Application Report
SPRA501

Reduced Electromagnetic Interference (EMI) with the TMS320C24x DSP

Martin Staebler *Digital Signal Processoring Solutions*

This document discusses how a designer can reduce electromagnetic interference (EMI) in digital motor control applications using the Texas Instruments (TI™) TMS320C24x digital signal processor (DSP) controller.

This document contains a printed circuit board (PCB) layout of the TMS320C24x with reduced electromagnetic interference (EMI) and includes techniques for implementing code on the TMS320C24x to derive an optimum PWM pattern regarding EMI.

Contents

Figures

Tables

Examples

Design Problem

How can I reduce electromagnetic interference (EMI) in digital motor control applications using the TMS320C24x DSP controller?

Solution

This document contains a printed circuit board (PCB) layout of the TMS320C24x with reduced electromagnetic interference (EMI). Since the highest currents are typically found with the PWM-controlled H-bridge (an optimized PWM switching pattern), likewise *space vector PWM* and/or *wobbling* the PWM carrier frequency further reduce EMI. This document also describes techniques to implement code on the TMS320C24x to derive an optimum PWM pattern regarding EMI.

DSP Layout

The electromagnetic compatibility (EMC) of electronics circuits is to a great extent determined by the way the components are laid out and interconnected. Signal lines with their corresponding return line form an antenna, which is able to radiate electromagnetic energy, where the magnitude is determined by current amplitude, frequency and the geometrical area of the current loops. There are three typical sources for EMI:

- Power supply lines
- Signal lines carrying high frequency
- Oscillator circuit

Power Supply: Whenever a CMOS inverter is changing its output state, both complementary transistors are conducting for a short time. The result will be a considerable increase in supply current, which causes current spikes on the supply lines. These current spikes lead to a more or less direct route to the power supply lines, which have been found to be the most significant causes of EMI.

It is good practice to decouple the supply voltage close to the supply pins with a 100nF ceramic bypass capacitor. However, as shown in reference [1], the parasitic components of the circuit, such as the impedance of the package leads and the supply lines, form an effective antenna, where the bypass capacitor does not significantly reduce the current peaks and hence the radiated interference. To suppress these current spikes (at least on the supply lines) so that the spikes do not reach other parts, an improvement can be achieved by adding an inductive coil Lh (ferrite beat) between the blocking capacitor and the power supply line, as shown in Figure 1. Lh should be close to the IC, from where on the interference is to be suppressed.

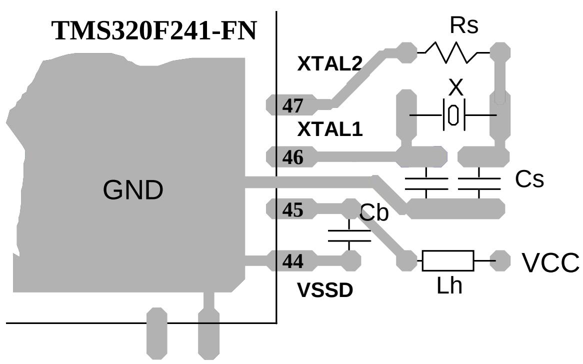
Signal Lines: Signal lines carrying high frequencies, e.g., the lower address lines, clock signals, serial ports, etc., are usually terminated by a CMOS input, providing a load of several 100k and 10p in parallel. Charging or discharging this load results in a high current peak. A possible way to reduce these currents is to connect a resistance of approximately 50Ω in serial with the output. Transmission line theory shows that this resistance has no negative influence on speed as long as the output resistance (internal + external resistance) is smaller or equal to the line impedance of typically 70-120Ω.

The second precaution is to make the antennas, (signal and corresponding return line) as small as possible. The most effective method is simply to keep the critical lines as short as possible, with the priority clock lines (1.), lower address lines (2.), other data lines. The TMS320C24x CPU clock, provided at CLKOUT1 after reset, can be switched off, which is recommended when it is not used in the application. When external memory is not used, one can pull-down/up the data lines to avoid drawing any (internal) current caused by a floating input. The address lines remain the last external address and hence do not have to be terminated.

Oscillator: The highest (continuous) frequencies in digital systems are usually found in the clock generator. When using a crystal in combination with the C24x internal oscillator, it is the aim to reduce high frequency currents, as well as the area enclosed by that current paths, to reduce EMI. The current at the resonant frequency of the crystal is very small due to the crystal's high resistance of several 100k at the resonant frequency. However, the output voltage of CMOS inverters is a square wave signal containing harmonics for which the crystal no longer represents a high resistance. This will result in significantly higher currents at the harmonics. A serial resistor can be added to reduce these current components.

