Freescale Semiconductor
Application Note

AN3225
Rev. 0, 2/2006

XGATE Library: PWM Driver

Generating flexible PWM signals on GPIO pins

by: Armin Winter, Field Applications, Wiesbaden
Daniel Malik, MCD Applications, East Kilbride
Steve McAslan, MCD Applications, East Kilbride

1 Introduction

There are several methods of generating pulse width modulated (PWM) signals. The simplest way is to generate the PWM signal directly from the dedicated PWM module of a microcontroller. Another option is to use the timer output compare feature, together with a software PWM implementation (as detailed in Freescale application notes AN2612 and AN1734).

A third option is to use the XGATE, available on the dual-core S12X family, to create software PWMs. This approach can create multiple PWM channels with no CPU loading, and so emulates a dedicated PWM hardware peripheral. This application note describes a flexible implementation that uses this approach and provides an example software project.

Table of Contents

2 Principle of PWM Generation Using XGATE

Generation of the PWM signal is done by software executed by the XGATE coprocessor, in conjunction with an internal periodic interrupt timer (PIT) channel. Each PWM period is divided into a number of time ticks, whose interval corresponds to the timeout period of the PIT. The number of ticks in each PWM period determines the resolution of the PWM signal.

As shown in Figure 1, the XGATE services the periodic timer interrupt. When entering this service routine at each interrupt, the XGATE decreases a counter variable providing the exact position within the period. Depending on this position and the desired duty cycle, the XGATE decides whether to set, clear, or leave unchanged, the signal on an output pin.

Several PWM channels with different duty cycles can be realized by using just one PIT time base. For improved EMC performance, output pin transition to high can be delayed individually for each channel.

