6 TIMER

Overview

The programmable interval timer can generate periodic interrupts based on multiples of the processor's cycle time. When enabled, a 16-bit count register is decremented every *n* cycles, where *n-1* is a scaling value stored in an 8-bit register. When the value of the count register reaches zero, an interrupt is generated and the count register is reloaded from a 16-bit period register.

The scaling feature of the timer allows the 16-bit counter to generate periodic interrupts over a wide range of periods. Given a processor cycle time of 80 ns, the timer can generate interrupts with periods of 80 ns up to 5.24 ms with a zero scale value. When scaling is used, time periods can range up to 1.34 seconds.

Timer interrupts can be masked, cleared and forced in software if desired. For additional information, refer to the section "Interrupts" in Chapter 3, "Program Control."

Timer Architecture

The timer includes two 16-bit registers, TCOUNT and TPERIOD and one 8-bit register, TSCALE. The extended Mode Control instruction enables and disables the timer by setting and clearing bit 5 in the Mode Status register, MSTAT. For a description of the Mode Control instructions, refer to the *ADSP-218x DSP Instruction Set Reference*. The timer registers, which are memory-mapped, are shown in Figure 6-1.

TCOUNT is the count register. When the timer is enabled, it is decremented as often as once every instruction cycle. When the counter reaches zero, an interrupt is generated. TCOUNT is then reloaded from the TPERIOD register and the count begins again.

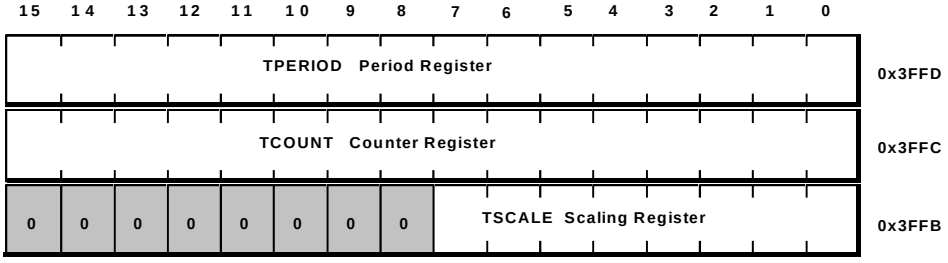

Figure 6-1. Timer Registers

TSCALE stores a scaling value that is one less than the number of cycles between decrements of TCOUNT. For example, if the value in TSCALE register is 0, the counter register decrements once every cycle. If the value in TSCALE is 1, the counter decrements once every 2 cycles. Figure 6-2 shows the timer block diagram.

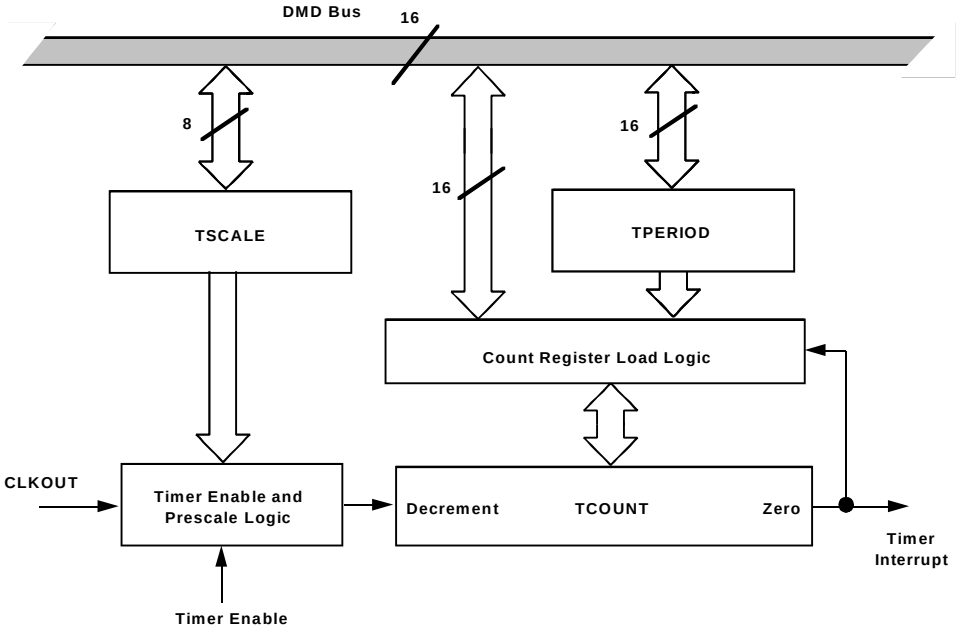

Figure 6-2. Timer Block Diagram

Resolution

TSCALE provides the capability to program longer time intervals between interrupts, extending the range of the 16-bit TCOUNT register. Table 6-1 shows the range and the relationship between period length and resolution for TPERIOD = maximum.