The two bypass capacitors provide a low resistance at the oscillating frequency; hence there is a significant current flow Cs-X-Cs. To minimize radiation, this area should be as small as possible. Figure 1 shows a proposal for an external crystal connected to the TMS320F241. The serial resistor is in the range of 1k. A resistor in parallel to the crystal might by added according the manufacturers recommendation.

Figure 1. Proposal for a TMS320F241-FN Single Layer PCB Layout



EMI Reduction by Optimized PWM Pattern

When the PCB is finished, the 'C24x PWM unit can be set up to provide an optimized switching pattern to further minimize EMI. The following considerations are made for a 3-phase H-bridge with DC voltage [U_{DC}], which is driven by the 'C24x. For more information on the layout of the power electronics (IGBT), refer to [2].

PWM Mode

The three typical PWM modes (asymmetric, symmetric and space vector PWM) all have different influence on the EMI radiation. All PWM modes are supported by the TMS320C24x PWM unit.

With the **asymmetric PWM**, all three (e.g. lower) switches of the 3-phase H-bridge are turned on simultaneously and switched off according to the duty cycle. With the s**ymmetrical PWM** the turn-on/turn-off time is symmetrical with respect to half of the PWM period, hence the commutations of the three phases, do mostly not occur at the same time. This reduces EMI related to du/dt and di/dt by approximately 66% compared to the asymmetric PWM. For both modes, when using sine wave modulation, the minimum DC link voltage [U_{DC}] of the H-bridge, as function of the effective motor voltage [u_{rms}], is given by

$$U_{DC} \geq 2\sqrt{2} \cdot u_{rms,motor}$$

The **space vector PWM** is symmetrical with respect to the PWM period, too. However, since only two transistors are switched during one PWM period, the switching losses as well as the EMI radiation are reduced by 30% compared to the symmetric PWM. A second advantage is with

$$U_{DC} \geq \sqrt{3}\sqrt{2} \cdot u_{rms,motor}$$

the minimum DC link voltage is approximately 15% lower than with sinusoidal symmetrical PWM and hence also du/dt can be reduced further.

Table 1. Comparison between Asymmetric, Symmetric and Space Vector PWM

	Asymmetric	Symmetric	Space Vector
Commutations per PWM Period	6	6	4
Commutations simultaneously	3	1	1
Maximum motor voltage for U,DC = 310V	110Vrms	110Vrms	127Vrms

TMS320F240 Application Code: The following C code shows a typical setup for the TMS320F240 PWM unit. All relevant PWM registers (compare, period and output pin polarity) are shadowed and reloaded at a timer 1 underflow. Either space vector PWM mode is chosen (when the constant SPACE_VECTOR_PWM is defined as shown below) or symmetric PWM mode (when undefined). The on-chip **dead-band** unit for each of the 3-phases ensures that there is no overlap between the turn-on period of upper and lower switch, which would cause additional current spikes. The F240 PWM unit registers are declared in the C240.h header file, listed in the appendix.

Example 1. Code Listing of Symmetric/Space Vector PWM Initialization

```
/*-----------------------------------------------------------*/
/* Initialize Full Compare PWM Unit                           */
/*-----------------------------------------------------------*/
#define SPACE_VECTOR_PWM

T1CON = 0x2840;              /* up/down count in 50ns steps */
T1PR  = PWM_PERIOD;          /* PWM carrier frequency */
                             /* fpwm = 50ns * 2 * PWM_PERIOD */
#ifdef SPACE_VECTOR_PWM      /* SPACE VECTOR PWM */
   COMCON= 0x1207;           /* CMPRx,T1PR reload at Timer1=0 */
                             /* ACTR reload at Timer1=0 */
                             /* enable PWM1-6 outputs */
#else
   COMCON |= 0x0207          /* SYMMETRICAL PWM */
#endif
ACTR = 0x0666;               /* PWM 1,3,5 active high*/
                             /*     2,4,6 active low */
DBTCON = 0x14E0;             /* dead band = 1us */
COMCON |= 0x8000;            /* enable compare unit */
```

Wobbling the PWM Carrier Frequency

When EMI related to the typically constant PWM carrier frequency and its harmonics is too high, modulation of this frequency can be used to decrease EMI. Modulation strategies are e.g., triangle, random noise, etc. Figure 2 shows the spectrum and the pulsed output voltage for a fixed PWM carrier frequency of 20kHz, where the peaks of the carrier amplitude and its harmonics are 36 dB above ground noise.

