Application Report
SPRA497

Driving Two Three-Phase Inverters with Deadband Using the TMS320F240 DSP Controller

Iain Hunter *Digital Signal Processing Solutions*

Abstract

The Texas Instruments (TI™) TMS320F240 digital signal processor (DSP) controller contains a hardware deadband unit that automatically generates a deadband in the six PWM (pulse width modulation) signals from the full compare unit. This document describes how the DSP can use the other six available PWM signals to drive a second three-phase inverter with deadband.

Contents

Figures

Design Problem

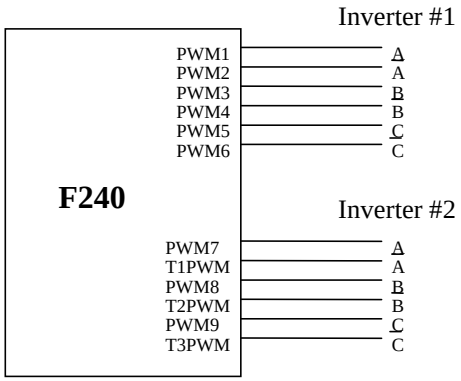
The TI TMS320F240 DSP controller contains a hardware deadband unit that automatically generates a deadband in the six PWM signals from the full compare unit. How can the DSP use the other six available PWM signals to drive a second three-phase inverter with deadband?

Solution

The shadow registers and interrupts available in the event manager make it possible to use software to generate a deadband with minimal overhead. The deadband is the time delay between switching off one power driver/transistor on a phase of an inverter and switching on the complementary driver . This ensures that any delay in switching off a device does not lead to a shoot-through short circuit that can damage the circuit when its partner is switched on.

Inverter #1 is connected to the full compare unit PWM (pulse-width modulation) outputs, PWM1-6. Inverter #2 uses the remaining six PWM signals. The simple compare outputs, PWM7-9, are connected to the high-side drivers and the individual compare outputs TxPWM are connected to the low-side drivers, as shown in Figure 1.

Figure 1. PWM Outputs Used to Drive Two Three-Phase Inverters



Configuring the F240 to generate the software deadband on the second inverter places the following limitations on the system design:

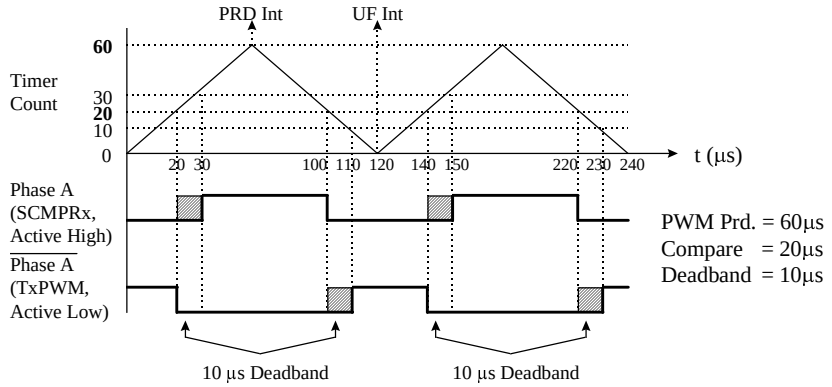
- Both inverters must have the same PWM period, as all three timers are synchronized.
- It is not possible to use the QEP inputs because all timers are in use for PWM generation. Therefore, speed feedback from motors must use the input capture units.

A second inverter without deadband can be implemented using the simple compare unit and complementing its three outputs, PWM7/8/9, to give !PWM7/8/9 as well. The deadband can be included by using the three independent PWM outputs (TxPWM) to give the complemented outputs. The simple compare outputs are configured as active high and the TxPWMs are configured as active low. Since we now have control over each of the six drivers' turn-on and turn-off times, it is easy to use software to delay turning on any driver by changing its compare register value.