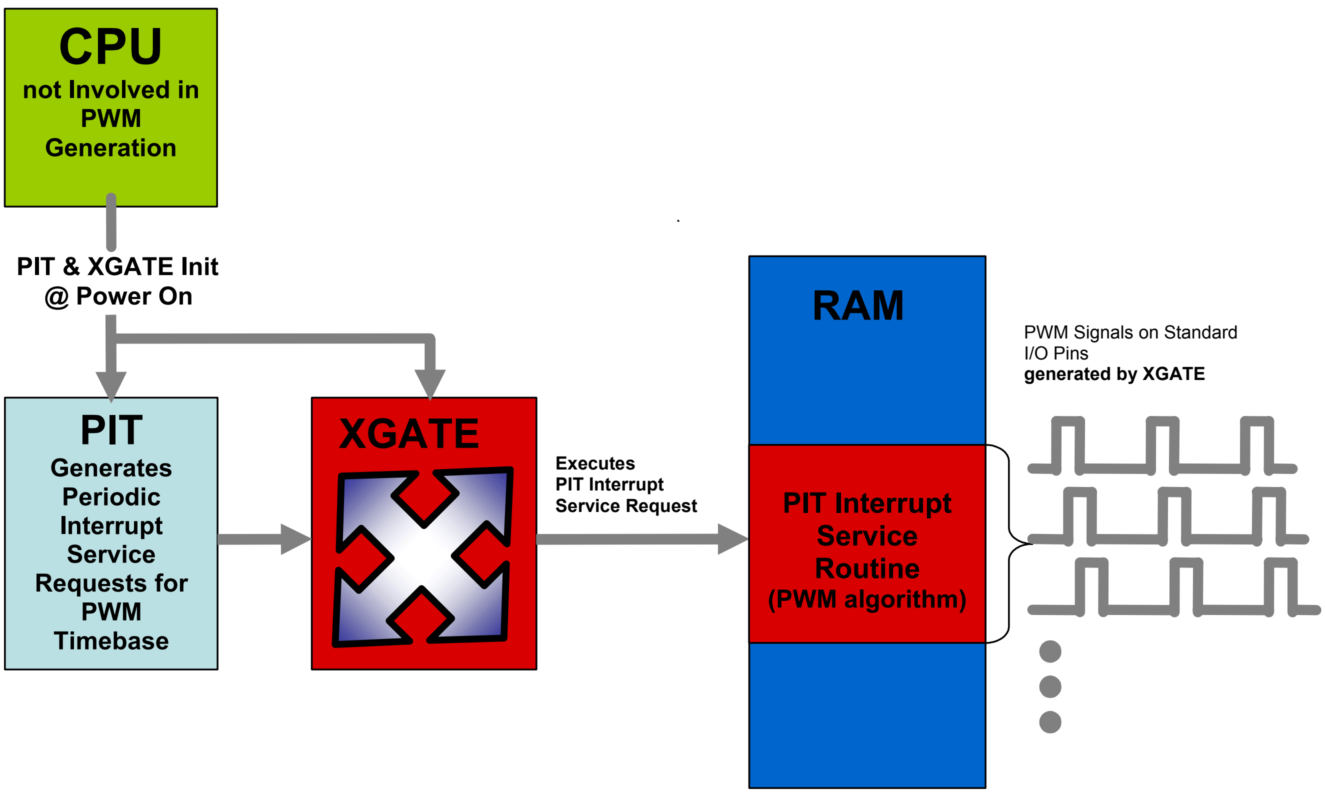

Figure 1. Principle of PWM Generation Using the XGATE

3 PWM Timing Example

To demonstrate the principle of the software PWM, we will create a PWM to drive an LED for a high quality lighting application. The requirement is for a frequency of 150 Hz at 0.5% resolution. In this case, the period of the signal will be 1/150 Hz = ~ 6.6 ms.

To achieve a resolution of 0.5%, the period must be divided by 200: 6.6 ms/200 = ~ 33 µs. In other words, 33 µs represents 0.5% of the period.

In this example, 33 µs represents a time tick. Therefore, the PIT interrupt timeout must be initialized to 33 µs.

In the case where a duty cycle of 12.5% is required, the corresponding output pin will be switched on for: 33 µs x 12.5 = 412.5 µs. Figure 2 shows an example of a PWM with these characteristics.

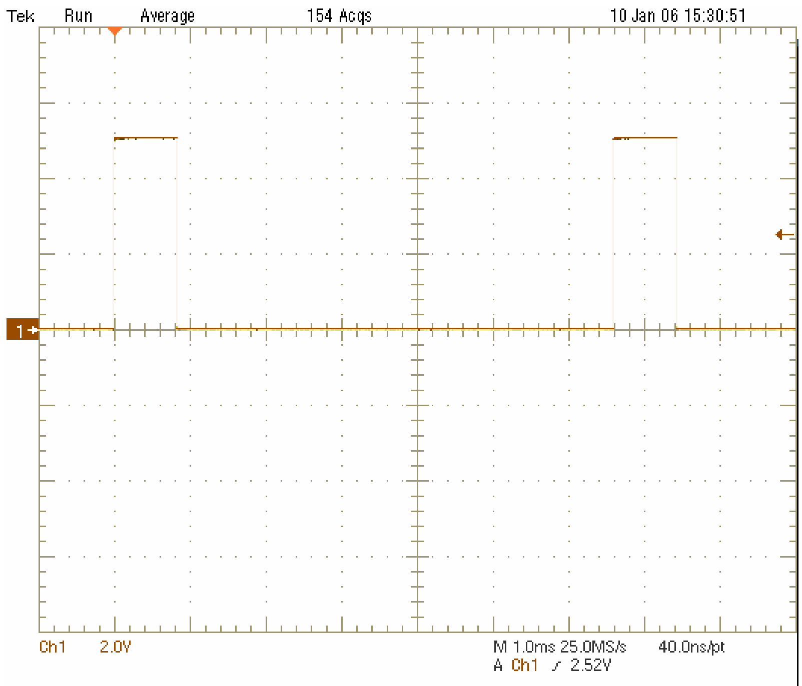

Figure 2. PWM Signal 150 Hz with a Duty Cycle of 12.5%

4 The Periodic Interrupt Timer

The PIT is an array of 24-bit timers. It can be used to trigger peripheral modules or raise periodic interrupts. On the MC9S12XDP152, there are four timers, each implemented as a modulus down-counter with independent timeout period.

The architecture of the PIT is shown in Figure 3. Each channel offers the following features.

