO.

# 3 Using Timer_A for Reference Frequency Measurement

A reference frequency must be provided in order to have software control of the DCO. This frequency can come from an external source such as the power line ( 50/60 Hz ), or from the internal LF oscillator (ACLK = 32.768 kHz). Since the DCO can be used to clock Timer_A, the associated capture register CCR2 can be used to measure this reference frequency. An external reference frequency (line frequency) can be connected to terminal P1.3, which is connected to the CCI2A capture input. If a 32.768-kHz quartz is required to generate the reference frequency, it is already connected to the CCI2B capture input.

The possible reference-frequency sources are:

- External frequency (such as the 50/60 Hz line frequency) connected to a Timer_A input capture pin.
- Internally-generated 32-kHz frequency (32-kHz quartz crystal is required), already connected internally to the CCI2B input capture signal of register CCR2.

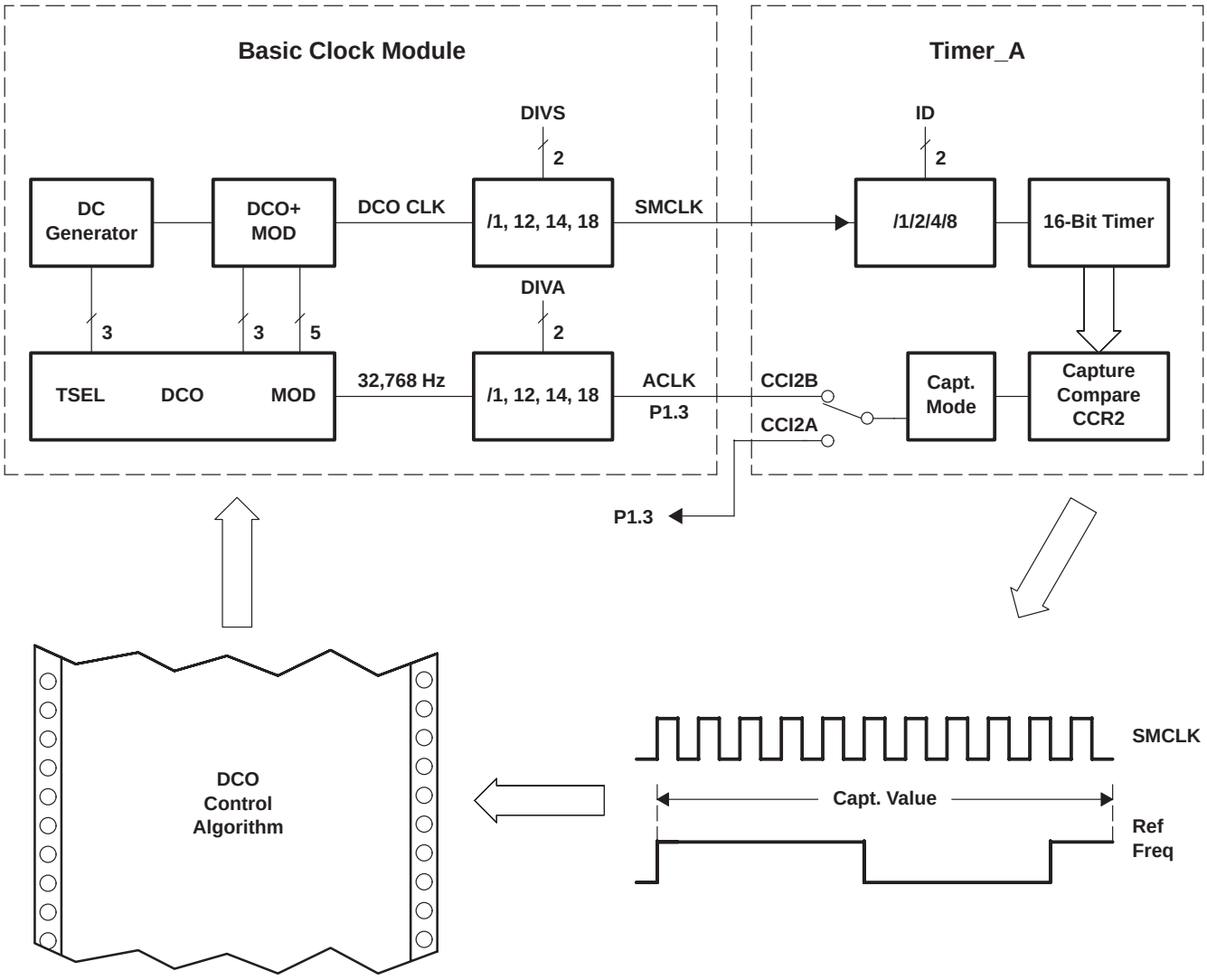


**Figure 1. Timer_A Usage for DCO Control**

In the DCO control algorithms described ahead, capture register CCR2 is used to measure either the internally-connected 32-kHz clock signal ACLK (CCI2B), or the externally-connected reference frequency at pin P1.3 (CCI2A), as shown in Figure 1. The capture unit must be configured to be triggered by a rising edge, by a falling edge, or by both rising and falling edges. This gives the user the possibility of selecting if a complete period (either falling or rising edge triggers capture), or only half the period (both edges trigger capture) of the reference frequency should be measured. These two tasks can be combined in many applications where the line frequency is the reference and a voltage zero-crossing detection is necessary (TRIAC control). In the following program example, the external 50-Hz reference frequency triggers the capture register at both edges. However, the dedicated interrupt-service routine HCCR2 provides the time for a whole period (adds two consecutive captures). Using a whole period as the reference eliminates duty-cycle variations and improves regulation accuracy.

The timer clock source is always the DCO frequency (SMCLK). This frequency can be divided by the Timer_A input divider and/or the SMCLK divider contained in the Basic Clock Module.

The following variable declarations are used in the program examples shown in Listing 1 and Listing 2:

```
.bss    VZC_LastCap,2                   ; last voltage zero-cross capture
              .bss    VZC_delta,2       ; timer value for half volt. wave
              .bss    VZC_2delta,2      ; timer value for voltage period
```

In Listing 1, the Timer_A capture interrupt-service routine for CCR2 is shown for an external 50-Hz reference frequency and will be called at every edge. The variable VZC_2delta contains the time for one full period.

**Listing 1. Capture Interrupt Service Routine for 50-Hz Reference Frequency**

```
HCCR2           push    R5
                mov     CCR2,R5                 ; meas voltage 1/2 period
                sub     VZC_LastCap,R5          ; R5 = voltage half period
                mov     VZC_delta,VZC_2delta
                add     R5,VZC_2delta           ; update volt full period
                mov     R5,VZC_delta
                mov     CCR2,VZC_LastCap
Cap2_End        pop     R5
                reti
```

In Listing 2, the Timer_A capture interrupt-service routine for CCR2 is shown using the internal 32-kHz ACLK reference frequency. It is intended to configure the capture unit to be triggered once a period only (falling or rising edge). The variable VZC_2delta contains the time for one full period.

**Listing 2. Capture Interrupt Service Routine for 32-kHz ACLK Reference Frequency**

```
HCCR2           push    R5
                mov     CCR2,R5                 ; meas voltage 1/2 period
                sub     VZC_LastCap,R5          ; R5 = voltage half period
                mov     R5,VZC_2delta
                mov     CCR2,VZC_LastCap
Cap2_End        pop     R5
                reti
```

# 4 Definitions Used in DCO Control

The following parameter definitions are used in the DCO control algorithms presented in subsequent sections. In this example, the 50-Hz line frequency should be used as the reference frequency routed to the Timer_A input capture CCR2 at pin P1.3, and the DCO frequency should be 1 MHz. These definitions have been implemented in Listing 3 for a 50-Hz reference frequency, and in Listing 4 for a 32-kHz reference frequency.