Figure 2 illustrates the principle of deadband for a pair of complemented PWM signals (phaseA, !phaseA) for one leg of an inverter. The PWM period is 60 μs with symmetric PWM, the compare value is 20 μs, and the deadband is 10 μs. Figure 2 shows the following two relationships used to generate the software deadband.

- The first relationship is that when a driver is turned off at the expected time, given the compare value, the associated turn on is delayed by the deadband. To illustrate this, Phase A switches off at 100 μs and !Phase A switches off at 20 μs as expected, given the compare value. However, Phase A switches on at 30 μs rather than 20 μs and !Phase A switches on at 110 μs rather than 100 μs.
- The second relationship is that during the up-count phase of the timer, delaying the Phase A switch turn on by 10 ms corresponds to an increase in the compare value by 10 (to 30). Conversely, during the down count, delaying the !Phase A switch turn on corresponds to a reduction in the compare value by 10 (to 10).

Figure 2. PWM with Deadband



The software deadband implementation for the second inverter makes use of these two relationships and the following hardware features of the F240:

- Each of the six PWM signals has its own compare register, allowing each signal to be independently controlled.
- The shadow registers on each of the compare units allow the compare registers to be updated at exactly the right time with no synchronization problems. They can be updated at the start of both the up and down counts (count = 0 and count = PRD, respectively).
- The separate timer underflow (count = 0) and timer period (count = PRD) interrupts can be used to calculate the compare values to be loaded at the next loading time. The underflow interrupt can be used to calculate the compare values (Phase A = *x* and !Phase A = *x* – *d*), where *x* is the desired compare value and *d* is the deadband,

which will be loaded at count = PRD for the down count. Similarly, the period interrupt can be used to calculate the compare values for the next up count (Phase A = *x + d* and !Phase A = *x*).

To control both three-phase inverters, the event manager of the F240 must be configured as follows:

- All three timers are configured identically in continuous up/down mode with internal clock, with timers 2 and 3 set up to use the timer 1 period and start bit.
- All compare output pins are enabled, with PWM1/3/5/7/8/9 active high and PWM2/4/6/TxPWM active low.
- The full compare unit is set up for shadow reload on count = 0 only. The simple compare and individual timer compares for both count = 0 and count = period.
- The hardware deadband unit is set up for the full compare unit.
- Since all three timers are identical, the underflow and period interrupts are taken from different timers to allow the use of separate interrupt vectors. This eliminates the additional interrupt latency involved with a secondary vector table. The underflow interrupt is taken from timer 1 on INT2. The period interrupt is taken from timer 2 on INT3. To allow a PDPINT (power protection interrupt) as an additional, if occasional, source for INT2, a simple test on the interrupt vector (EVIVRA) is used in the interrupt service routine (ISR) rather than a secondary vector table.

The following code example assumes that both inverters are controlled by a control loop called every 5 PWM periods. The actual control function control() generates the six PWM compare values for the two inverters. Inverter #1 has the compare values inv1a/b/c and inverter #2 has inv2a/b/c. The code consists of a main C routine with C callable assembly language functions and ISRs.

