Interrupt Generation Using the AT91 Timer/Counter

AT91 ARM® Thumb® Microcontroller

Application Note

Introduction

This application note describes how to generate an Interrupt by using the Timer/Counter (TC) in the AT91 series of microcontrollers.

Timer/Counter Overview

The AT91 series features a Timer/Counter block, which includes three identical 16-bit timer counter channels. Each channel can be independently programmed, through its two operating modes, to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing, pulse width modulation and interrupt generation.

Each Timer Counter channel has 3 external clock inputs, 5 internal clock inputs, and 2 multi purpose input/output signals, which can be configured by the user. Each channel drives an internal interrupt signal, which can be programmed to generate processor interrupts via the Advanced Interrupt Controller (AIC). The three Timer Counter channels are independent and identical in operation. Each Timer Counter channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the bit COVFS in TCx_SR (Status Register) is set.

The current value of the counter is accessible in real-time by reading TCx_CV. A trigger can reset the counter. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

Rev. 2683A-ATARM–01/03

Operating Modes

Each Timer Counter channel can operate independently in two different modes:

- Capture Mode allows measurement on signals
- Waveform Mode allows wave generation

The Timer Counter Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register (TCx_CMR). In Capture Mode, TIOA and TIOB are configured as inputs. In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

Common Triggers

The following triggers are common to both operating modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TCx_CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in TCx_CMR.