**Table 1. Control Parameters for a 50-Hz Reference and 1-MHz DCO Frequency**

| PARAMETER | SYMBOL | VALUE | COMMENT |
|---|---|---|---|
| DCO set frequency | dco_set | 1000 | Valid for 1 MHz. |
| Reference frequency | f_ref. | 50 | 50 Hz line frequency |
| Timer division rate | Tclk_div | 1 | Timer_A will be clocked with DCO frequency |
| Nominal selection resistor | Rsel_nom | 4 | 900-kHz nominal startup DCO frequency |
| Timer set frequency | fset | See Note 1 | |

NOTE 1: Look at parameter descriptions below.

**DCO set frequency:** DCO-provided frequency in kHz

**Reference Frequency:** Reference frequency in hertz to which the set frequency is synchronized. It also determines how often to perform the regulation algorithm and how often to update the DCO.

**Timer division rate:** Defines the division rate of the DCO clock that runs Timer_A. If Timer_A runs at the same speed as the DCO, it must be set to 1. Other possible values are 2, 4, 8, 16, 32, and 64 (see value in SMCLK predivider control register BCSCTL2 and Timer_A input divider register TACTL).

**Nominal Rsel:** Defines which internal resistor controls the DCO current, and consequently the nominal DCO frequency at start-up. The start-up regulation behavior can be dramatically improved by selecting the appropriate value from the table provided in the program listing.

**Timer set frequency:** Defines the number of DCO clock cycles which the DCO should perform within one reference-frequency cycle:

$$fset = \frac{dco\_set}{f\_ref \times Tclk\_div} \times 1000$$

All the settings previously described can be found in Listing 3.

**Listing 3. Definitions for 1 MHz DCO Frequency and 50-Hz Reference Frequency**

```
;-------------------------------------------------------------------------
; definitions for DCO regulation
;-------------------------------------------------------------------------
; select set frequency for the DCO
dco_set         .set    1000    ; 1000 = 1 MHz
                                ; f_soll max = 6553 -> 6.5 MHz
; select internal Resistor for nominal frequency out of below table
Rsel_nom        .set    4  ; rsel    0    1    2    3    4    5    6    7
                           ; f@3V   0.12 0.19 0.31 0.5  0.8  1.2  2.0  3.1
                           ; f@5V   0.13 0.21 0.34 0.55 0.9  1.4  2.4  4.2
; configure external voltage zero cross frequency
f_ref           .set    50      ; 50Hz
; configure Timer_A clock division rate
Tclk_div        .set    1       ; Timer clock division rate, depends on
                                ; - Timer_A clock divider
                                ; - Basic Clock SMCLK divider
; define number of DCO clocks within one reference clock
fset            .set    dco_set*1000/(f_ref*Tclk_div)
```

In Listing 4 the 32-kHz ACLK is used as the reference frequency to generate a 1-MHz DCO frequency. To get more time for DCO measurement, ACLK is divided by 8 in the basic clock module.

**Listing 4. Definitions for 1 MHz DCO Frequency and 32-kHz ACLK Reference Frequency**

```
; select set frequency for the DCO
dco_set         .set    1000    ; 1000 = 1 MHz
                                ; f_soll max = 6553 -> 6.5 MHz
; select internal Resistor for nominal frequency out of below table
Rsel_nom        .set    4  ; rsel    0    1    2    3    4    5    6    7
                           ; f@3V   0.12 0.19 0.31 0.5  0.8  1.2  2.0  3.1
                           ; f@5V   0.13 0.21 0.34 0.55 0.9  1.4  2.4  4.2
; configure external voltage zero cross frequency
f_ref           .set    32768/8 ; 32kHz /8 = 4kHz
; configure Timer_A clock division rate
Tclk_div        .set    1       ; Timer clock division rate, depends on
                                ; - Timer_A clock divider
                                ; - Basic Clock SMCLK divider
; define number of DCO clocks within one reference clock
fset            .set    dco_set*1000/(f_ref*Tclk_div)
```

# 5 P-Control Implementation

Very fast frequency regulation can be achieved by implementing a P-control algorithm if the appropriate proportional factor is selected. The formula for the P-control algorithm used in this application is:

$$dcof = dcof + prop \times (f\_set - factual)$$

Where:

*dcof* = DCO frequency (DCOCTL)
*prop* = Proportional factor
*f_set* = Set frequency
*factual* = Measured frequency

The P-control algorithm calculates the error by subtracting the number of DCO cycles measured within one complete reference clock (*factual*) from the calculated set value (*f_set*). This error is then weighted against the proportional factor (*prop*).

The best value for the proportional factor *prop* depends on the set frequency and can be calculated as:

$$prop = \frac{mod \times fstep}{f\_set \times Tclk\_div}$$

Where:

*mod* = Modulation possibilities (320)
*fstep* = Difference between discrete DCO frequency steps in %

The modulation value *mod* and discrete frequency step value *fstep* are characterized in the MSP430x11x data sheet:

```
; DCO characteristic out of data sheet
mod             .set    32      ; 32 modulation possibilities
fstep           .set    7       ; 7 % is every min discrete frequency step
```

With these DCO characteristics, and with a set frequency of 1 MHz and a 32-kHz reference frequency, the proportional factor *prop* can be calculated as:

```
; defines the proportional factor for p-control
prop            .set    mod*fstep*65536/(2*fset*Tclk_div)
```

**NOTE:** The proportional factor has been scaled by 65,536 to generate a more accurate result. The proportional factor *prop* should be limited to 7FFFh to avoid sign overflow errors within the P-control calculation routine. Therefore, the proportional factor prop is additionally divided by 2 when using 32 kHz as the reference frequency.

With these definitions and the control algorithm shown in Listing 5, the following frequency responses have been measured while changing the set frequency from 1 MHz to 4 MHz:

**Table 2. Frequency Response for 32-kHz and 50-Hz Reference Frequencies**

| SET FREQUENCY | DCO FREQUENCY 32-kHz REFERENCE prop=mod×fstep×65536/(fset×2×Tclk_div) | DCO FREQUENCY 50-Hz REFERENCE prop=mod×fstep×65536/(fset×Tclk_div) |
|---|---|---|
| 1.0 MHz | 1.0 MHz | 1.0 MHz |
| 4.0 MHz | 1.39 MHz | 1.48 MHz |
| 4.0 MHz | 1.92 MHz | 2.49 MHz |
| 4.0 MHz | 2.50 MHz | 3.24 MHz |
| 4.0 MHz | 2.77 MHz | 3.96 MHz |
| 4.0 MHz | 3.57 MHz | 4.24 MHz |
| 4.0 MHz | 4.05 MHz | 4.04 MHz |

As shown in Table 2, the DCO frequency is moved from 1 MHz to 4.05 MHz within 6 control cycles, including changing the internal Rsel twice. This fast-response time reflects the appropriate selection of the proportional factor for the P-control algorithm. The *prop* factor was divided by 2 to prevent sign overflow in the P-control calculation routine. Therefore, the regulation dynamics have been decreased as well. This compromise is intended to make the same control routine usable with much-slower reference frequencies such as the 50-Hz line frequency. To further improve the control accuracy, the reference frequency (ACLK) can be divided using software: instead of counting the DCO clocks within just one reference-frequency clock cycle, the clocks are integrated over more than one reference-frequency cycle. The exact regulation of the DCO frequency can also be performed using the sliding-mode control algorithm, which is described in the following section. The P-control algorithm shown in Listing 5 needs 66 bytes (without a multiplication routine), and requires 227 cycles.