Corresponding Code

```
#include <stdlib.h>
#include "c240_C.h" /* This is a C based version of c240.h */
      /* defining pointers to F240 registers */

/* Declare the variables defined in the assembly routines and used by C */
extern int int_cnt;

/* Prototype the functions defined in assembly and used by C */
extern void dis_watchdog(void);
extern void init_ev(void);

/* Prototype the C functions defined in this file */
void main(void);

/*****************************************************/
/* void main(void)           */
/* This is the main C routine which sets up the variables and    */
/* then enters an infinite loop, with the system functions being */
/* entirely interrupt driven.          */
/*****************************************************/
```

```
void main(void)
{int i=0;

  /* Set up F240 Peripherals */
    dis_watchdog();     /* Disable watchdog */
    init_ev();     /* Initialise the Event Manager */

  /* Set up system variables, both C and assembly */
  int_cnt=0;    /* Initialise isr counter */

    asm(" clrc INTM");   /* Enable Interrupts */

    for(;;);              /* Loop */
}
*****************************************************************
* This file contains 2 functions which are used to set up the C240 Event      *
* Manager and ADCs. The functions can be called by either C or assembly *
* functions because they are declared as void function(void) so no              *
* parameters are passed.                    *
*  _dis_watchdog - This function disables the watchdog timer if Vccp=+5V*
*  _init_ev      - This function initialises the Event Manager        *
*****************************************************************
  .include  c240.h

; Declare functions as global to allow C code to call them.
; The leading underscore '_' allows the functions and variables to
; be referenced by the same name (without the underscore) in C.
; However if they are referenced from assembly code the underscore is
; required
  .def  _dis_watchdog
  .def  _init_ev

  ; Define the constant for the PWM Period. In this case we have
  ; up down counting, so 500 * 100ns = 50us
  ; Define software DEADBAND to be 500ns, equal to 10 cycles
PWM_PERIOD   .equ 200   ; PWM Period *50ns = 12.8us
DEADBAND .equ 10
  .def  PWM_PERIOD
  .def  DEADBAND

  .text    ; Start the program code
***********************************************
* void dis_watchdog(void)           *
* Function to disable the watchdog timer      *
***********************************************
_dis_watchdog:
  ; Disable watchdog
  ldp #0E0h    ; Change DP to Event Manager, 07400h
  splk  #006Fh, WD_CNTL    ; Disable the Watchdog (WD) if
                                                ; Vccp=+5V
  splk  #05555h, WD_KEY  ; Write 55h then AAh to
                                                ; Watchdog Key register
  splk  #0AAAAh, WD_KEY ; to clear the WD counter
  ldp #0h
```

```
  ret

*****************************************************
* void init_ev(void)                 *
* Function to initialise the Event Manager for 2 invertors   *
* GPTimer 1 => Full PWM                                      *
* GP Timer 2&3, identical to GP Timer 1 with Simple Compare *
* and Individual compares all enabled.                       *
* Enable Timer 1==0 int. on INT2 and Timer 2==PRD on INT3 *
* All other pins are IO              *
*************************************************************
_init_ev:
  ; Set up SYSCLK and PLL for C24 EVM with 10MHz External Clk
  ldp #CKCR1/128
    splk  #00000010b,CKCR0 ; PLL disabled, CLKIN/2
          ; LPM0
          ; ACLK enabled
          ; SYSCLK 5MHz
  splk  #10110001b,CKCR1 ; 10MHz clk in for ACLK
          ; Do not divide PLL
          ; PLL ratio x2
    splk  #10000011b,CKCR0 ; PLL enabled x2
          ; LPM0
          ; ACLK enabled
          ; SYSCLK 10MHz
  ; Set up CLKOUT to be SYSCLK
    splk  #40C0h,SYSCR

    ; Clear all reset variables
    lacc  SYSSR
    and   #0FFh