- Timeout period selectable between 1 and 224 bus clock cycles (timeout equals m*n bus clock cycles with 1 <= m <= 256 and 1 <= n <= 65536)
- Individual enable for each timer channel
- Four timeout interrupts
- Four timeout trigger output signals available to trigger peripheral modules
- Start of timer channels can be aligned to each other

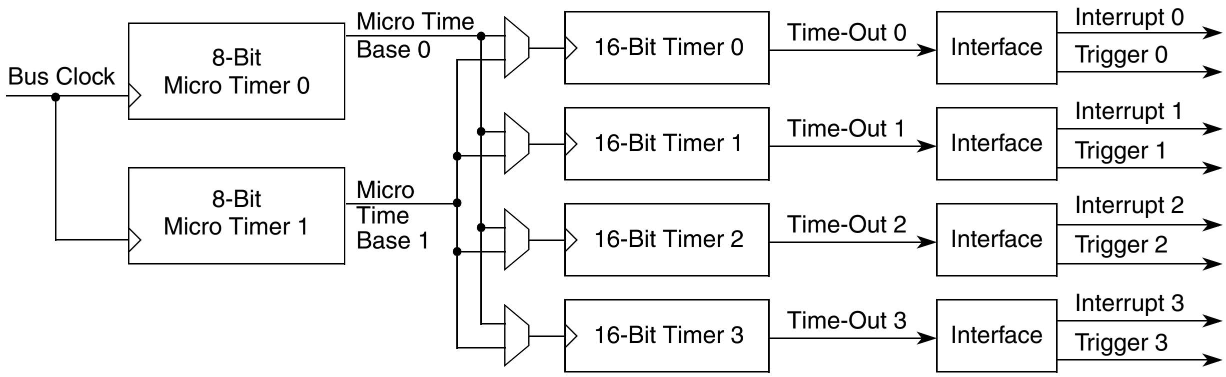

Figure 3. PIT Simplified Block Diagram

5 The XGATE

The XGATE is a programmable core that operates independently of the main CPU, has access to all of the S12X peripherals, and features an RISC instruction set.

To achieve full performance of the system, XGATE code should be downloaded into RAM. RAM protection can be set after downloading, thus preventing overwriting of XGATE code or CPU RAM variables. The XGATE vector base register must also be initialized appropriately.

XGATE initialization routines, including RAM download functions, are part of compiler vendor software libraries.

For information on how to configure the XGATE, refer to AN2685, “How to Configure and Use the XGATE on S12X Devices”.

6 PWM Driver Initialization

There are three steps to take when initializing the PWM Driver.

1. Configure the necessary I/O ports as outputs.
2. Configure the PIT channel to give the required PWM tick value.
3. Configure the PWM channels (period, duty cycle etc.) as required.

This initialization can be performed by the XGATE or by the CPU. If performed by the XGATE then, by convention, software interrupt 0 is used (see AN3145 for more information). This section describes the CPU initialization configuration, but the functionality is the same, if the XGATE is used instead.

6.1 I/O Initialization

Each PWM channel uses a standard I/O port to output the signal. Each of these must be configured as an output, and this task takes place, typically, after reset, but before PWM driver is initialized. In the example, the task is carried out by the CPU, in its initialization phase, and is found in the main.c file.

6.2 PIT Initialization

The PIT provides the time base for the PWM functionality, and so it must be configured with the correct timeout value. Additionally, the interrupt from the PIT module must be directed towards the XGATE, rather than to the CPU (which is the reset condition).

Figure 4 shows how code can initialize PIT channel 2 to give an interrupt every 33 µs. Firstly, the code enables the timer channel and connects channel 2 to microtimer 1, which is assigned a division ratio of 1. Then timer channel 2 is set to divide by 1320, which gives a timeout of 33 µs for a 40 MHz bus clock. Finally, the code forces a reload of the timer channel, and enables the module and the interrupts.

To allow the XGATE to service the interrupt, the corresponding RQST bit must be set in the interrupt controller. In this case, PIT channel 2 is assigned to XGATE and this is achieved by the code shown in Figure 5.