Figure 3 demonstrates the result of wobbling the carrier frequency at 20kHz ± 2kHz, using a random noise to generate a spread spectrum. Compared to a fixed carrier, EMI is reduced by -12dB. A further reduction of -18dB is possible by a +/-4kHz modulation. A random noise generator requires only 6 CPU clock cycles.

Figure 2. FFT (10 Average) of a Fixed 20kHz PWM Carrier Frequency

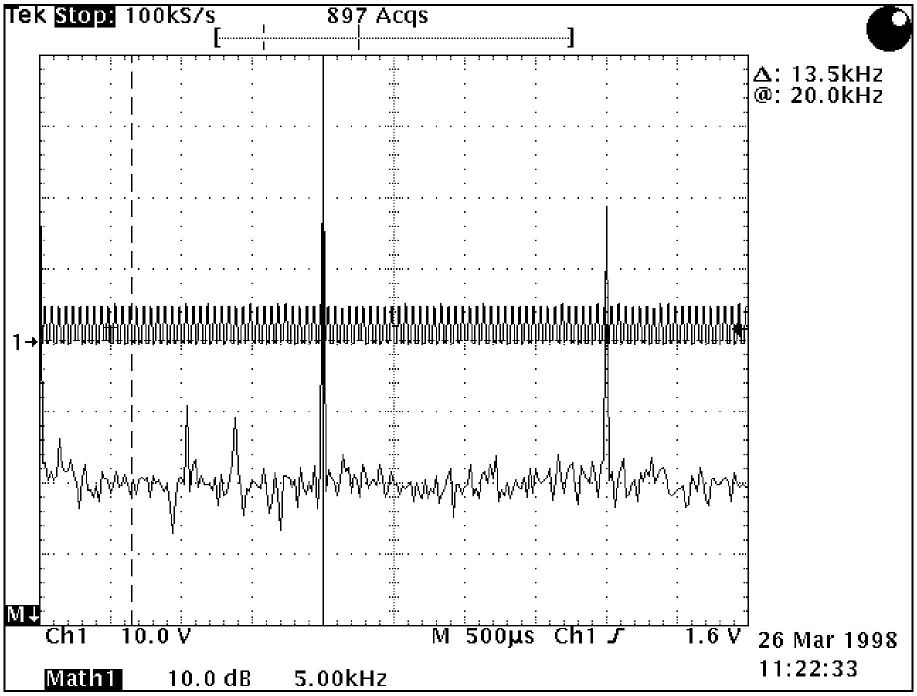
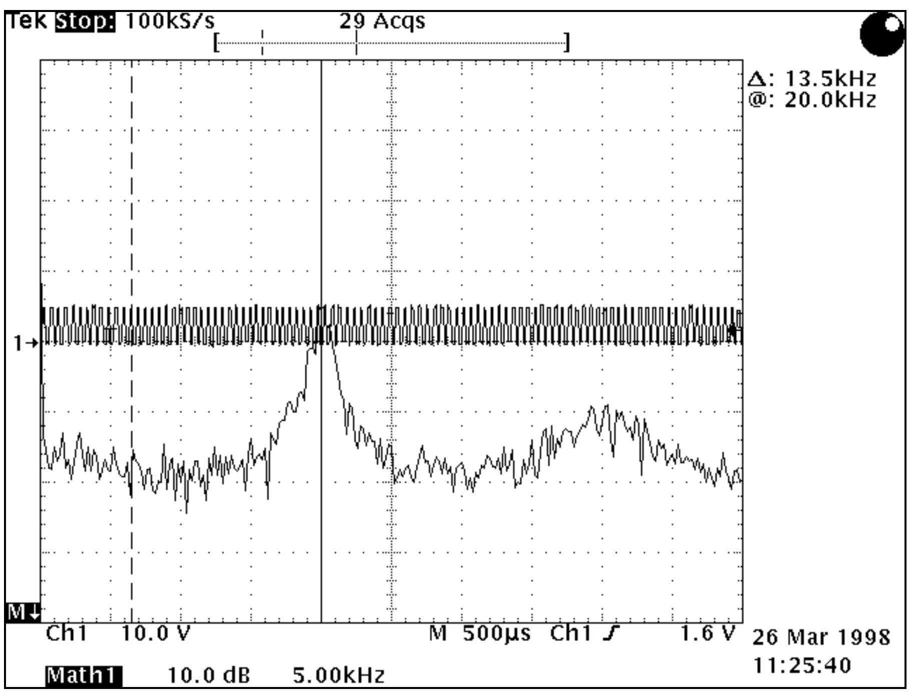


Figure 3. FFT (10 Averages) of a 20kHz with +/- 2kHz Random-Noise-Modulated PWM Carrier Frequency