**Listing 5. P-Control Algorithm**

```
;-------------------------------------------------------------------------
; FUNCTION DEF: dco_ctl
; DESCRIPTION:  dco p controller, calculates the formula
;               dcof = dcof+[p*(set-actual)/2^16]
;               dcof = dco frequency contained in &DCOCTL
;               input:  R5 = measured timer value
;               output: new dco frequency
; REGISTER USE: R4, R5, R10-R15
; CALLS:        MPYS
; ORIGINATOR:   Anton Muehlhofer
;-------------------------------------------------------------------------
dco_ctl         push    R4
                mov.b   &DCOCTL,R4
                mov     #prop,IROP2L
                mov     #fset,IROP1
                sub     R5,IROP1        ; build difference set-actual
                call    #MPYS           ; p*(set-actual)
                add     IRACM,R4        ; dcof=dcof+[p*(set-actual)/2^16]
                cmp     #dco_min,R4     ; new Rsel has to be configured?
                jl      dco_dec_Rsel
                cmp     #dco_max,R4
                jge     dco_inc_Rsel
                mov.b   R4,&DCOCTL      ; configure new dco frequency
                pop     R4
                ret
dco_dec_Rsel    dec.b   Rsel
                jmp     dco_ctl_end
dco_inc_Rsel    inc.b   Rsel
dco_ctl_end     call    #Rsel_set
                mov.B   #60h,&DCOCTL    ; center dco into new frequency
                pop     R4
                ret
```

# 6 Sliding Mode Control Implementation

This type of control algorithm only changes the output value by one. This results in a robust and accurate regulation characteristic with a very long transient response. Another advantage is the small algorithm, which only needs to calculate the error between the set and measured values and to return the sign as the output. This makes it ideal for controlling the DCO of MSP430x11x devices when the DCO frequency is already near the set frequency, or when a longer transient response can be tolerated. Requiring no multiplication results in a very fast algorithm that can be recalled quite often without significantly slowing down the processor.

The function shown in Listing 6 expects the set frequency *fset* defined as described before and the measured DCO frequency in the 16-bit variable VZC_2delta coming from the Timer_A capture interrupt-service routine. After the control algorithm settles down, the maximum error is ±0.5% within 32 DCO-clock cycles. When a very high long-term accuracy is required, the measured error can be integrated and compensated for after a certain time to achieve an even smaller overall error. This function needs 90 bytes and requires only 22 cycles.

**Listing 6. Sliding Mode Control Algorithm**

```
;-------------------------------------------------------------------------
; FUNCTION DEF: dco_step
; DESCRIPTION:  regulates the DCO by changing the DCO value by 1 only
;               VCZ_2delta must contain the measured DCO frequency
;               fset defines the set frequency
; REGISTER USE: R10, R11
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         01. Dec. 98
;-------------------------------------------------------------------------
dco_step        mov.b   &DCOCTL,R11     ; read current DCO selection
                cmp.b   #dco_max,R11    ; should Rsel be increased ?
                jeq     inc_Rsel        ; yes
                cmp.b   #dco_min,R11    ; should Rsel be decreased ?
                jeq     dec_Rsel        ; yes
                mov     VZC_2delta,R10  ; read timer for 2 zero crosses
                cmp     #fset,R10
                jlo     inc_dco_step    ; increase DCO by one
                jeq     dco_step_end    ; do nothing
dec_dco_step    dec.b   &DCOCTL         ; decrease DCO by one
                ret
inc_dco_step    inc.b   &DCOCTL
dco_step_end    ret
inc_Rsel        mov.b   &BCSCTL1,R10    ; is max Rsel already selected?
                bic.b   #0f8h,R10
                cmp.b   #7,R10
                jge     dco_step_end    ; yes, cannot increase Rsel!
                inc.b   &BCSCTL1        ; Rsel + 1
                mov.b   #60h,&DCOCTL    ; center DCO
                ret
dec_Rsel        mov.b   &BCSCTL1,R10    ; is min Rsel already selected?
                bic.b   #0f8h,R10
                cmp.b   #0,R10
                jeq     dco_step_end    ; yes, cannot decrease Rsel!
                dec.b   &BCSCTL1        ; Rsel - 1
                mov.b   #60h,&DCOCTL    ; center DCO
                ret
```

# 7 Application Example

Figure 2 shows the schematic of an actual stove's vent-hood application. The MSP430C111 does not use any external components for clock generation in order for cost savings. The DCO is used to supply the CPU and the peripherals. The frequency is controlled by the line frequency. This works well with 50 Hz and 60 Hz power line frequencies without modification since the DCO frequency is synchronized with the line frequency. Only the absolute DCO frequency is different: 1 MHz at 50-Hz line frequency, and 1.2 MHz at 60-Hz line frequency.