In the example, the PIT and interrupt configuration code is found in the main.c file.

```
void PIT2_Init(void)
{
 PIT.pitce.bit.pce2 = 1;                // Enable PIT channel 2
 PIT.pitmtld1.byte = 0;                 // Divide by 1
 PIT.pitmux.bit.pmux2 = 1;              // Assign PIT channel 2 to microtimer 1
 PIT.pitld2.word = 1320-1;              // 150Hz @ 0.5% -> 33us/25ns   = 1320

// Enable the PIT module and force reload of the micro counter
 PIT.pitcflmt.byte = PITE | PITFRZ | PFLMT1;
 PIT.pitflt.bit.pflt2 = 1;              // Force reload of counter 2
 PIT.pitinte.bit.pinte2 = 1;            // Enable interrupts from channel 2
}
```

Figure 4. Initialization of PIT

```
void SetIntPrio(char channel, char prio)
{
  Interrupt.int_cfaddr = (channel << 1) & 0xf0;
  Interrupt.int_cfdata[channel & 0x07].byte = prio;
}

SetIntPrio(0x3b, RQST|1);       //Assign PIT2 (channel $3B) to XGATE, priority 1
```

Figure 5. Implementation and Calling of SetIntPrio

6.3 PWM Configuration

The XGATE driver configuration is done within a structure. For each PWM channel, it includes

- a pointer to the port that the PWM signal is assigned to
- the period
- the position of the channel within the period
- the duty cycle for the PWM signal
- the bit-position mask for the I/O port pin.

Figure 7 shows an example where nine PWMs are configured across PORTA and PORTB. In the example the structure definition and configuration are found in the XGATE_XPWM.cxgate file. The C declaration of this structure is shown in Figure 6.

```
typedef struct {
 tU08 *port;
 tU08 period;
 tU08 cntr;
 tU08 duty;
 tU08 bit_mask;
} tPWMChDescr;
```

Figure 6. PWM Definition

```
tPWMChDescr PWM_Channels[]={
/* 1*/    {(unsigned char *)(&PORTA.byte),200,31,  2,0x01},
/* 2*/    {(unsigned char *)(&PORTA.byte),200,30,  1,0x02},
/* 3*/    {(unsigned char *)(&PORTA.byte),200,29,  0,0x04},
/* 4*/    {(unsigned char *)(&PORTA.byte),200,28,200,0x08},
/* 5*/    {(unsigned char *)(&PORTA.byte),200,27,199,0x10},
/* 6*/    {(unsigned char *)(&PORTA.byte),200,26,195,0x20},
/* 7*/    {(unsigned char *)(&PORTA.byte),200,25,190,0x40},
/* 8*/    {(unsigned char *)(&PORTA.byte),200,24,185,0x80},
/* 9*/    {(unsigned char *)(&PORTB.byte),200,23,180,0x01}
};
```

Figure 7. Example of PWM Configuration

7 PWM Driver Implementation

The flowchart in Figure 8 shows how the flexible interrupt service routine for PWM generation is implemented. In the example, it is serviced every 33 µs by XGATE and controls the status of the selected I/O (PWM) pins.

Each PWM channel has a counter variable (cntr), which contains the current position within the PWM period. Depending on the value of cntr and the desired duty cycle, the output pin is set, cleared, or left in its current state.

NOTE

The counter variable cntr can be set to any initial value (smaller than the period).