    sacl    SYSSR

      ; Set up zero wait states for external memory
    lacc  #0008h
    sacl  *
    out *,WSGR

  ;Set IMR for INT2and INT3 and clear any Flags
  ldp #0h
  lacc  #0FFh
  sacl  IFR     ; Clear all core INT Flags
  lacc  #06h                ; Set core IMR for INT2/3.
  sacl  IMR

  ; Configure IO\function MUXing of pins
  ldp #OPCRA/128
    splk  #3F00h,OPCRA  ; Ports A all IO, Port B0-5 PWM
    splk  #00F1h,OPCRB  ; Port C as non IO function
        ; Note that XF/BIO~ require a 0 to select
        ; non-IO function

  ; Clear Event Manager IFR and set up the Event Manager IMR regs
  ldp #DP_EV
```

```
splk  #07FFh,IFRA
splk  #00FFh,IFRB
splk  #000Fh,IFRC

; Enable T1 Period Int and PDPINT in INT2 (Group A)
splk  #081h,IMRA

; Enable T2 Underflow Int (Group B)
splk  #0004h,IMRB

; Disable all  INT 4 (Group C) interrupts
splk  #0000h,IMRC

; Initialise CAPFIFO by clearing upper word
splk  #0FFh,CAPFIFO

; Initialise the Full PWM Unit based on GP Timer 1 for Invertor A
splk  #55h,GPTCON ; Enable TxCMP, all active low
splk  #05E8h,DBTCON; 500ns deadband on Full Compare Unit        ; Dead Band Timer = 5
      ; Prescaler = 2
      ; Enable all 3 DB timers
splk  #666h,ACTR ; Bits 15-12 not used, no space vector
    ; PWM compare actions
    ; PWM6/PWM5 - Active Low/Active High
    ; PWM4/PWM3 - Active Low/Active High
    ; PWM2/PWM1 - Active Low/Active High
splk  #02Ah,SACTR ; SCMPx all active high
splk  #32Fh,COMCON; FIRST enable PWM operation
      ; Reload Full Compare when T1CNT=0
      ; Disable Space Vector
      ; Reload Full Compare ACTR, T1CNT=0
      ; Enable Full & Simple Compare Outputs
      ; Simple Compare time base Timer 1.
      ; Reload Simple Compare, when
                                ;T1CNT=0 & PRD
      ; Reload Simple Compare Action when
                                ;T1CNT=0 & PRD
      ; Full Compare its in PWM Mode
splk  #832Fh,COMCON; THEN enable Compare operation

; Set up GP Timer 1
splk  #PWM_PERIOD,T1PER  ; Set T1 period
splk  #0,T1CNT
splk  #02806h,T1CON ; Ignore Emulation suspend
      ; Cont Up/Down Mode
      ; x/1 prescalar
      ; Use own TENABLE
      ; Disable Timer,enable later
      ; Internal Clock Source
      ; Reload Compare Register when
                                ; T1CNT=0 & PRD
      ; Enable Timer Compare operation
      ; Use own period
```

```
  ; Set up GP Timer 2 identically to GP Timer 1
  splk  #0h,T2CNT
  splk  #02887h,T2CON ; Ignore Emulation suspend
        ; Cont Up/Down Mode
        ; x/1 prescalar
        ; Use T1 ENABLE bit
        ; Disable Timer,enable later
        ; Internal Clock Source
        ; Reload Compare Register when
                                    ;T1CNT=0 & PRD
        ; Enable Timer Compare operation
        ; Use T1 Period

  ; Set up Timer 3 identically to Timer 1
  splk  #0,T3CNT
  splk  #0A887h,T3CON; Ignore Emulation suspend
        ; Cont Up/Down Mode
        ; x/1 prescalar
        ; Use T1 ENABLE bit
        ; Disable Timer,enable later
        ; Internal Clock Source
        ; Reload Compare Register when
                                    ; T1CNT=0 & PRD
        ; Enable Timer Compare operation
        ; Use T1 Period

  ; Enable/Start Timer 1 (and thus timers 2 & 3 as well)
  lacc  T1CON
  or  #40h   ; Set bit 6 to enable timer
  sacl  T1CON

  ret
  .end


*********************************************************************
* This file contains the assembly Interrupt Service Routines, with the context*
* saving required by assembly ISRs called in a C environment and/or while a*
* debugger is being used.                  *
* The functions are:                  *
*  _c_int2   - Group A Event Manager ISR, here only T1 Period INT and if   *
*       it happens PDPINT.                  *
*  _c_int3   - Group B Event Manager ISR,  here only T2 Underflow INT      *
*  _PHANTOM  - ISR to trap spurious other INTs              *
*********************************************************************
  .include  c240.h