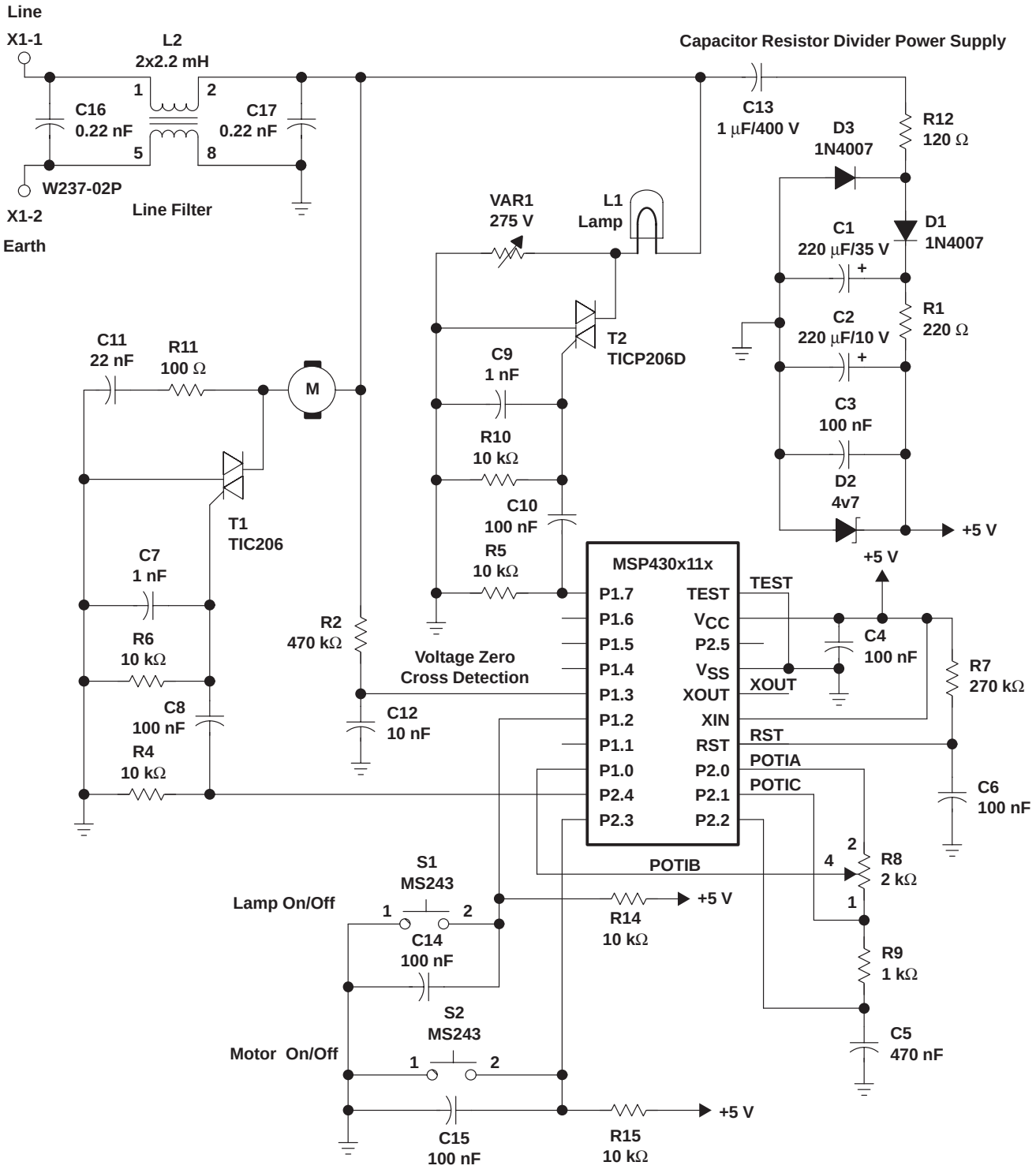


**Figure 2. DCO Control Application Example**

```
;*******************************************************************
; File Name:    fll32k.asm
; Project:      software fll for MSP430x112
; Originator:   Anton Muehlhofer   (Texas Instruments Deutschland)
;
; Target Sys:   MSP430x112
;
; Description:  Complete example how the DCO could be controlled
;
; Status:       tested with 32kHz quartz and 1 MHz DCO frequency
;
; Last Update:  Feb 11, 1999
;*******************************************************************
                .include 110.inc
;-------------------------------------------------------------------------
; Clock Oscillator Setup
;-------------------------------------------------------------------------
LF1             .set    0
XT1             .set    1
DCO             .set    2
; select clock for CPU (CPU_CLK) and peripheral clock SMCLK (PP_CLK)
CPU_CLK         .set    DCO             ; CPU clock
PP_CLK          .set    DCO             ; Peripheral clock
;-------------------------------------------------------------------------
; definitions for DCO regulation
;-------------------------------------------------------------------------
; select set frequency for the DCO
dco_set         .set    1000    ; 1000 = 1 MHz
                                ; f_soll max = 6553 -> 6.5 MHz
; select internal Resistor for nominal frequency out of below table
Rsel_nom        .set    4  ; rsel    0    1    2    3    4    5    6    7
                           ; f@3V   0.12 0.19 0.31 0.5  0.8  1.2  2.0  3.1
                           ; f@5V   0.13 0.21 0.34 0.55 0.9  1.4  2.4  4.2
; configure external voltage zero cross frequency
f_ref           .set    32768/8 ; 32kHz / 8 = 4 kHz
; configure Timer_A clock division rate
Tclk_div        .set    1       ; Timer clock division rate, depends on
                                ; - Timer_A clock divider
                                ; - Basic Clock SMCLK divider
; define number of DCO clocks within one reference clock
fset            .set    dco_set*1000/(f_ref*Tclk_div)
; DCO characteristic out of data sheet
mod             .set    32      ; 32 modulation possibilities
fstep           .set    7       ; 7 % is every min discrete frequency step
; defines the proportional factor for p-control
prop            .set    mod*fstep*65536/(fset*Tclk_div)
dco_min         .set    05h     ; below this value, select lower Rsel
dco_max         .set    0E2h    ; above this value, select higher Rsel
VZC             .set    08h     ; Reference frequency input at P1.3
Testpin         .equ    40h     ; P1.6
;-------------------------------------------------------------------------
; Memory Setup
;-------------------------------------------------------------------------
```

```
; define interrupt vector table start address
Ivecs           .set    0FFE0h
; define Stack pointer and available RAM
RAM_Start       .set    00200h          ; Free Memory startadress
RAM_End         .set    00300h          ; RAM endadress
SP_Start        .set    00300h          ; stackpointer
EPROM_Start     .set    0f000h          ; start of 4k EPROM
;-------------------------------------------------------------------------
; Status flag definition
;-------------------------------------------------------------------------
Task_Ovr        .equ    01h             ; another task indicator
;-------------------------------------------------------------------------
; definitions for function MPYS, MACS, MPYU, MACU
;-------------------------------------------------------------------------
IRACM           .equ    R10             ; result high word
IRACL           .equ    R11             ; result low word
IROP1           .equ    R12             ; first operand
IROP2L          .equ    R13             ; second operand low word
IROP2M          .equ    R14             ; second operand high word
IRBT            .equ    R15             ; bit test register mpy
;-------------------------------------------------------------------------
; Variable definitions
;-------------------------------------------------------------------------
                .bss    dummy,0,200h
                .even
                .bss    VZC_LastCap,2   ; last voltage zero-cross capture
                .bss    VZC_delta,2     ; timer value for half volt. wave
                .bss    VZC_2delta,2    ; timer value for voltage period
                .bss    Rsel,1          ; selection for internal resistor
                .bss    Status,1        ; general purpose status byte
                .bss    TStat_10ms,1    ; Task status

;=========================================================================
; Program starts here after reset
;=========================================================================
                .sect   ”MAIN”,EPROM_Start
RESET
                mov     #SP_Start,SP    ; initialize stack pointer
                mov     #(WDTHold+WDTPW),&WDTCTL ; Stop Watchdog Timer
                clr.b   &IE1
                clr.b   &IFG1           ; clears oscillator fault
;-------------------------------------------------------------------------
; select CPU clock
;-------------------------------------------------------------------------
.if CPU_CLK = XT1
    bis.b  #XTS,&BCSCTL1                ; select XT1, disable LF1
    bis.b  #SELM0+SELM1,&BCSCTL2        ; select XT1 as MCLK
.elseif CPU_CLK = DCO
    bic.b  #SELM0+SELM1,&BCSCTL2        ; select DCO as MCLK
.endif

;-------------------------------------------------------------------------
; select Peripheral clock
```

```
;-------------------------------------------------------------------------
        .if PP_CLK = XT1
                bis.b   #XTS,&BCSCTL1   ; select XT1, disable LF1
                bis.b   #SELS,&BCSCTL2  ; select XT1 as SMCLK and ACLK
        .elseif PP_CLK = LF1
                bic.b   #XTS,&BCSCTL1   ; select LF1, disable XT1
                bis.b   #SELS,&BCSCTL2  ; select LF1 as SMCLK and ACLK
        .elseif PP_CLK = DCO
                bic.b   #SELS,&BCSCTL2  ; select DCO as SMCLK
        .endif

;-------------------------------------------------------------------------
; select RSEL for DCO
;-------------------------------------------------------------------------
                mov.b   #Rsel_nom,Rsel  ; initialize Rsel
                call    #Rsel_set       ; initialize Rsel in BCSCTL1
                bis.b   #030h,&BCSCTL1  ; ACLK / 8
;-------------------------------------------------------------------------
; output of SMCLK and ACLK for control purposes
;-------------------------------------------------------------------------
                bis.b   #010h,&P1SEL    ; output SMCLK at p1.4
                bis.b   #010h,&P1DIR
                bis.b   #01h,&P2SEL     ; output ACLK at p2.0
                bis.b   #01h,&P2DIR
;-------------------------------------------------------------------------
; initialize global variables
;-------------------------------------------------------------------------
                clr.b   TStat_10ms      ; clear Task status register
                clr.b   Status          ; clear general purpose status
;-------------------------------------------------------------------------
; configure Timer A
;-------------------------------------------------------------------------
                mov     #0204h,&TACTL   ; counts up continuous
                                        ; no interrupt at overflow
                                        ; timer cleared
                                        ; timer stopped, need input select
                bic     #04h,&TACTL     ; release timer clear
                bis     #20h,&TACTL     ; start timer with MCLK
; configure Reference Voltage input capture pin at CCR2
                mov     #0101100100100000b,&CCTL2
                                        ; CC2 is in capture mode
                                        ; zero cross capture pin CCI2B
                                        ; configure synchronous cap mode
                                        ; pos edge triggers capture
                                        ; enables cap2 interrupt
                mov     #0FFFFh,R5      ; wait some time to let the
Wait            nop                     ; ACLK settle
                dec     R5
                jnz     Wait
; configure testpin output low
                bic.b   #Testpin,&P1OUT
                bis.b   #Testpin,&P1DIR
;-------------------------------------------------------------------------
```

```
; enable interrupts
;-------------------------------------------------------------------------
                bis     #0010h,&CCTL2       ; enable cctl2 interrupt
                eint
;-------------------------------------------------------------------------
; Startup phase
;-------------------------------------------------------------------------
startup_0       mov     #4,R5               ; loop counter
startup_1       bit.b   #Task_Ovr,Status    ; initialize VZC values
                jz      startup_1
                bic.b   #Task_Ovr,Status
                dec     R5
                jnz     startup_1           ; perform 4 loops min.
;-------------------------------------------------------------------------
; regulate DCO with P control algorithm
;-------------------------------------------------------------------------
                mov     VZC_2delta,R5
                call    #dco_ctl
;=========================================================================
; main loop
;=========================================================================