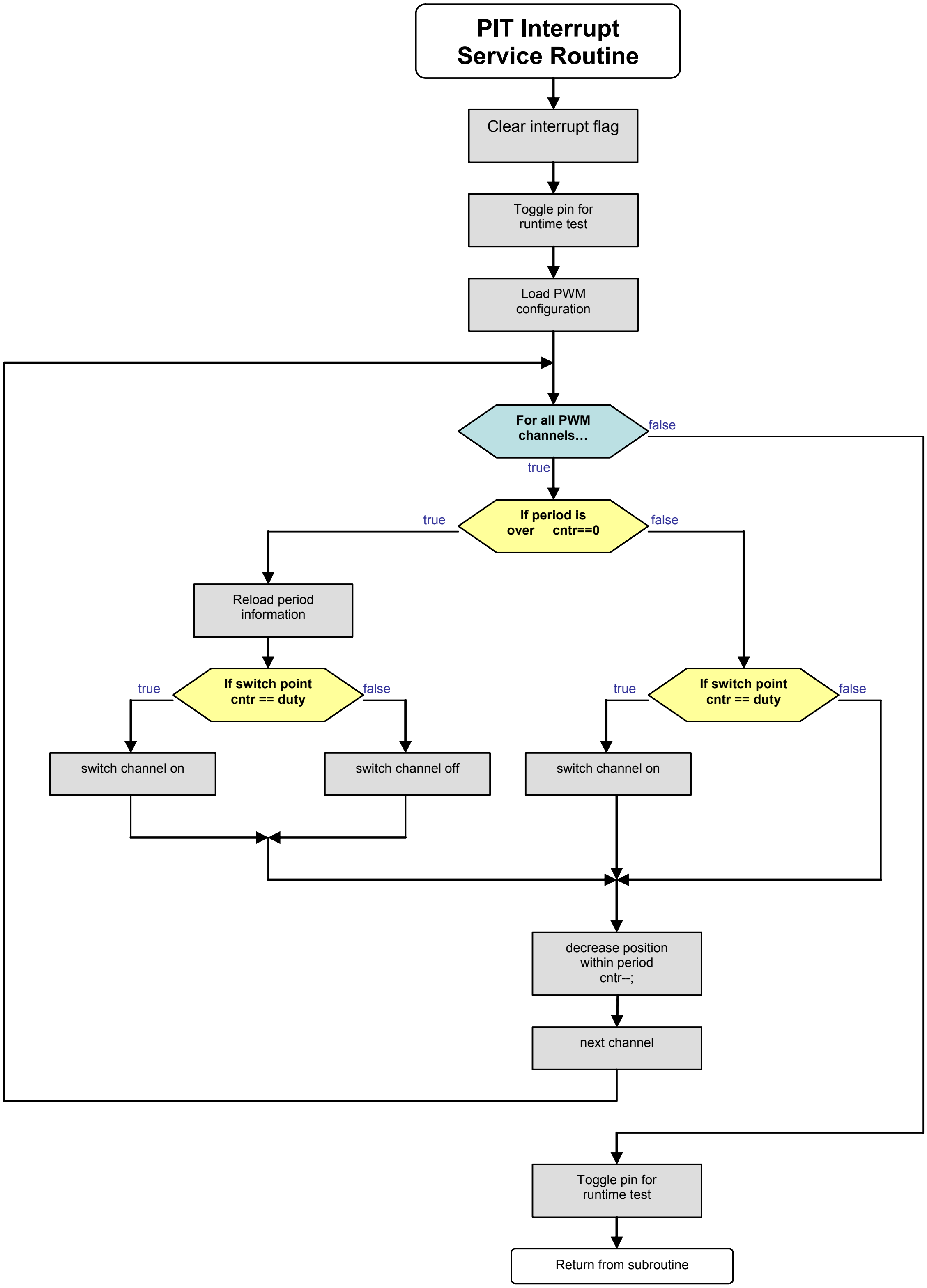

Figure 8. XGATE PWM Algorithm

8 Features of PWM Generation by XGATE

A key advantage of this approach is that the CPU is not involved in the PWM generation. The XGATE operates independently of the CPU, although the CPU still has full control of the PWM behavior. In fact, the XGATE implementation makes the PWM channels appear as a virtual PWM module.

The software implementation allows selectable switching delays, for excellent EMC behavior. Switching on all PWM channels at the same time has a negative effect on the EMC behavior of an electronic module. With the XGATE PWM implementation, it is possible to switch on one channel after the other, depending on the selected initialization values of the cntr variable. See Figure 9 for an example of four PWMs with switching offsets.