Table 6-1. Timer Range and Resolution

Cycle Time = 30 ns		
TSCALE	**Interrupt Every…**	**Resolution**
0	1.97 ms	30 ns
255	.5 s	7.65 µs

Timer Operation

Table 6-2 shows the effect of operating the timer with TPERIOD = 5, TSCALE = 1 and TCOUNT = 5. After the timer is enabled (cycle n–1) the counter begins. Because TSCALE is 1, TCOUNT is decremented on every other cycle. The reloading of TCOUNT and continuation of the counting occurs, as shown, during the interrupt service routine.

Table 6-2. Example Of Timer Operation

Cycle	**TCOUNT**	**Action**
n–4		TPERIOD loaded with 5
n–3		TSCALE loaded with 1
n–2		TCOUNT loaded with 5

Table 6-2. Example Of Timer Operation (Cont'd)

Cycle	TCOUNT	Action
n–1	5	ENA TIMER executed
n	5	Since TSCALE = 1, no decrement
n+1	5	Decrement TCOUNT
n+2	4	No decrement
n+3	4	Decrement TCOUNT
n+4	3	No decrement
n+5	3	Decrement TCOUNT
n+6	2	No decrement
n+7	2	Decrement TCOUNT
n+8	1	No decrement
n+9	1	Decrement TCOUNT
n+10	0	No decrement
n+11	0	Zero reached, interrupt occurs load TCOUNT from TPERIOD
n+12	5	No decrement
n+13	5	Decrement TCOUNT
n+14	4	No decrement
n+15	4	Decrement TCOUNT, etc.

One interrupt occurs every (TPERIOD +1) * (TSCALE +1) cycles. To set the first interrupt at a different time interval from subsequent interrupts, load TCOUNT with a different value from TPERIOD. The formula for the first interrupt is (TCOUNT+1) * (TSCALE+1).

If you write a new value to TSCALE or TCOUNT, the change is effective immediately. If you write a new value to TPERIOD, the change does not take effect until after TCOUNT is reloaded.

Enabling the Timer

This section tells you how to enable the timer and generate interrupts. It lists the steps you need to use and provides sample code (see Listing 6-1).

To enable the timer:

1. Set values for TCOUNT, TPERIOD, and TSCALE.
2. Set bit 0 in IMASK to enable interrupt.
3. Execute ENA TIMER instruction to start counting down (bit 5 in MSTAT register).

Listing 6-1. Sample Code for Enabling the Timer and Generating Interrupts

```
#include <def2181.h>

.SECTION/PM interrupts;

/* ------Interrupt vector table------ */
JUMP Start; NOP; NOP; NOP;              /* reset vector */
RTI; NOP; NOP; NOP;                     /* IRQ2 */
RTI; NOP; NOP; NOP;                     /* IRQL1 */
RTI; NOP; NOP; NOP;                     /* IRQL0 */
RTI; NOP; NOP; NOP;                     /* SPORT0 transmit */
RTI; NOP; NOP; NOP;                     /* SPORT0 receive */
RTI; NOP; NOP; NOP;                     /* IRQE */
```

```
RTI; NOP; NOP; NOP;                     /* BDMA */
RTI; NOP; NOP; NOP;                     /* SPORT1 transmit */
RTI; NOP; NOP; NOP;                     /* SPORT1 receive */
JUMP Interrupt_Hit; NOP; NOP; NOP;      /* timer */

.SECTION/PM Program;
Start:
/* set TSCALE to decrement every cycle */
AX0 = 0;
DM(Tscale_Reg) = AX0;
/* set TCOUNT to generate first interrupt at 50 cycles */
AX0 = 49;
DM(Tcount_Reg) = AX0;
/* set TPERIOD to reload TCOUNT with 99 at interrupt */
AX0 = 99;
DM(Tperiod_Reg) = AX0;
/* enable the timer interrupt */
IMASK = 0x1;
/* start the count down after executing this */
ENA TIMER;

/* ------Wait for timer interrupt------------ */
wait:   IDLE;
        Jump wait;

Interrupt_Hit:RTI;
```