                .newblock
mainloop
                bit.b   #Task_Ovr,Status
                jz      mainloop
                call    #Task_10ms
                jmp     mainloop
;-------------------------------------------------------------------------
; Task management for tasks called every 10 ms (voltage zero cross)
;-------------------------------------------------------------------------
Task_10ms       mov.b   TStat_10ms,R5
                mov.b   Tbl_10ms(R5),R5
                add     R5,PC
Tbl_10ms
                .byte   T1-Tbl_10ms         ; Task 1
                .byte   T2-Tbl_10ms         ; Task 2: dco control
                .byte   T3-Tbl_10ms         ; Task 3
                .byte   T4-Tbl_10ms         ; last task
;-------------------------------------------------------------------------
T1              nop
                jmp     T_End
;-------------------------------------------------------------------------
T2
                call    #dco_step           ; dco control by just 1 DCO step
                jmp     T_End
;-------------------------------------------------------------------------
T3              nop
                jmp     T_End
;-------------------------------------------------------------------------
T4
                call    #dco_step
                clr.b   TStat_10ms          ; start with task T1 next time
                jmp     T_Ret
```

```
;-------------------------------------------------------------------------
T_End           inc.b   TStat_10ms
T_Ret           bic.b   #Task_Ovr,Status
                ret
;-------------------------------------------------------------------------
; FUNCTION DEF: dco_ctl
; DESCRIPTION:  dco p controller, calculates the formula
;               dcof = dcof+[p*(set-actual)/2^16]
;               dcof = dco frequency contained in &DCOCTL
;               input:  R5 = measured timer value
;               output: new dco frequency
; REGISTER USE: R4, R5, R10-R15
; CALLS:        MPYS
; ORIGINATOR:   Anton Muehlhofer
; DATE:         13. Nov. 98
;-------------------------------------------------------------------------
dco_ctl         push    R4
                mov.b   &DCOCTL,R4
                mov     #prop,IROP2L
                mov     #fset,IROP1
                sub     R5,IROP1        ; build difference set-actual
                call    #MPYS           ; p*(set-actual)
add             IRACM,R4                ; dcof=dcof+[p*(set-actual)/2^16]
                cmp     #dco_min,R4     ; new Rsel has to be configured?
                jl      dco_dec_Rsel
                cmp     #dco_max,R4
                jge     dco_inc_Rsel
                mov.b   R4,&DCOCTL      ; configure new dco frequency
                pop     R4
                ret
dco_dec_Rsel    dec.b   Rsel
                jmp     dco_ctl_end
dco_inc_Rsel    inc.b   Rsel
dco_ctl_end     call    #Rsel_set
                mov.B   #60h,&DCOCTL    ; center dco into new frequency
                pop     R4
                ret
;-------------------------------------------------------------------------
; FUNCTION DEF: dco_step
; DESCRIPTION:  regulates the DCO by changing the DCO value by 1 only
;               VCZ_2delta must contain the measured DCO frequency
;               fset defines the set frequency
; REGISTER USE: R10, R11
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         01. Dec. 98
;-------------------------------------------------------------------------
dco_step        mov.b   &DCOCTL,R11     ; read current DCO selection
                cmp.b   #dco_max,R11    ; should Rsel be increased ?
                jeq     inc_Rsel        ; yes
                cmp.b   #dco_min,R11    ; should Rsel be decreased ?
                jeq     dec_Rsel        ; yes
                mov     VZC_2delta,R10  ; read timer for 2 zero crosses
```

```
                cmp     #fset,R10
                jlo     inc_dco_step    ; increase DCO by one
                jeq     dco_step_end    ; do nothing
dec_dco_step    dec.b   &DCOCTL         ; decrease DCO by one
                ret
inc_dco_step    inc.b   &DCOCTL
dco_step_end    ret
inc_Rsel        mov.b   &BCSCTL1,R10    ; is max Rsel already selected?
                bic.b   #0f8h,R10
                cmp.b   #7,R10
                jge     dco_step_end    ; yes, cannot increase Rsel!
                inc.b   &BCSCTL1        ; Rsel + 1
                mov.b   #60h,&DCOCTL    ; center DCO
                ret
dec_Rsel        mov.b   &BCSCTL1,R10    ; is min Rsel already selected?
                bic.b   #0f8h,R10
                cmp.b   #0,R10
                jeq     dco_step_end    ; yes, cannot decrease Rsel!
                dec.b   &BCSCTL1        ; Rsel - 1
                mov.b   #60h,&DCOCTL    ; center DCO
                ret
;-------------------------------------------------------------------------
; FUNCTION DEF: Rsel_set
; DESCRIPTION:  initializes the RSEL in BCSCTL with the value defined in
;               the variable RSEL
; REGISTER USE: R14, R15
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         13. Nov. 98
;-------------------------------------------------------------------------
Rsel_set        mov.b   &BCSCTL1,R14
                mov.b   R14,R15
                bic.b   #0F8h,R15
                cmp.b   Rsel,R15
                jeq     Rsel_end
                jl      Rsel_inc
Rsel_dec        dec.b   R14
                mov.b   R14,&BCSCTL1
                jmp     Rsel_set
Rsel_inc        inc.b   R14
                mov.b   R14,&BCSCTL1
                jmp     Rsel_set
Rsel_end        ret
;-------------------------------------------------------------------------
; FUNCTION DEF: MPYS, MACS, MPYU, MACU
; DESCRIPTION:  16bit x 16 bit signed and unsigned multiply
; REGISTER USE: R4-R9
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         13. Okt. 98
;-------------------------------------------------------------------------
                .newblock
MPYS            clr     IRACL
```

```
                clr     IRACM
MACS            tst     IROP1
                jge     $1
                sub     IROP2L,IRACM
$1              tst     IROP2L
                jge     MACU
                sub     IROP1,IRACM
                jmp     MACU

MPYU            clr     IRACL
                clr     IRACM
MACU            clr     IROP2M
                mov     #1,IRBT
$2              bit     IRBT,IROP1
                jz      $3
                add     IROP2L,IRACL
                addc    IROP2M,IRACM
$3              rla     IROP2L
                rlc     IROP2M
                rla     IRBT
                jnc     $2
                ret

;-------------------------------------------------------------------------
; Timer A Capture/Compare Interrupt Service Routine
;-------------------------------------------------------------------------
Int_TA_IV       add     &TAIV,PC ; read TA intrpt vector and clear int flag
                reti
                jmp     HCCR1
                jmp     HCCR2
Int_TA_end      reti
;-------------------------------------------------------------------------
; Voltage Zero Cross Interrupt Service Routine
; used by MSP430x11x
;-------------------------------------------------------------------------
HCCR2           push    R5
                mov     &CCR2,R5                ; meas voltage 1/2 period
                sub     VZC_LastCap,R5          ; R5 = voltage half period
                mov     R5,VZC_2delta
                mov     &CCR2,VZC_LastCap
                bis.b   #Task_Ovr,Status        ; initiate Task proceeding
Cap2_End        pop     R5
                reti

;-------------------------------------------------------------------------
; CCR1 Interrupt Service Routine     - unused -
;-------------------------------------------------------------------------
HCCR1           reti
;-------------------------------------------------------------------------
; unused Timer_A interrupt service routines
;-------------------------------------------------------------------------
Int_TA_CC0
                reti
```

```
;-------------------------------------------------------------------------
; all other interrupts
;-------------------------------------------------------------------------
Int_P1                                  ; Port1
Int_P2                                  ; Port2
Int_WDT_T                               ; Watchdog / Timer
               reti

;=========================================================================
; Interrupt vectors
;=========================================================================
                .sect   ”Int_Vect”,Ivecs
                .word   RESET           ; Port0, bit 2 to 7, n/a at 112
                .word   RESET           ; Basic Timer, n/a at 112
                .word   Int_P1          ; Port1
                .word   Int_P2          ; Port2
                .word   RESET           ; Timer Port, n/a at 112
                .word   RESET           ; no source
                .word   RESET           ; UART Transmit, n/a at 112
                .word   RESET           ; UART Receive, n/a at 112
                .word   Int_TA_IV       ; Timer A
.word   Int_TA_CC0      ; Timer A
                .word   Int_WDT_T       ; Watchdog/Timer, Timer mode
                .word   RESET           ; no source
                .word   RESET           ; UART handler, n/a at 112
                .word   RESET           ; P0.0, n/a at 112
                .word   RESET           ; NMI, Osc. fault
                .word   RESET           ; POR, ext. Reset, Watchdog
                .end
```