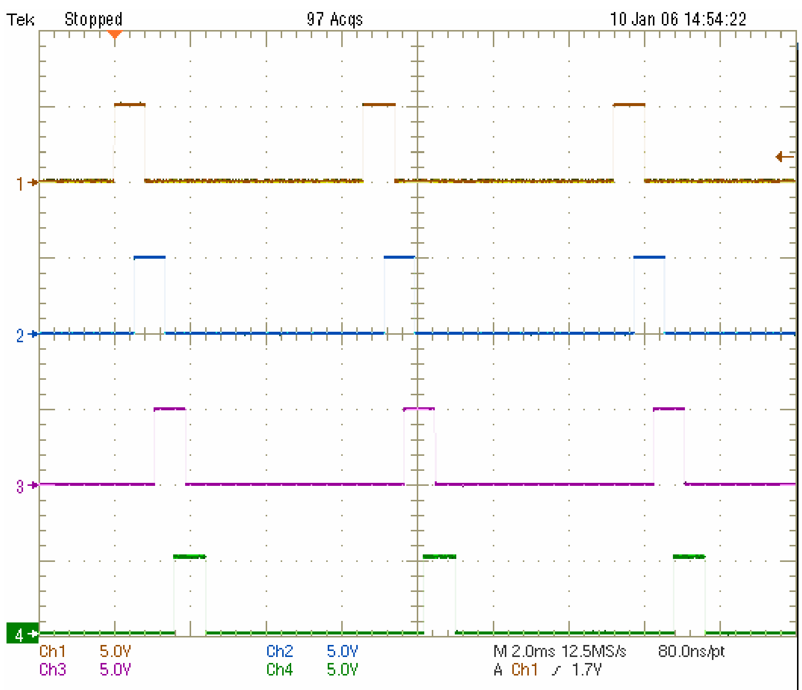

Figure 9. Four PWM channels with different switching points

The flexibility of the software is such that during EMC evaluation of the electronic module, the effect of altering the delay can be measured by simply downloading a new software file. The PIT timeout determines the smallest possible EMC delay. Figure 10 shows four PWM signals delayed by ~ 0.5 ms.

The inrush current of PWM loads can also be managed by these individual adjustments to cntr for each channel.

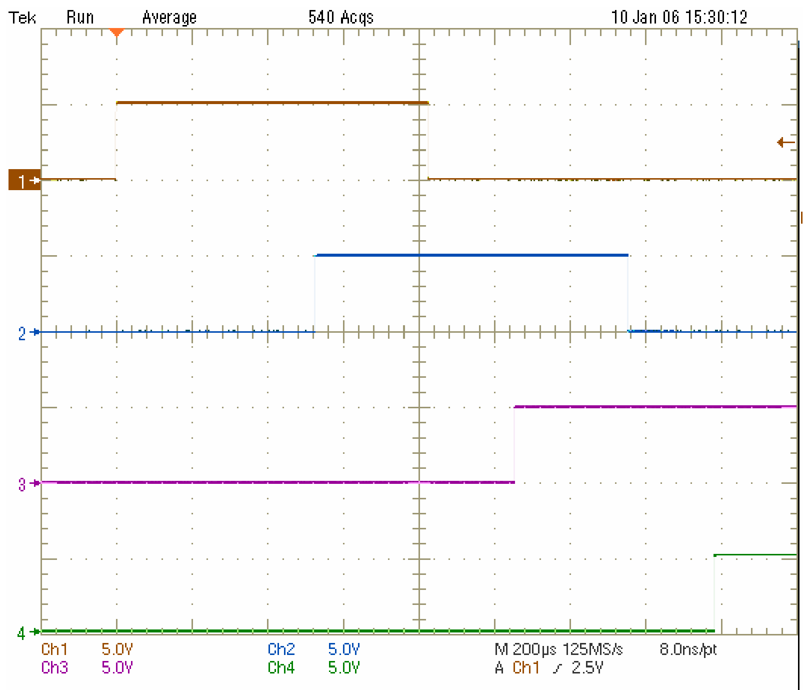

Figure 10. Switching Delay Between PWM Channels

Other advantages of the software approach include the fact that any I/O pin can be used for PWM operation, which allows great scalability and flexibility, in terms of the number of channels, their frequency, resolution, and I/O assignment.

The efficiency of the software running on the XGATE means that there is still enough XGATE performance headroom available for other tasks. Also, small code size and low RAM requirement mean that this virtual peripheral uses minimal MCU resources.

As discussed in Section 10, "Possible Enhancements and Limitations", easy and efficient implementation of load diagnostics is possible if the PWM is combined with the analog-to-digital converter (ATD).

9 Driver Performance

Table 1. Fixed Required Resources

Parameter	Value
Parameter	Value
Code size	90 bytes
Peripheral use	PIT channel, general purpose outputs

The code size shown in Table 1 does not vary for different configurations of the driver. Table 2 and Figure 11 indicate how execution time and XGATE load vary across the number of channels, frequency, and resolution. Memory footprint data has been extracted from the map file provided by CodeWarrior Development Studio for Freescale S12X version 4.5.

9.1 Notes on Performance Specification

The maximum execution time occurs when all PWMs are configured identically (same period and duty cycle). Therefore, the values presented here are higher than a typical application would require. The maximum latency for each configuration is calculated by subtracting the worst case execution time from the tick time for the PWM configuration. This determines the maximum latency for correct operation; however, the amount by which this latency varies determines the jitter on the PWM.

