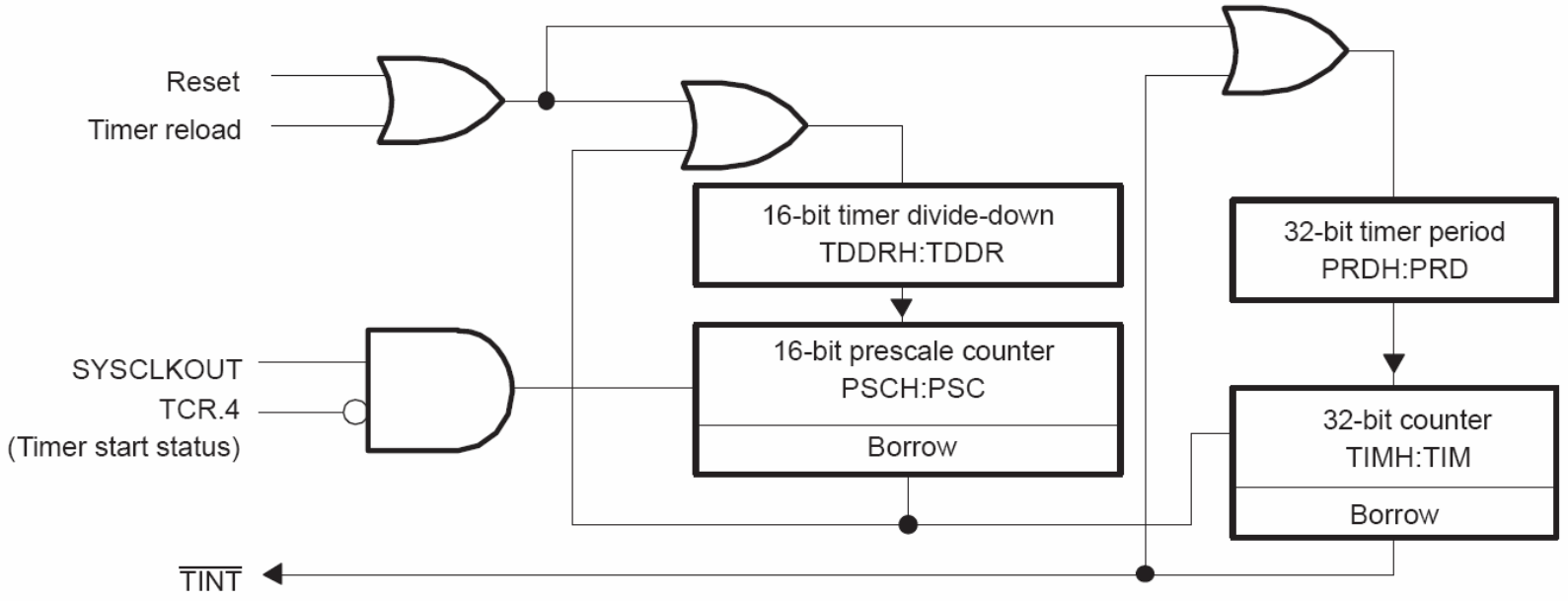
Micro Processor & Controller

Timer & State Machine

32 Bits Timers (0,1,2)

DSP Bios use Timer2

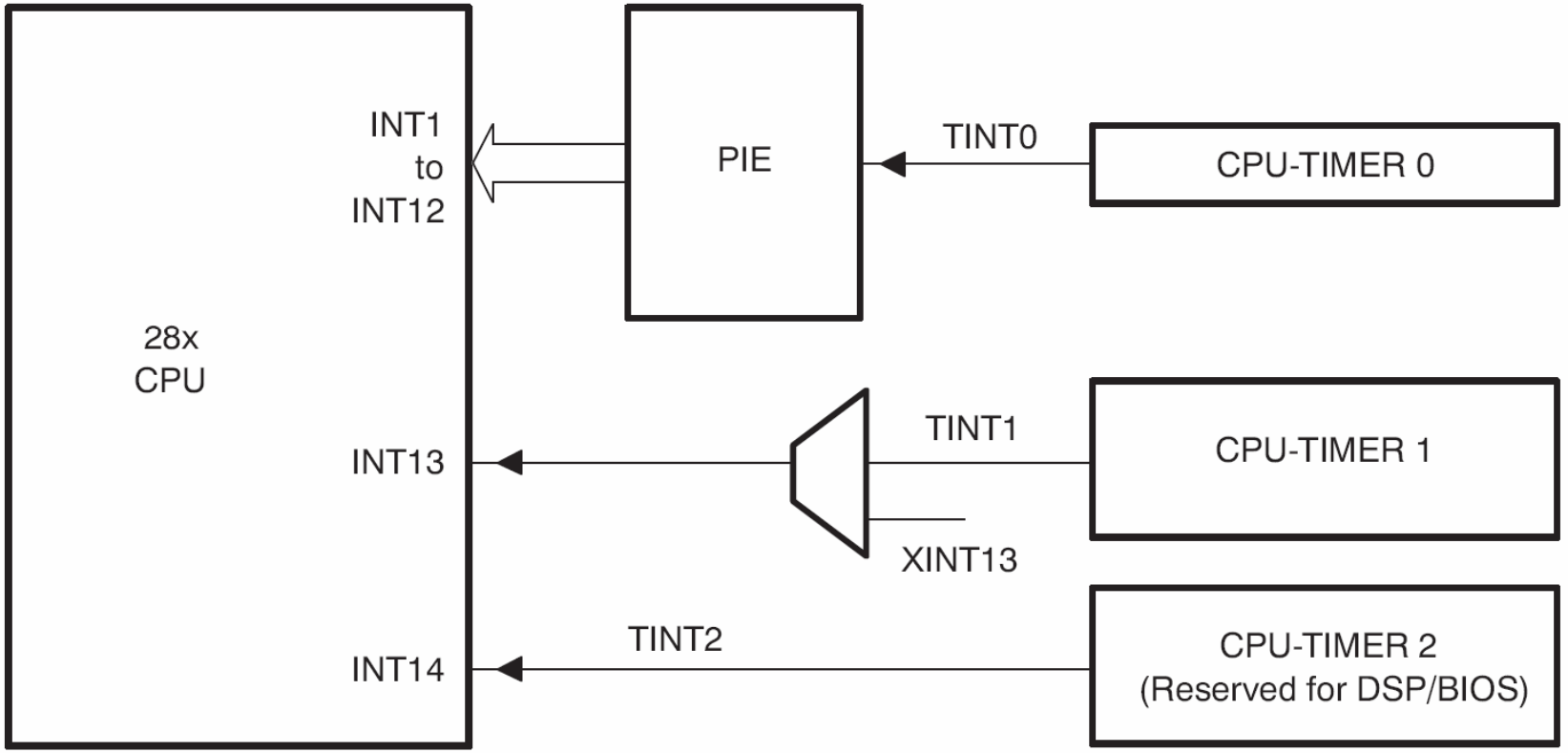
Figure 3-19. CPU-Timers



The general operation of the CPU-timer is as follows: The 32-bit counter register TIMH:TIM is loaded with the value in the period register PRDH:PRD. The counter register decrements at the SYSCLKOUT rate of the 28x. When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse. The registers listed in Table 3-18 are used to configure the timers.