# Appendix A DCO Control With External 50-Hz Reference Frequency

```
;*******************************************************************
; File Name:    fll.asm
; Project:      software fll for MSP430x112
; Originator:   Anton Muehlhofer   (Texas Instruments Deutschland)
;
; Target Sys:   MSP430x112
;
; Description:  Complete example how the DCO could be controlled
;
; Status:       tested with 50Hz-100Hz external frequency
;
; Last Update:  Dec 8, 1998
;*******************************************************************
                .include 110.inc
;-------------------------------------------------------------------
; Clock Oscillator Setup
;-------------------------------------------------------------------
LF1             .set    0
XT1             .set    1
DCO             .set    2
; select clock for CPU (CPU_CLK) and peripheral clock SMCLK (PP_CLK)
CPU_CLK         .set    DCO             ; CPU clock
PP_CLK          .set    DCO             ; Peripheral clock
;-------------------------------------------------------------------
; definitions for DCO regulation
;-------------------------------------------------------------------
; select set frequency for the DCO
dco_set         .set    1000    ; 1000 = 1 MHz
                                ; f_soll max = 6553 -> 6.5 MHz
; select internal Resistor for nominal frequency out of below table
Rsel_nom        .set    4  ; rsel    0    1    2    3    4    5    6    7
                           ; f@3V   0.12 0.19 0.31 0.5  0.8  1.2  2.0  3.1
                           ; f@5V   0.13 0.21 0.34 0.55 0.9  1.4  2.4  4.2
; configure external voltage zero cross frequency
f_ref           .set  50
; configure Timer_A clock division rate
Tclk_div        .set    1       ; Timer clock division rate, depends on
                                ; - Timer_A clock divider
                                ; - Basic Clock SMCLK divider
; define number of DCO clocks within one reference clock
fset            .set    dco_set*1000/(f_ref*Tclk_div)
;  these values are fixed
mod             .set    32      ; 32 modulation possibilities
fstep           .set    7       ; 7 % is every min discrete frequency step

; defines the proportional factor for p-control
prop            .set    mod*fstep*65536/(fset*Tclk_div)
dco_min         .set    05h     ; below this value, select lower Rsel
dco_max         .set    0E2h    ; above this value, select higher Rsel
VZC             .set    08h     ; Reference frequency input at P1.3
Testpin         .equ    40h     ; P1.6
```

```
;-------------------------------------------------------------------------
; Memory Setup
;-------------------------------------------------------------------------
; define interrupt vector table start address
Ivecs           .set    0FFE0h
; define Stack pointer and available RAM
RAM_Start       .set    00200h  ; Free Memory startadress
RAM_End         .set    00300h  ; RAM endadress
SP_Start        .set    00300h  ; stackpointer
EPROM_Start     .set    0f000h  ; start of 4k EPROM
;-------------------------------------------------------------------------
; Status flag definition
;-------------------------------------------------------------------------
Task_Ovr        .equ    01h     ; another task indicator
;-------------------------------------------------------------------------
; definitions for function MPYS, MACS, MPYU, MACU
;-------------------------------------------------------------------------
IRACM           .equ    R10     ; result high word
IRACL           .equ    R11     ; result low word
IROP1           .equ    R12     ; first operand
IROP2L          .equ    R13     ; second operand low word
IROP2M          .equ    R14     ; second operand high word
IRBT            .equ    R15     ; bit test register mpy
;-------------------------------------------------------------------------
; Variable definitions
;-------------------------------------------------------------------------
                .bss    dummy,0,200h
                .even
                .bss    VZC_LastCap,2   ; last voltage zero-cross capture
                .bss    VZC_delta,2     ; timer value for half volt. wave
                .bss    VZC_2delta,2    ; timer value for voltage period
                .bss    Rsel,1          ; selection for internal resistor
                .bss    Status,1        ; general purpose status byte
                .bss    TStat_10ms,1    ; Task status

;=========================================================================
; Program starts here after reset
;=========================================================================
                .sect   "MAIN",EPROM_Start
RESET
                mov     #SP_Start,SP    ; initialize stack pointer
                mov     #(WDTHold+WDTPW),&WDTCTL ; Stop Watchdog Timer
                clr.b   &IE1
                clr.b   &IFG1           ; clears oscillator fault
;-------------------------------------------------------------------------
; select CPU clock
;-------------------------------------------------------------------------
        .if CPU_CLK = XT1
                bis.b  #XTS,&BCSCTL1           ; select XT1, disable LF1
                bis.b  #SELM0+SELM1,&BCSCTL2   ; select XT1 as MCLK
        .elseif CPU_CLK = DCO
                bic.b  #SELM0+SELM1,&BCSCTL2   ; select DCO as MCLK
        .endif
```

```
;-------------------------------------------------------------------------
; select Peripheral clock
;-------------------------------------------------------------------------
        .if PP_CLK = XT1
                bis.b   #XTS,&BCSCTL1   ; select XT1, disable LF1
                bis.b   #SELS,&BCSCTL2  ; select XT1 as SMCLK and ACLK
        .elseif PP_CLK = LF1
                bic.b   #XTS,&BCSCTL1   ; select LF1, disable XT1
                bis.b   #SELS,&BCSCTL2  ; select LF1 as SMCLK and ACLK
        .elseif PP_CLK = DCO
                bic.b   #SELS,&BCSCTL2  ; select DCO as SMCLK
        .endif

;-------------------------------------------------------------------------
; select RSEL for DCO
;-------------------------------------------------------------------------
                mov.b   #Rsel_nom,Rsel  ; initialize Rsel
                call    #Rsel_set       ; initialize Rsel in BCSCTL1
;-------------------------------------------------------------------------
; output of SMCLK and ACLK for control purposes
;-------------------------------------------------------------------------
                bis.b   #010h,&P1SEL    ; output SMCLK at p1.4
                bis.b   #010h,&P1DIR
                bis.b   #01h,&P2SEL     ; output ACLK at p2.0
                bis.b   #01h,&P2DIR
; switch XT1 off
                bis     #OSCOFF,SR      ; switch XT1 off, saves current
;-------------------------------------------------------------------------
; initialize global variables
;-------------------------------------------------------------------------
                clr.b   TStat_10ms      ; clear Task status register
                clr.b   Status          ; clear general purpose status
;-------------------------------------------------------------------------
; configure Timer A
;-------------------------------------------------------------------------
                mov     #0204h,&TACTL   ; counts up continuous
                                        ; no interrupt at overflow
                                        ; timer cleared
                                        ; timer stopped, need input select
                bic     #04h,&TACTL     ; release timer clear
                bis     #20h,&TACTL     ; start timer with MCLK
; configure Reference Voltage input capture pin at CCR2
                bic.b   #VZC,&P1DIR     ; input capture zero cross
                mov     #1100100100100000b,&CCTL2
                                        ; CC2 is in capture mode
                                        ; zero cross capture pin CCI2A
                                        ; configure synchronous cap mode
                                        ; both edges triggers capture
                                        ; enables cap2 interrupt

                bis.b   #VZC,&P1SEL
```

```
; configure testpin output low
                bic.b   #Testpin,&P1OUT
                bis.b   #Testpin,&P1DIR
;-------------------------------------------------------------------------
; enable interrupts
;-------------------------------------------------------------------------
                bis     #0010h,&CCTL2   ; enable cctl2 interrupt
                eint
;-------------------------------------------------------------------------
; Startup phase
;-------------------------------------------------------------------------
startup_0       mov     #6,R5
startup_1       bit.b   #Task_Ovr,Status        ; initialize VZC values
                jz      startup_1
                bic.b   #Task_Ovr,Status
                dec     R5
                jnz     startup_1               ; perform 4 loops min.
;-------------------------------------------------------------------------
; regulate DCO
;-------------------------------------------------------------------------
                mov     VZC_2delta,R5
                call    #dco_ctl
;=========================================================================
; main loop
;=========================================================================
                .newblock
mainloop
                bit.b   #Task_Ovr,Status