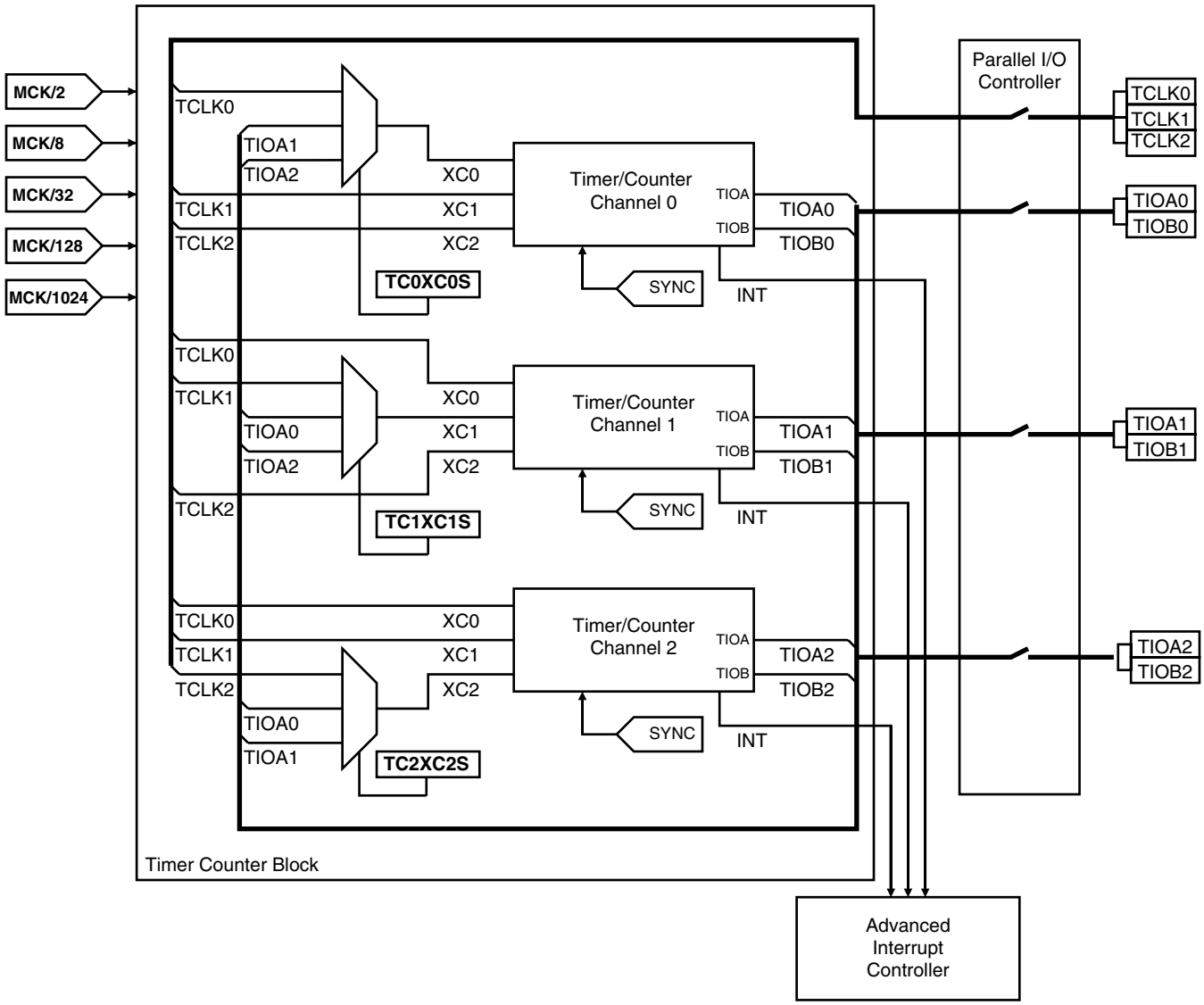
External Trigger

The Timer Counter channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting ENETRG in TCx_CMR.

If an external trigger is used, the duration of the pulses must be longer than the system clock (MCK) period in order to be detected.

Timer/Counter Block Diagram

Figure 1. Timer/Counter Block Diagram



Clock Source

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: MCK/2, MCK/8, MCK/32, MCK/128, MCK/1024
- External clock signals: XC0, XC1 or XC2

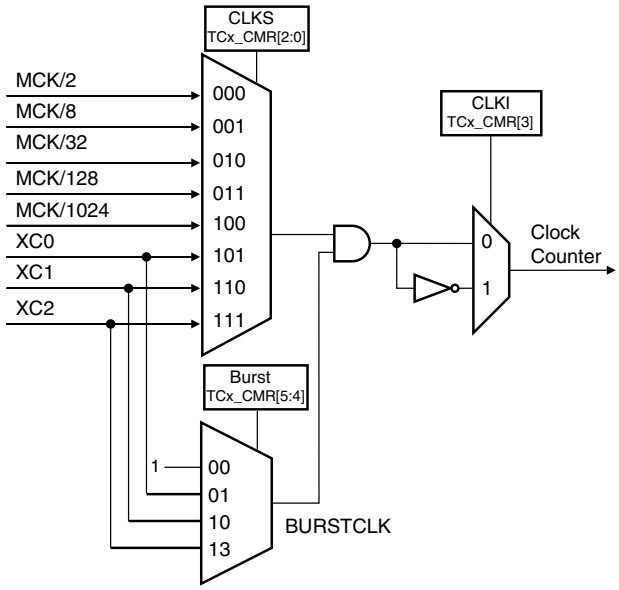
The three-bit TCCLKS field of the mode register TCx_CMR determines whether the counter is clocked by one of the five internal clock sources (MCK/x) or one of the three external clock sources (TCLKx).

The selected clock can be inverted with the CLKI bit in TCx_CMR (Channel Mode Register). This enables counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2).

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the system clock (MCK) period. The external clock frequency must be at least 2.5 times lower than the system clock (MCK).

Figure 2. Timer/Counter Clock Source



The maximal counter duration when an internal clock is used, is determined by the internal clock MCK and the prescaler number:

maximal counter duration (seconds) = $2^{16}/F_{TC}$ where F_{TC} is in Hz.

counter resolution = $1/F_{TC}$

Table 1. Maximum Counter Duration for Various MCK

MCK	5 MHz	10 MHz	20 MHz	33 MHz	66 MHz
MCK/2	26.21ms	13.10ms	6.55ms	3.97ms	1.98ms
MCK/8	104.8ms	52.4ms	26.22ms	14.89ms	7.45ms
MCK/16	419.4ms	209.7ms	104.86ms	63.86ms	31.98ms
MCK/128	1.68s	838.8ms	420.4ms	254.2ms	127.1ms
MCK/1024	13.42s	6.71s	3.36ms	2.03s	1.02s

Timer Interrupt Generation

Each Timer/Counter channel drives an internal interrupt signal which can be programmed to generate processor interrupts via the AIC (Advanced Interrupt Controller). Each Timer/Counter channel contains a total of 8 interrupts, which can be enabled or disabled from the registers TCx_IER and TCx_IDR. The interrupts are available according to the operating mode as shown below in Table 2.