Table 2. Performance Considerations @ f_{bus} = 40 MHz

PWM Channels	Frequency (Hz)	Resolution (%)	Maximum Execution Time (µs)	Load (%)	Data Size (bytes)	Maximum Latency (µs)
16	80	1.0	10.65	4.59	98	114.35
16	80	0.5	10.65	9.11	98	51.85
16	100	1.0	10.65	5.74	98	89.35
16	100	0.5	10.65	11.39	98	39.35
16	150	1.0	10.65	8.69	98	55.35
16	150	0.5	10.65	17.25	98	22.35
24	80	1.0	15.85	6.79	146	109.15
24	80	0.5	15.85	13.47	146	46.65
24	100	1.0	15.85	8.48	146	84.15
24	100	0.5	15.85	16.83	146	34.15
24	150	1.0	15.85	12.85	146	50.15
24	150	0.5	15.85	25.50	146	17.15
32	80	1.0	21.05	8.96	194	103.95
32	80	0.5	21.05	17.82	194	41.45
32	100	1.0	21.05	11.20	194	78.95
32	100	0.5	21.05	22.28	194	28.95
32	150	1.0	21.05	16.97	194	44.95
32	150	0.5	21.05	33.75	194	11.95

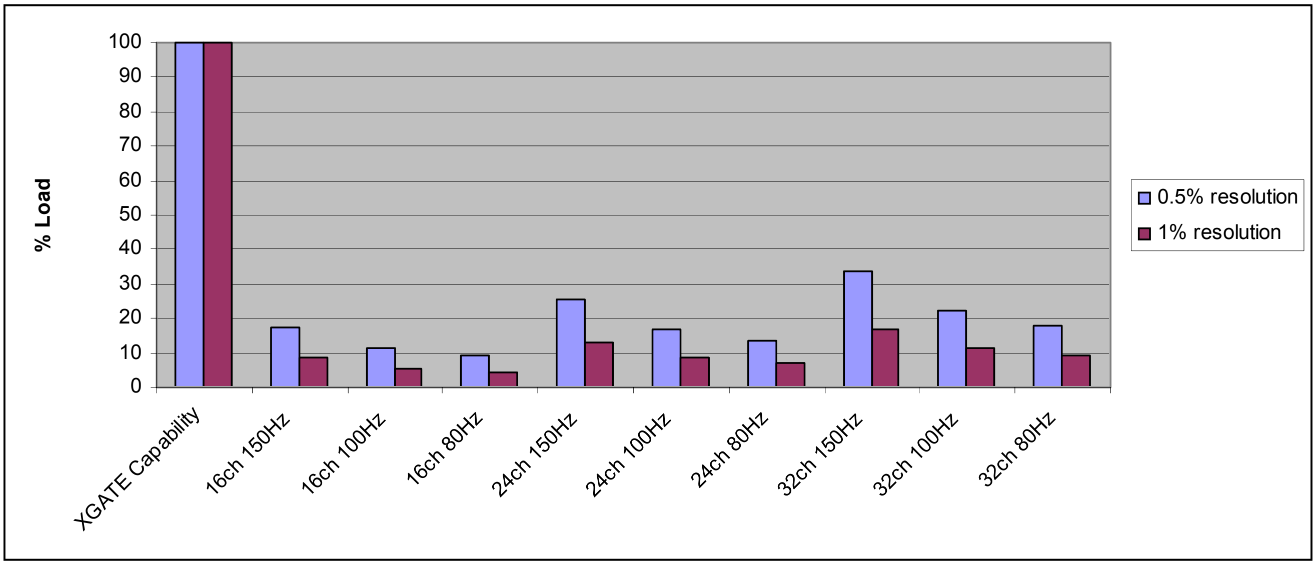

Figure 11. XGATE load Across Frequency and Resolution (40 MHz Bus)

10 Possible Enhancements and Limitations

The application might require a diagnostic function for the PWM loads.

Because the software provides the position of each channel within its period, a simple "if" command can launch an ATD measurement (e.g. if(current_chp->cntr==current_chp->atd_launch_position){ … }).

To determine a suitable value for atd_launch_position, the inrush current of the load must be considered. Good practice is to start the ATD measurement somewhere close to the end of the active phase of the period. This provides a very simple and efficient method for diagnosis of PWM loads.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN3225
Rev. 0, 2/2006