Timers Interrupt

Figure 3-20. CPU-Timer Interrupts Signals and Output Signal



Timers Register Summary

Table 3-18. CPU-Timers 0, 1, 2 Configuration and Control Registers

Name	Address	Size (x16)	Description	Bit Description
TIMER0TIM	0x0C00	1	CPU-Timer 0, Counter Register	Figure 3-21
TIMER0TIMH	0x0C01	1	CPU-Timer 0, Counter Register High	Figure 3-22
TIMER0PRD	0x0C02	1	CPU-Timer 0, Period Register	Figure 3-23
TIMER0PRDH	0x0C03	1	CPU-Timer 0, Period Register High	Figure 3-24
TIMER0TCR	0x0C04	1	CPU-Timer 0, Control Register	Figure 3-25
Reserved	0x0C05	1		
TIMER0TPR	0x0C06	1	CPU-Timer 0, Prescale Register	Figure 3-26
TIMER0TPRH	0x0C07	1	CPU-Timer 0, Prescale Register High	Figure 3-27
TIMER1TIM	0x0C08	1	CPU-Timer 1, Counter Register	Figure 3-21
TIMER1TIMH	0x0C09	1	CPU-Timer 1, Counter Register High	Figure 3-22
TIMER1PRD	0x0C0A	1	CPU-Timer 1, Period Register	Figure 3-23
TIMER1PRDH	0x0C0B	1	CPU-Timer 1, Period Register High	Figure 3-24
TIMER1TCR	0x0C0C	1	CPU-Timer 1, Control Register	Figure 3-25
Reserved	0x0C0D	1		
TIMER1TPR	0x0C0E	1	CPU-Timer 1, Prescale Register	Figure 3-26
TIMER1TPRH	0x0C0F	1	CPU-Timer 1, Prescale Register High	Figure 3-27
TIMER2TIM	0x0C10	1	CPU-Timer 2, Counter Register	Figure 3-21
TIMER2TIMH	0x0C11	1	CPU-Timer 2, Counter Register High	Figure 3-22
TIMER2PRD	0x0C12	1	CPU-Timer 2, Period Register	Figure 3-23
TIMER2PRDH	0x0C13	1	CPU-Timer 2, Period Register High	Figure 3-24
TIMER2TCR	0x0C14	1	CPU-Timer 2, Control Register	Figure 3-25
Reserved	0x0C15	1		
TIMER2TPR	0x0C16	1	CPU-Timer 2, Prescale Register	Figure 3-26
TIMER2TPRH	0x0C17	1	CPU-Timer 2, Prescale Register High	Figure 3-27

Timer Registers Details

Figure 3-21. TIMERxTIM Register (x = 1, 2, 3)

15		0
	TIM	
	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 3-19. TIMERxTIM Register Field Descriptions

Bits	Field	Description
15-0	TIM	CPU-Timer Counter Registers (TIMH:TIM): The TIM register holds the low 16 bits of the current 32-bit count of the timer. The TIMH register holds the high 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDRH:TDDR+1) clock cycles, where TDDRH:TDDR is the timer prescale divide-down value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers. The timer interrupt (TINT) signal is generated.

Figure 3-22. TIMERxTIMH Register (x = 1, 2, 3)

15		0
	TIMH	
	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 3-20. TIMERxTIMH Register Field Descriptions

Bits	Field	Description
15-0	TIMH	See description for TIMERxTIM.

Figure 3-23. TIMERxPRD Register (x = 1, 2, 3)

15		0
	PRD	
	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 3-21. TIMERxPRD Register Field Descriptions