Table 2. Operating Mode Interrupts

Interrupt	Capture Mode	Waveform Mode
Counter Overflow Interrupt COVFS	X	X
Load Overrun Interrupt LOVRS	X	
Compare Register A Interrupt CPAS		X
Compare Register B Interrupt CPBS		X
Compare Register C Interrupt CPCS	X	X
Load Capture Register A Interrupt LDRAS	X	

Application Example

Use the AT91 Timer/Counter to generate an interrupt and blink one LED every 1s. This application example is based on the AT91EB40A Evaluation Board but is applicable to all AT91 products.

Timer configuration

The RC can generate a trigger if bit CPCTRG in the TC Mode Register is set to 1. A trigger resets the counter so that RC can control the timer period needed. The RC compare interrupt will be used to generate an interrupt every 1s. The RC compare interrupt is available in both mode, compare and waveform modes so the timer can be configured even in compare mode or in waveform mode.

The Master Clock MCK on the AT91EB40A Evaluation Board is 66 MHz. As described previously, the timer period is controlled by the compare register RC. The value needed must be determined in the compare register C in order to obtain a timer period of 1s.

The minimal prescaler value required to select the timer clock F_{TC} must first be determined. The maximal counter value is 0xFFFF (65535):

$$DIV_{min} = t \times \frac{MCK}{65535} = 1 \times \frac{66000000}{65535} = 1007.095$$

The value ≥ 1007.095 is DIV = 1024. Therefore the timer clock F_{TC} must be at least MCK/2 MCK/1024 to have a RC compare period of 1s.

In an application, the required compare register values must be calculated using the following equation:

$$\text{Compare Value} = (t \times F_{TC}) - 1$$

Where

t = desired timer compare period (second)

F_{TC} = timer clock frequency(Hertz)

Compare register RC:

$$RC = (t \times F_{TC}) - 1$$

$$\Rightarrow RC = \left(1 \times \frac{66000000}{1024}\right) - 1 = 64453 = 0xFBC5$$

Software Code

The following software code example blinks LED8 on the AT91EB40A Evaluation Board every 1s using the Timer/Counter 1 RC compare interrupt and is applicable to the entire AT91 series.

This software example is built around two files:

- **irq_timer.s** assembly file which defines the assembler timer interrupt assembly handler.
- **timer_interrupt.c** C file which includes the main function with the timer configuration and the C timer interrupt handler.