TMS320F240 Application Code: In many applications the output voltages are fractional numbers (Q15) scaled to the maximum positive/negative output voltage. For a 3-phase output PWM, these are the phase voltages u(a), u(b), u(c) in case of a sinusoidal PWM or a space vector voltage, characterized by two 60 degree displaced positive fractional vectors u(x), u(x+60), the sector (1-6) to which this vector belongs, and the rotation direction of the vector. The sector is determined by u(x), in case of anti-clockwise rotation, and u(x+60), in case of clockwise rotation. For both PWM modes the fractional voltages u(a),…u(c) or u(x), u(x60) have to be multiplied with the PWM period, to get the corresponding PWM duty cycle. Hence, wobbling the PWM period does not add any overhead when updating the PWM compare values!

The C code in Example 2 shows how to include a ±10% PWM carrier modulation into the service routine for the current controller interrupt.

Example 2. Code Listing of PWM Carrier Modulation

```
/*-------------------------------------------------------*/
/* Current Controller Interrupt Service Routine           */
/*-------------------------------------------------------*/
/*  -> insert here your current controller, which         */
/*     calculates the fractional space vector voltage     */
/*     u_x, u_x60, sector, direction                      */

/*-----------------------------------*/
/* Wobble PWM (Timer 1) period by ±10% */
/*-----------------------------------*/
pwm_period = PWM_PERIOD + wobble_random(PWM_PERIOD/10);

/*------------------------*/
/* Update Space Vector PWM */
/*------------------------*/
SV_PWM_Update(pwm_period,u_x,u_x60,sector,direction);
```

The functions wobble_random(), SV_PWM_Update() are written in assembler to minimize runtime but provide a C compatible interface, hence can be called from C to allow a better readable software structure. Using inline assembler for the routines takes advantage of the C variable auto-initialization.

Example 3. Code Listing of wobble_random().

```
/*========================================================*/
/* int wobble_random(int wobble_amplitude)                */
/*========================================================*/
/* Function:      Random noise generator                  */
/* Arguments:     wobble_amplitude (Q15)                  */
/* Return value:  wobble_amplitude * random_number (Q15)  */
/*========================================================*/
/*--------------------------------------------------------*/
/* static variables --> map to one page                   */
/*--------------------------------------------------------*/
static  int     random_number = 21845;
static  int     INC = 13849;
static  int     MULT = 31821;

/*--------------------------------------------------------*/
/* int wobble_random(int)                                 */
/*--------------------------------------------------------*/
asm("           .text                                      ");
asm("           .globl _wobble_random                      ");
asm("_wobble_random:                                       ");
asm("     ;------------                                    ");
asm("     ;random noise                                    ");
asm("     ;------------                                    ");
asm("           setc    sxm                                ");
asm("           ldp     #_random_number                    ");
asm("           lt      _random_number  ;T = random_number ");
asm("           mpy     _MULT           ;P = MULT*T        ");
asm("           pac                     ;ACC = P           ");
asm("           add     _INC            ;ACC += INC        ");
asm("           sacl    _random_number  ;random_number=ACC ");
asm("    ;------------                                     ");
asm("    ;get argument                                     ");
asm("    ;------------                                     ");
asm("           mar     *-       ;AR1 -> argument          ");
asm("           mpy     *+       ;P=random_number*argument ");
asm("           pac              ;ACC = P = Q30 number     ");
asm("    ;------------                                     ");
asm("    ;return                                           ");
asm("    ;------------                                     ");
asm("           sach    *,1      ;push Q15 on stack        ");
asm("           lacc    *        ;pop Q15 into lower ACC   ");
asm("           ret                                        ");
```