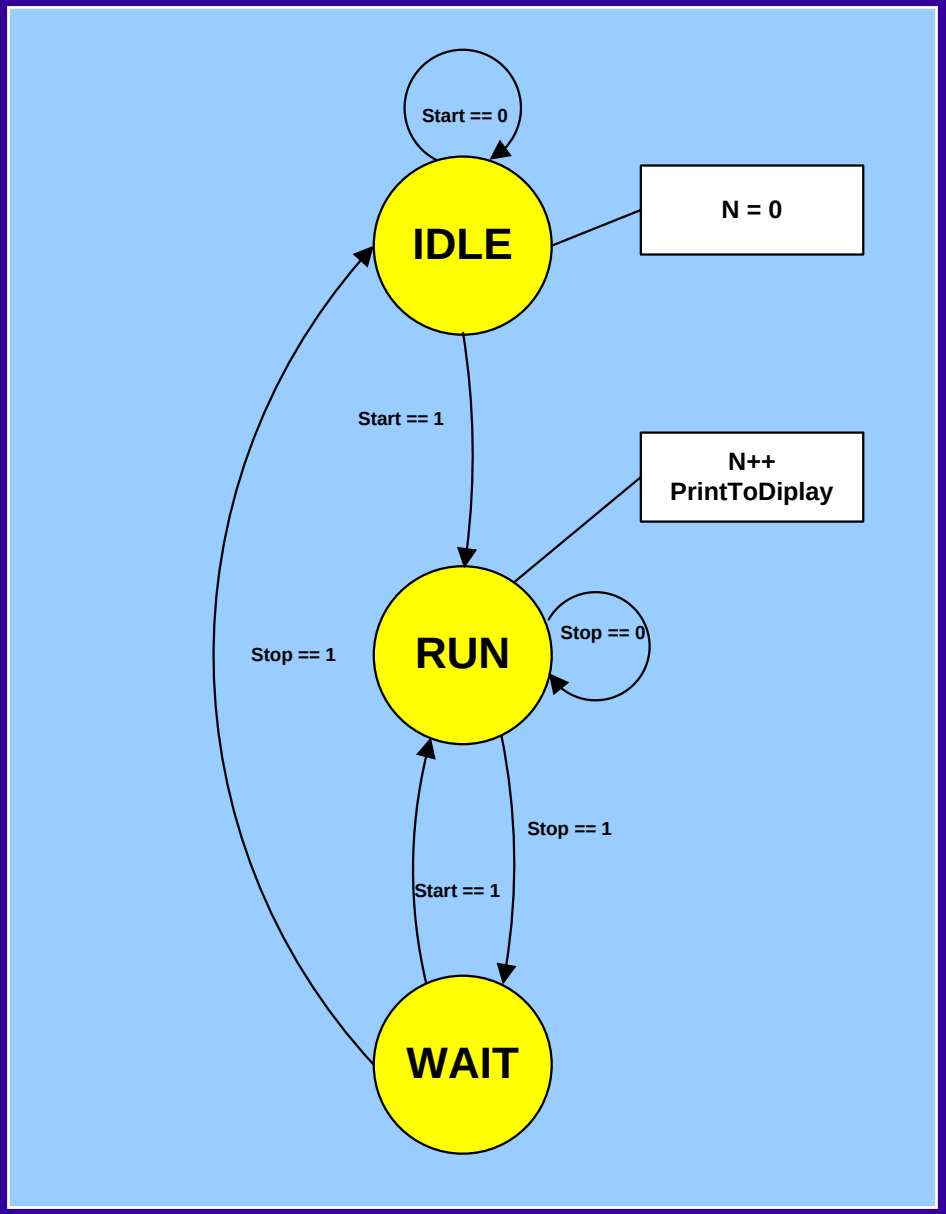
Bits	Field	Description
15-0	PRD	CPU-Timer Period Registers (PRDH:PRD): The PRD register holds the low 16 bits of the 32-bit period. The PRDH register holds the high 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR).

Figure 3-24. TIMERxPRDH Register (x = 1, 2, 3)

15		0
	PRDH	
	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

State Machine Model



State Machine - C Implementation

```
typedef enum {IDLE, RUN, WAIT} StateMachineType;

int MyCounter = 0;

void RunStateMachine(int Start, int Stop)
{
static StateMachineType MyStatMachine;

    switch (MyStatMachine)
        {
        case IDLE:
            if (Start == 1)
                MyStatMachine = RUN;
            MyCounter = 0;
            break;

        case RUN:
            if (Stop == 1)
                MyStatMachine = WAIT;
            MyCounter++;
            // Update the display with the new value
            break;

        case WAIT:
            if (Start == 1)
                MyStatMachine = RUN;
            else if (Stop == 1)
                MyStatMachine = IDLE;
            break;
        }
}
```