  ; Declare this files functions as global
  .def     _c_int2
  .def     _c_int3
  .def     _PHANTOM

  ; Declare externally defined functions/constants as global
    .ref   PWM_PERIOD
  .ref   DEADBAND
```

```
  ; Declare a C callable variable to count the number of ISR calls
  .bss     _int_cnt,1
  .def     _int_cnt

  .bss     _inv1a,1 ; 3 locations to write desired Invertor A
  .bss     _inv1b,1 ; PWMs
  .bss     _inv1c,1
  .bss     _inv2a,1 ; 3 locations to write desired Invertor B
  .bss     _inv2b,1 ; PWMs
  .bss     _inv2c,1
  .def     _inv1a
  .def     _inv1b
  .def     _inv1c
  .def     _inv2a
  .def     _inv2b
  .def     _inv2c

  .text
;*****************************************************************
; c_int2: PWM interrupt handler                 *
; Every PWM period interrupt int_cnt is incremented. The Full Compare      *
; Units are updated for Invertor 1 to load at CNT=0. The TxCMP/SCMPRx  *
; units have DEADBAND applied to drive Invertor 2 to load at CNT=0       *
; If the PDPINT occurs then it is caught and dealt with          *
;*****************************************************************
_c_int2:
  ; Carry out the generic context saving for an ISR called in a
  ; C/debugger environment
  mar  *,AR1    ; Ensure ARP=AR1 and  ARB=previous ARP
  mar  *+ ; Increment stack for safety

  sst  #1,*+     ; Save the 2 status registers onto the stack
  sst  #0,*+
  popd *+ ; Pop the hardware stack and push onto sw stack
  sar AR0,*+     ; Save FP
  sar AR1,*    ; and SP onto the stack
  lar  AR0,*+    ; and set AR0 as the new FP

  ; Carry out the specific context saves for this ISR
      sach    *+       ; save H of ACC
          sacl    *+       ; save L of ACC

  ; Handle the Interrupt Flags
  ; The INT2 flag in IFR is automatically cleared by hardware when
          ;  the ISR is taken. However the specific Event Manager IFRA flag
          ; needs to be cleared. This is done by reading the appropriate
          ; Peripheral Vector Address Register
          LDP     #DP_EV
          LACC    IVRA         ; load interrupt ID, this resets IFRA flag
                               ; automatically

  ; In normal operation only T1PRD int will occur, although need to
          ; check for  PDPINT and deal with it
```

```
        sub#020h
        bcnd _pdpint,EQ

        ; Now start the actual ISR code on an T1PRD Int
        ; Increment the counter
                LDP     #_int_cnt
                LACC    _int_cnt
                ADD     #1
                SACL    _int_cnt

                ; In this Interrupt we update all the compare values for Invertors 1 &
                ;2  to take effect at CNT=0.
        ldp #DP_EV
        lar  AR3,#_inv1a
        mar  *,AR3
        lacc  *+
        sacl  CMPR1    ; Update Full Compare Unit
        lacc  *+
        sacl  CMPR2
        lacc  *+
        sacl  CMPR3

        ; For Invertor 2 need to add software deadband. Minimum deadband
                  ; must be ensured by control() to be less than
                  ;(PWM_PERIOD-DEADBAND)
        lacc  *+
        sacl  T1CMP    ; _inv2a -> T1CMPR
        add   #DEADBAND
        sacl  SCMPR1  ; _inv2a+DEADBAND -> SCMPR1
        lacc  *+
        sacl  T2CMP    ; _inv2b -> T2CMPR
        add   #DEADBAND
        sacl  SCMPR2  ; _inv2b+DEADBAND -> SCMPR2
        lacc  *+
        sacl  T3CMP    ; _inv2c -> T3CMPR
        add   #DEADBAND
        sacl  SCMPR3  ; _inv2c+DEADBAND -> SCMPR3

        ; Restore the specific context saves
        mar  *,AR1
        mar  *-
                lacl    *-          ; load L ACC
                add     *-, 16      ; load H ACC
                sbrk  1

        ; Restore the generic context save
          lar  AR0,*-     ; Restore old FP
          pshd *-
        lst  #0,*-
                lst     #1,*-
        clrc  INTM    ; INT2 is to be non-interruptible
                ret

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