Example 4. Code Listing Updating the Space Vector PWM, SV_PWM_Update()

```
/*============================================================*/
/* void SV_PWM_Update(unsigned pwm_period,int u_x,int u_x60,   */
/*                    int sector, int direction);              */
/*============================================================*/
/* Function:      Update T1PR (PWM carrier) and space vectors  */
/*                CMPR1,CMPR2, ACTR(DIR,SECTOR)                */
/*                                                             */
/*                if anti-CW (direction=0)                     */
/*                   CMPR1 = pwm_period*u_x/2                  */
/*                   CMPR2 = pwm_period*(u_x+u_x60)/2          */
/*                   ACTR.bit 14-12 = sector (u_x)             */
/*                   ACTR.bit 15 = 0                           */
/*                else (direction=1)                           */
/*                   CMPR1 = pwm_period/2 * u_x60              */
/*                   CMPR2 = pwm_period/2 * u_x+u_x60          */
/*                   ACTR.bit 14-12 = sector (u_x60)           */
/*                   ACTR.bit 15 = 1                           */
/* Arguments:     pwm_period                                   */
/*                u_x, u_x60 positive Q15                      */
/*                sector, direction                            */
/* Return value:  none                                         */
/*============================================================*/
asm("T1PR       .set    7403h                                  ");
asm("CMPR1      .set    7417h                                  ");
asm("CMPR2      .set    7418h                                  ");
asm("ACTR       .set    7413h                                  ");
asm("           .global _SV_PWM_Update                         ");
asm("           .text                                          ");
asm("_SV_PWM_Update:                                           ");
asm("                                                          ");
asm("    ;get arguments                                        ");
asm("    ;-------------                                        ");
asm("           sar     AR1,*                                  ");
asm("           lar     AR2,*,AR2                              ");
asm("           mar     *-              ;AR2 -> pwm_period     ");
asm("                                                          ");
asm("    ;CMPR2 = (u_x+u_x60)/2 * pwm_period (Q15)             ");
asm("    ;---------------------------------------              ");
asm("           ldp     #T1PR/128                              ");
asm("           lacc    *                                      ");
asm("           sacl    T1PR            ;T1PR = pwm_period     ");
asm("           lt      *-              ;T = pwm_period        ");
asm("           mpy     *-              ;P = u_x * pwm_period ");
asm("           pac                     ;ACC = P               ");
asm("           mpy     *-              ;P = u_x60*pwm_period ");
asm("           apac                    ;ACC = P               ");
asm("           sach    CMPR2           ;store to CMPR2        ");
asm("                                                          ");
asm("    ;update sector/direction (ACTR)                       ");
asm("    ;------------------------------                       ");
asm("           lacc    ACTR                                   ");
asm("           and     #0FFFh                                 ");
```

```
asm("           add     *-,12           ;bit 14-12 = sector   ");
asm("           add     *, 15           ;bit 15 = direction   ");
asm("           sacl    ACTR            ;update ACTR          ");
asm("                                                         ");
asm("    ;CMPR1 = u_x/2  * pwm_period (direction = 0)         ");
asm("    ;CMPR1 = u_x60/2* pwm_period (direction = 1)         ");
asm("    ;------------------------------------------          ");
asm("           lacc    *               ;ACC = direction      ");
asm("           bcnd    CW,NEQ          ;CW, if direction=1   ");
asm("ACW:       adrk    3                                     ");
asm("           mpy     *               ;P = u_x*pwm_period   ");
asm("           sph     CMPR1           ;store Q14 to CMPR2   ");
asm("           mar     *,AR1                                 ");
asm("           ret                                           ");
asm("                                                         ");
asm("CW:        adrk    2                                     ");
asm("           mpy     *               ;P = u_x60*pwm_period ");
asm("           sph    CMPR1            ;store Q14 to CMPR2   ");
asm("           mar     *,AR1                                 ");
asm("           ret                                           ");
```

Example 5. C24x PWM Register C Declaration (Macros), c240.h

```
/*----------------------------*/
/* PWM Register declaration */
/*----------------------------*/
#define T1CNT   *(volatile unsigned int*) 0x7401
#define T1CMPR  *(volatile unsigned int*) 0x7402
#define T1PR    *(volatile unsigned int*) 0x7403
#define T1CON   *(volatile unsigned int*) 0x7404

#define COMCON  *(volatile unsigned int*) 0x7411
#define ACTR    *(volatile unsigned int*) 0x7413
#define DBTCON  *(volatile unsigned int*) 0x7415
#define CMPR1   *(volatile unsigned int*) 0x7417
#define CMPR2   *(volatile unsigned int*) 0x7418
#define CMPR3   *(volatile unsigned int*) 0x7419
```

References

1) Haseloff, E.: *Printed Circuit Board Layout for Improved Electromagnetic Compatibility*, Texas Instruments 1996. (#EB215E)

2) Kirchenberger,U., Beierke, S.: *DSP and Power-Optimized Solution for AC Drives*, PCIM, Nürnberg, 1998.

INTERNET

www.ti.com

Register with TI&ME to build custom information pages and receive new product updates automatically via email.

TI Semiconductor Home Page
http://www.ti.com/sc

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

PRODUCT INFORMATION CENTERS

US TMS320

Hotline	(281) 274-2320
Fax	(281) 274-2324
BBS	(281) 274-2323
email	dsph@ti.com

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
Francais	+33-(0) 1-30 70 11 64
Italiano	+33-(0) 1-30 70 11 67
Fax	+33-(0) 1-30-70 10 32
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	+886-2-3786800
Domestic	
Australia	1-800-881-011

Asia (continued)

TI Number	-800-800-1450
China	10811
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