Irq_timer.s

```
;-------------------------------------------------------------------------------------------------------------------
; The software is delivered "AS IS" without warranty or condition of any kind, either express, implied or
  statutory.
; This includes without limitation any warranty or condition with respect to merchantability or fitness for
  any particular purpose, or
; against the ;infringements of intellectual property rights of others.
;-------------------------------------------------------------------------------------------------------------------
;- File source          : irq_timer.s
;- Object               : Assembler timer Interrupt Handler
;- Author               : AT91 Application Group
;-------------------------------------------------------------------------------------------------------------------

;-------------------------------------------------------------------------------------------------------------------
                ;- Area Definition
;-------------------------------------------------------------------------------------------------------------------
AREA    TIMER_ASM_HANDLER, CODE, READONLY

AIC_BASE        EQU         0xFFFFF000

AIC_IVR         EQU         0x100
AIC_EOICR       EQU         0x130

;- ARM Core Mode and Status Bits
ARM_MODE_IRQ    EQU         0x12
ARM_MODE_SYS    EQU         0x1F

I_BIT           EQU         0x80

            MACRO
            IRQ_ENTRY   $reg

;- Adjust and save LR_irq in IRQ stack
            sub         r14, r14, #4
            stmfd       sp!, {r14}

;- Write in the IVR to support Protect Mode
;- No effect in Normal Mode
;- De-assert the NIRQ and clear the source in Protect Mode
            ldr         r14, =AIC_BASE
            str         r14, [r14, #AIC_IVR]
```

```
;- Save SPSR and r0 in IRQ stack
            mrs         r14, SPSR
            stmfd       sp!, {r0, r14}

;- Enable Interrupt and Switch in SYS Mode
            mrs         r0, CPSR
            bic         r0, r0, #I_BIT
            orr         r0, r0, #ARM_MODE_SYS
            msr         CPSR_c, r0

;- Save scratch/used registers and LR in User Stack
            IF  "$reg" = ""
            stmfd       sp!, { r1-r3, r12, r14}
            ELSE
            stmfd       sp!, { r1-r3, $reg, r12, r14}
            ENDIF

            MEND

            MACRO
            IRQ_EXIT    $reg

;- Restore scratch/used registers and LR from User Stack
            IF  "$reg" = ""
            ldmia       sp!, { r1-r3, r12, r14}
            ELSE
            ldmia       sp!, { r1-r3, $reg, r12, r14}
            ENDIF

;- Disable Interrupt and switch back in IRQ mode
            mrs         r0, CPSR
            bic         r0, r0, #ARM_MODE_SYS
            orr         r0, r0, #I_BIT:OR:ARM_MODE_IRQ
            msr         CPSR_c, r0

;- Mark the End of Interrupt on the AIC
            ldr         r0, =AIC_BASE
            str         r0, [r0, #AIC_EOICR]

;- Restore SPSR_irq and r0 from IRQ stack
            ldmia       sp!, {r0, r14}
            msr         SPSR_cxsf, r14

;- Restore adjusted  LR_irq from IRQ stack directly in the PC
            ldmia       sp!, {pc}^

            MEND
```

```
;-------------------------------------------------------------------------------------------------------------
;- Function             : timer1_asm_irq_handler
;- Treatments           : Timer 1 interrupt handler.
;- Called Functions     : timer1_c_irq_handler
;- Called Macros        : IRQ_ENTRY, IRQ_EXIT
;-------------------------------------------------------------------------------------------------------------
               EXPORT      timer1_asm_irq_handler
            IMPORT      timer1_c_irq_handler

timer1_asm_irq_handler
;- Manage Exception Entry
            IRQ_ENTRY
;- Call the timer Interrupt C handler
            ldr         r1, =timer1_c_irq_handler
            mov         r14, pc
            bx          r1
;- Manage Exception Exit
           IRQ_EXIT
            END
```