```
; c_int3: PWM interrupt handler                *
; Every PWM underflow interrupt int_cnt is incremented. The TxCMP/      *
; SCMPRx units have DEADBAND applied to drive Invertor 2 to load at     *
; CNT=PRD and the main Controller routine is called every 5 PWM cycles  *
;***********************************************************************
_c_int3:
  ; Carry out the generic context saving for an ISR called in a
  ; C/debugger environment
  mar  *,AR1    ; Ensure ARP=AR1 and that ARB=previous ARP
  mar  *+ ; Increment stack for safety
  sst  #1,*+     ; Save the 2 status registers onto the stack
  sst  #0,*+
  popd *+ ; Pop the hardware stack and push onto sw stack
  sar AR0,*+ ; Save FP
  sar AR1,* ; and SP onto the stack
  lar  AR0,*+ ; and set AR0 as the new FP

  ; Carry out the specific context saves for this ISR
      sach    *+       ; save H of ACC
          sacl    *+       ; save L of ACC

  ; Handle the Interrupt Flags
  ; The INT3 flag in IFR is automatically cleared by hardware when
          ; the ISR is taken. However the specific Event Manager IFRB flag
          ; needs to be cleared. This is done by reading the appropriate
          ;Peripheral Vector Address Register
          LDP     #DP_EV
          LACC    IVRB         ; load interrupt ID, this resets IFRA flag
                               ; automatically

  ; Since INT3 contains the control routine which may take several
          ; PWM periods to run it needs to be interruptible by both itself and
          ;INT3 to keep track of the PWM UF and PRD ints.
  clrc   INTM

  ; In normal operation only T2UFINT int will occur
  ; Increment the counter
         LDP     #_int_cnt
         LACC    _int_cnt
         ADD     #1
         SACL    _int_cnt


  ; If this is the 5th symmetric PWM cycle (int_cnt=10) then call
          ; control()
          sub   #10
  bcnd not_control,LT

  ; Reset _int_cnt and call control routine
  splk  #0,_int_cnt

  ; Here call the main control routine that will update _inv1a -_inv2c
  ; It is essential that the control routine prevents the case of _inv2x
          ; being  less than DEADBAND or greater than PWMPERIOD-
```

```
            ;DEADBAND.
   Call  _control

not_control:
   ldp #DP_EV
   lar  AR3,#_inv2a
   mar  *,AR3

   ; Here we only update Invertor 2 values to be loaded at PRD
   ; For Invertor 2 need to add software deadband. Minimum deadband
            ; must be ensured by control routine to be _inv2x>DEADBAND
   lacc  *+
   sacl  SCMPR1  ; _inv2a -> SCMPR1
   sub#DEADBAND ;
   sacl  T1CMP    ; _inv2a-DEADBAND -> T1CMPR
   lacc  *+
   sacl  SCMPR2  ; _inv2b -> SCMPR2
   sub#DEADBAND ;
   sacl  T2CMP    ; _inv2b-DEADBAND -> T2CMPR
   lacc  *+
   sacl  SCMPR3  ; _inv2c -> SCMPR3
   sub#DEADBAND ;
   sacl  T3CMP    ; _inv2c-DEADBAND -> T3CMPR

   ; Restore the specific context saves
   mar  *,AR1
   mar  *-
            lacl       *-          ; load L ACC
            add        *-, 16      ; load H ACC
            sbrk 1

   ; Restore the generic context save
     lar  AR0,*-     ; Restore old FP
     pshd *-
   lst  #0,*-
            lst     #1,*-
            ret

;*********************************************************************
;  Dummy ISR to catch any spurious interrupts      *
;*********************************************************************
_PHANTOM b _PHANTOM

;*********************************
; Dummy routine to catch PDPINT  *
;*********************************
_pdpint  b    _pdpint

;*****************************************
; Dummy control routine to set compare values *
;*****************************************
_control:
   ldp #_inv1a
   splk  #10,_inv1a
```

```
  splk  #20,_inv1b
  splk  #30,_inv1c
  splk  #10,_inv2a
  splk  #20,_inv2b
  splk  #30,_inv2c
  ret
  .end