                jz      mainloop
                call    #Task_10ms
                jmp     mainloop
;-------------------------------------------------------------------------
; Task management for tasks called ervery 10 ms (voltage zero cross)
;-------------------------------------------------------------------------
Task_10ms       mov.b   TStat_10ms,R5
                mov.b   Tbl_10ms(R5),R5
                add     R5,PC
Tbl_10ms
                .byte   T1-Tbl_10ms     ; Task 1
.byte   T2-Tbl_10ms     ; Task 2: dco control
                .byte   T3-Tbl_10ms     ; Task 3
                .byte   T4-Tbl_10ms     ; last task
;-------------------------------------------------------------------------
T1              nop
                jmp     T_End
;-------------------------------------------------------------------------
T2
                call    #dco_step       ; dco control by just 1 DCO step
                jmp     T_End
;-------------------------------------------------------------------------
T3              nop
                jmp     T_End
;-------------------------------------------------------------------------
```

```
T4
                call    #dco_step
                clr.b   TStat_10ms      ; start with task T1 next time
                jmp     T_Ret
;-------------------------------------------------------------------------
T_End           inc.b   TStat_10ms
T_Ret           bic.b   #Task_Ovr,Status
                ret
;-------------------------------------------------------------------------
; FUNCTION DEF: dco_ctl
; DESCRIPTION:  dco p controller, calculates the formula
;               dcof = dcof+[p*(set-actual)/2^16]
;               dcof = dco frequency contained in &DCOCTL
;               input:  R5 = measured timer value
;               output: new dco frequency
; REGISTER USE: R4, R5, R10-R15
; CALLS:        MPYS
; ORIGINATOR:   Anton Muehlhofer
; DATE:         13. Nov. 98
;-------------------------------------------------------------------------
dco_ctl         push    R4
                mov.b   &DCOCTL,R4
                mov     #prop,IROP2L
                mov     #fset,IROP1
                sub     R5,IROP1        ; build difference set-actual
                call    #MPYS           ; p*(set-actual)
                add     IRACM,R4        ; dcof=dcof+[p*(set-actual)/2^16]
                cmp     #dco_min,R4     ; new Rsel has to be configured?
                jl      dco_dec_Rsel
                cmp     #dco_max,R4
                jge     dco_inc_Rsel
                mov.b   R4,&DCOCTL      ; configure new dco frequency
                pop     R4
                ret
dco_dec_Rsel    dec.b   Rsel
                jmp     dco_ctl_end
dco_inc_Rsel    inc.b   Rsel
dco_ctl_end     call    #Rsel_set
                mov.b   #60h,&DCOCTL    ; center dco into new frequency
pop     R4
                ret
;-------------------------------------------------------------------------
; FUNCTION DEF: dco_step
; DESCRIPTION:  regulates the DCO by changing the DCO value by 1 only
; REGISTER USE: R10, R11
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         01. Dec. 98
;-------------------------------------------------------------------------
dco_step
                mov.b   &DCOCTL,R11     ; read current DCO selection
                cmp.b   #dco_max,R11    ; should Rsel be increased ?
                jhs     inc_Rsel        ; yes
```

```
                cmp.b   #dco_min,R11    ; should Rsel be decreased ?
                jlo     dec_Rsel        ; yes
                mov     VZC_2delta,R10  ; read timer for 2 zero crosses
                cmp     #fset,R10
                jlo     inc_dco_step    ; increase DCO by one
                jeq     dco_step_end    ; do nothing
dec_dco_step    dec.b   &DCOCTL         ; decrease DCO by one
                ret
inc_dco_step    inc.b   &DCOCTL
dco_step_end    ret
inc_Rsel        mov.b   &BCSCTL1,R10    ; is max Rsel already selected?
                bic.b   #0f8h,R10
                cmp.b   #7,R10
                jge     dco_step_end    ; yes, cannot increase Rsel!
                inc.b   &BCSCTL1        ; Rsel + 1
                mov.b   #60h,&DCOCTL    ; center DCO
                ret
dec_Rsel        mov.b   &BCSCTL1,R10    ; is min Rsel already selected?
                bic.b   #0f8h,R10
                cmp.b   #0,R10
                jeq     dco_step_end    ; yes, cannot decrease Rsel!
                dec.b   &BCSCTL1        ; Rsel - 1
                mov.b   #60h,&DCOCTL    ; center DCO
                ret
;-------------------------------------------------------------------------
; FUNCTION DEF: Rsel_set
; DESCRIPTION:  initializes the RSEL in BCSCTL with the value defined in
;               the variable RSEL
; REGISTER USE: R14, R15
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         13. Nov. 98
;-------------------------------------------------------------------------
Rsel_set        mov.b   &BCSCTL1,R14
                mov.b   R14,R15
                bic.b   #0F8h,R15
                cmp.b   Rsel,R15
                jeq     Rsel_end
                jl      Rsel_inc
Rsel_dec        dec.b   R14
                mov.b   R14,&BCSCTL1
                jmp     Rsel_set
Rsel_inc        inc.b   R14
                mov.b   R14,&BCSCTL1
                jmp     Rsel_set
Rsel_end        ret
;-------------------------------------------------------------------------
; FUNCTION DEF: MPYS, MACS, MPYU, MACU
; DESCRIPTION:  16bit x 16 bit signed and unsigned multiply
; REGISTER USE: R4-R9
; CALLS:        -
; ORIGINATOR:   Anton Muehlhofer
; DATE:         13. Okt. 98
```

```
;-------------------------------------------------------------------------
                .newblock
MPYS            clr     IRACL
                clr     IRACM
MACS            tst     IROP1
                jge     $1
                sub     IROP2L,IRACM
$1              tst     IROP2L
                jge     MACU
                sub     IROP1,IRACM
                jmp     MACU

MPYU            clr     IRACL
                clr     IRACM
MACU            clr     IROP2M
                mov     #1,IRBT
$2              bit     IRBT,IROP1
                jz      $3
                add     IROP2L,IRACL
                addc    IROP2M,IRACM
$3              rla     IROP2L
                rlc     IROP2M
                rla     IRBT
                jnc     $2
                ret

;-------------------------------------------------------------------------
; Timer A Capture/Compare Interrupt Service Routine
;-------------------------------------------------------------------------
Int_TA_IV       add     &TAIV,PC ; read TA intrpt vector and clear int flag
                reti
                jmp     HCCR1
                jmp     HCCR2
Int_TA_end      reti
;-------------------------------------------------------------------------
; Voltage Zero Cross Interrupt Service Routine
; used by MSP430x11x
;-------------------------------------------------------------------------
HCCR2           push    R5
                mov     CCR2,R5                 ; meas voltage 1/2 period
sub     VZC_LastCap,R5          ; R5 = voltage half period
                mov     VZC_delta,VZC_2delta
                add     R5,VZC_2delta           ; update volt full period
                mov     R5,VZC_delta
                mov     CCR2,VZC_LastCap
                bis.b   #Task_Ovr,Status        ; initiate Task proceeding
Cap2_End        pop     R5
                reti
;-------------------------------------------------------------------------
; CCR1 Interrupt Service Routine     - unused -
;-------------------------------------------------------------------------
HCCR1           reti
;-------------------------------------------------------------------------
; unused Timer_A interrupt service routines
```

```
;-------------------------------------------------------------------------
Int_TA_CC0
               reti
;-------------------------------------------------------------------------
; all other interrupts
;-------------------------------------------------------------------------
Int_P1                                  ; Port1
Int_P2                                  ; Port2
Int_WDT_T                               ; Watchdog / Timer
               reti

;=========================================================================
; Interrupt vectors
;=========================================================================
                .sect   ”Int_Vect”,Ivecs
                .word   RESET           ; Port0, bit 2 to 7, n/a at 112
                .word   RESET           ; Basic Timer, n/a at 112
                .word   Int_P1          ; Port1
                .word   Int_P2          ; Port2
                .word   RESET           ; Timer Port, n/a at 112
                .word   RESET           ; no source
                .word   RESET           ; UART Transmit, n/a at 112
                .word   RESET           ; UART Receive, n/a at 112
                .word   Int_TA_IV       ; Timer A
                .word   Int_TA_CC0      ; Timer A
                .word   Int_WDT_T       ; Watchdog/Timer, Timer mode
                .word   RESET           ; no source
                .word   RESET           ; UART handler, n/a at 112
                .word   RESET           ; P0.0, n/a at 112
                .word   RESET           ; NMI, Osc. fault
                .word   RESET           ; POR, ext. Reset, Watchdog
                .end
```