Timer_interrupt.c

```
//*-------------------------------------------------------------------------------------------------------------
//* File Name: Timer_interrupt.c
//* Object              : AT91EB40A - Timer Counter - Interrupt
//* Author: AT91 Application Group
//*-------------------------------------------------------------------------------------------------------------

#define  TC1_CCR  ((volatile unsigned int *) 0xFFFE0040)
#define  TC1_CMR  ((volatile unsigned int *) 0xFFFE0044)
#define  TC1_RC   ((volatile unsigned int *) 0xFFFE005C)
#define  TC1_SR   ((volatile unsigned int *) 0xFFFE0060)
#define  TC1_IER  ((volatile unsigned int *) 0xFFFE0064)
#define  TC1_IDR  ((volatile unsigned int *) 0xFFFE0068)

#define  PIO_PER  ((volatile unsigned int *) 0xFFFF0000)
#define  PIO_OER  ((volatile unsigned int *) 0xFFFF0010)
#define  PIO_SODR ((volatile unsigned int *) 0xFFFF0030)
#define  PIO_CODR ((volatile unsigned int *) 0xFFFF0034)
#define  PIO_PDSR ((volatile unsigned int *) 0xFFFF003C)

#define  AIC_SMR5 ((volatile unsigned int *) 0xFFFFF014)
#define  AIC_SVR5 ((volatile unsigned int *) 0xFFFFF094)
#define  AIC_IECR ((volatile unsigned int *) 0xFFFFF120)
#define  AIC_IDCR ((volatile unsigned int *) 0xFFFFF124)
#define  AIC_ICCR ((volatile unsigned int *) 0xFFFFF128)

#define  TC1_ID     5       /* Timer Channel 1 interrupt */

//* TC_CMR: Timer Counter Channel Mode Register Bits Definition
```

```
#define    TC_CLKS_MCK1024           0x4
#define    TC_CPCTRG                  0x4000


//* TC_CCR: Timer Counter Control Register Bits Definition
#define   TC_CLKEN                    0x1
#define   TC_CLKDIS                   0x2
#define   TC_SWTRG                    0x4

//* TC_SR: Timer Counter Status Register Bits Definition
#define   TC_CPCS                     0x10        /* RC Compare Status */

//* AIC_SMR: Interrupt Source Mode Registers
#define    AIC_SRCTYPE_INT_LEVEL_SENSITIVE     0x00    /* Level Sensitive */

//* Leds Definition
#define   LED1                        (1<<16)
#define   LED8                        (1<<6)

extern void timer1_asm_irq_handler(void);

//*-----------------------------------------------------------------------------
//* Function Name       : timer1_c_irq_handler
//* Object              : Timer 1 interrupt Handler
//*-----------------------------------------------------------------------------
void timer1_c_irq_handler (void)
//* Begin
{
    unsigned int dummy ;

    dummy = *TC1_SR;       /* Read TC1 Status Register to clear it */

   if ( (*PIO_PDSR & LED8) == LED8 )
   *PIO_CODR = LED8 ;
   else
   *PIO_SODR = LED8 ;
}
//* End

void delay (void)
{
  unsigned int i;

  for (i=0; i<1000000 ; i++);
}
```

```
//*-----------------------------------------------------------------------------
//* Function Name       : main
//* Object              : AT91 - Timer Counter- PWM generation
//* Input Parameters    : none
//* Output Parameters   : none
//* Functions called    : none
//*-----------------------------------------------------------------------------
int main ( void )
//* Begin
{
unsigned int dummy ;

    *PIO_PER  = LED8 | LED1 ;  /* Enable the PIO/LED8 pin */
    *PIO_OER  = LED8 | LED1;   /* Enable the PIO/LED8 pin as Output */
    *PIO_CODR = LED8 | LED1 ;  /* Set LED8 */

// Timer1 Init
    *TC1_CCR = TC_CLKDIS ;                                  /* Disable the Clock Counter */
    *TC1_IDR = 0xFFFFFFFF ;
    dummy = *TC1_SR ;
    *TC1_CMR = TC_CLKS_MCK1024 |
      TC_CPCTRG ;
    *TC1_CCR = TC_CLKEN ;                                   /* Enable the Clock counter */
    *TC1_IER  = TC_CPCS ;                                   /* Validate the RC compare interrupt */

    *AIC_IDCR = (1<<TC1_ID) ;                               /* Disable timer 1 interrupt at AIC level */
    *AIC_SVR5 = (unsigned int) timer1_asm_irq_handler ;     /* Set the TC1 IRQ handler address */
    *AIC_SMR5 = ( AIC_SRCTYPE_INT_LEVEL_SENSITIVE | 0x4 ); /* Set the trigg and priority for TC1 interrupt */
    *AIC_ICCR = (1<<TC1_ID) ;                              /* Clear the TC1 interrupt */
    *AIC_IECR = (1<<TC1_ID) ;                               /* Enable the TC1 interrupt */

    *TC1_RC = 0xFBC5;
    *TC1_CCR = TC_SWTRG ;

    while (1)
    {
    *PIO_CODR = LED1 ;
    delay();
   *PIO_SODR = LED1 ;
   delay();
    }

    return(0) ;

}//*End
```

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

2683A–ATARM–01/03 0M