*************************************************************
* This file contains the complete interrupt vector table for the C240 *
* with all the possible vectors identified.          *
*************************************************************
  ; Define the ISRs declared elsewhere as global to allow this file
  ; "see" them
  .ref_c_int0
  .ref_c_int2
  .ref_c_int3
  .ref_PHANTOM

  ; Define the interrupt vector table to be a separate section called
  ; ".intvecs" as this makes it easier for the linker to place it
  ; in the correct program memory ie between 0h and 3Fh
          .sect  ".intvecs"

RSVECT    B    _c_int0            ; PM 0   Reset Vector      1
INT1      B    _PHANTOM           ; PM 2   Ext Int 1         4
INT2      B    _c_int2            ; PM 4   Ext Int 2         5
INT3      B    _c_int3            ; PM 6   Ext Int 3         6
INT4      B    _PHANTOM           ; PM 8   Ext Int 4         7
INT5      B    _PHANTOM           ; PM A   Ext Int 5         8
INT6      B    _PHANTOM           ; PM C   Ext Int 6         9
RESERVED  B    _PHANTOM           ; PM E   (Analysis Int)    10
SW_INT8   B    _PHANTOM           ; PM 10  User S/W int      -
SW_INT9   B    _PHANTOM           ; PM 12  User S/W int      -
SW_INT10  B    _PHANTOM           ; PM 14  User S/W int      -
SW_INT11  B    _PHANTOM           ; PM 16  User S/W int      -
SW_INT12  B    _PHANTOM           ; PM 18  User S/W int      -
SW_INT13  B    _PHANTOM           ; PM 1A  User S/W int      -
SW_INT14  B    _PHANTOM           ; PM 1C  User S/W int      -
SW_INT15  B    _PHANTOM           ; PM 1E  User S/W int      -
SW_INT16  B    _PHANTOM           ; PM 20  User S/W int      -
TRAP      B    _PHANTOM           ; PM 22  Trap vector       -
NMI       B    _PHANTOM           ; PM 24  Non maskable Int  3
EMU_TRAP  B    _PHANTOM           ; PM 26  Emulator Trap     2
SW_INT20  B    _PHANTOM           ; PM 28  User S/W int      -
SW_INT21  B    _PHANTOM           ; PM 2A  User S/W int      -
SW_INT22  B    _PHANTOM           ; PM 2C  User S/W int      -
SW_INT23  B    _PHANTOM           ; PM 2E  User S/W int      -

                  .end
```

INTERNET

TI Semiconductor Home Page
www.ti.com/sc

TI Distributors
www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas

Phone	+1(972) 644-5580
Fax	+1(972) 480-7800
Email	sc-infomaster@ti.com

Europe, Middle East, and Africa

Phone	
Deutsch	+49-(0) 8161 80 3311
English	+44-(0) 1604 66 3399
Español	+34-(0) 90 23 54 0 28
Francais	+33-(0) 1-30 70 11 64
Italiano	+33-(0) 1-30 70 11 67
Fax	+44-(0) 1604 66 33 34
Email	epic@ti.com

Japan

Phone	
International	+81-3-3457-0972
Domestic	0120-81-0026
Fax	
International	+81-3-3457-1259
Domestic	0120-81-0036
Email	pic-japan@ti.com

Asia

Phone	
International	+886-2-23786800
Domestic	
Australia	1-800-881-011
TI Number	-800-800-1450
China	10810
TI Number	-800-800-1450
Hong Kong	800-96-1111
TI Number	-800-800-1450
India	000-117
TI Number	-800-800-1450
Indonesia	001-801-10
TI Number	-800-800-1450
Korea	080-551-2804
Malaysia	1-800-800-011
TI Number	-800-800-1450
New Zealand	000-911
TI Number	-800-800-1450
Philippines	105-11
TI Number	-800-800-1450
Singapore	800-0111-111
TI Number	-800-800-1450
Taiwan	080-006800
Thailand	0019-991-1111
TI Number	-800-800-1450
Fax	886-2-2378-6808
Email	tiasia@ti.com

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 acknowledgement, 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.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (“CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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.