# Appendix B MSP430x11x Port Definitions

```
;-----------------------------------------------------------------------
; File:         110.inc
; Originator:   Anton Muehlhofer
; Date:         01.01.1999
;-----------------------------------------------------------------------
; Control register address definitions
;-----------------------------------------------------------------------
IE1             .equ    0h
IFG1            .equ    02h
;-----------------------------------------------------------------------
; IE1 bit definitions
;-----------------------------------------------------------------------
WDTIE           .equ    01h             ; Watchdog interrupt enable
OFIE            .equ    02h             ; Oscillator fault intrpt enable
;-----------------------------------------------------------------------
; IFG1 bit definitions
;-----------------------------------------------------------------------
WDTIFG          .equ    01h             ; Watchdog interrupt flag
OFIFG           .equ    02h             ; Oscillator fault interrupt flag
NMIIFG          .equ    10h             ; Signal at RST/NMI pin
;-----------------------------------------------------------------------
; Status flag bit definitions
;-----------------------------------------------------------------------
GIE             .equ    08h
CPUOFF          .equ    10h
OSCOFF          .equ    20h
SCG0            .equ    40h
SCG1            .equ    80h
;-----------------------------------------------------------------------
; System Clock Control Register address definition
;-----------------------------------------------------------------------
DCOCTL          .equ    056h
BCSCTL1         .equ    057h
BCSCTL2         .equ    058h
;-----------------------------------------------------------------------
; BCSCTL1 bit definition
;-----------------------------------------------------------------------
XT2OFF          .equ    80h
XTS             .equ    40h
XT5V            .equ    08h
;-----------------------------------------------------------------------
; BCSCTL2 bit definition
;-----------------------------------------------------------------------
SELS            .equ    08h
DCOR            .equ    01h
SELM1           .equ    80h
SELM0           .equ    40h
;-----------------------------------------------------------------------
; Port 1 Control Register address definition
;-----------------------------------------------------------------------
P1IN            .equ    020h
```

```
P1OUT           .equ    021h
P1DIR           .equ    022h
P1IFG           .equ    023h
P1IES           .equ    024h
P1IE            .equ    025h
P1SEL           .equ    026h
;-------------------------------------------------------------------------
; Port 2 Control Register address definition
;-------------------------------------------------------------------------
P2IN            .equ    028h
P2OUT           .equ    029h
P2DIR           .equ    02Ah
P2IFG           .equ    02Bh
P2IES           .equ    02Ch
P2IE            .equ    02Dh
P2SEL           .equ    02Eh
;-------------------------------------------------------------------------
; Timer A Control Register address definition
;-------------------------------------------------------------------------
TAIV            .equ   12Eh
TACTL           .equ   160h
TAR             .equ   170h
CCTL0           .equ   162h
CCR0            .equ   172h
CCTL1           .equ   164h
CCR1            .equ   174h
CCTL2           .equ   166h
CCR2            .equ   176h
;-------------------------------------------------------------------------
; Timer A Control Register bit definition
;-------------------------------------------------------------------------
CAP             .equ   0100h
OUT             .equ   0004h
CCIFG           .equ   0001h
P0IN0           .equ    001h
CCIE            .equ   0010h
CCI             .equ   0008h

;-------------------------------------------------------------------------
; Watchdog Control Register address and bit definition
;-------------------------------------------------------------------------
WDTCTL          .equ   120h             ;watchdog control register address
WDTPW           .equ   5A00h            ;password for watchdog access
WDTCL           .equ   8h               ;bit position for watchdog reset
WDTHold         .equ   80h
```

## IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Copyright © 2000, Texas Instruments Incorporated