Physical Computing
2007-2008
Interpretatie I
Programmeerproject I

Coen De Roover - Christophe Scholliers - Yves Vandriessche

Programming Technology Lab
Vrije Universiteit Brussel
Belgium
(define timer0 #xE0004000) ; Timer0
(define timer1 #xE0008000) ; Timer1
(define timer-control #x04) ; TCR offset
(define timer-count #x08) ; TxTC offset
(define timer-period #x0C) ; TxPC offset

<table>
<thead>
<tr>
<th>Generic Name</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
<th>TIMER0 Address &amp; Name</th>
<th>TIMER1 Address &amp; Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR</td>
<td>Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.</td>
<td>R/W</td>
<td>0</td>
<td>0xE0004000 T0IR</td>
<td>0xE0004000 T1IR</td>
</tr>
<tr>
<td>TCR</td>
<td>Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.</td>
<td>R/W</td>
<td>0</td>
<td>0xE0004004 T0TCR</td>
<td>0xE0004004 T1TCR</td>
</tr>
<tr>
<td>TC</td>
<td>Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pulse. The TC is controlled through the TCR.</td>
<td>R/W</td>
<td>0</td>
<td>0xE0004000 T0TC</td>
<td>0xE0004000 T1TC</td>
</tr>
<tr>
<td>PR</td>
<td>Prescale Register. The TC is incremented every PR+1 cycles of pulse.</td>
<td>R/W</td>
<td>0</td>
<td>0xE0004000 C0PR</td>
<td>0xE0004000 C1PR</td>
</tr>
<tr>
<td>PC</td>
<td>Prescale Counter. This 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented.</td>
<td>R/W</td>
<td>0</td>
<td>0xE0004010 T0PC</td>
<td>0xE0004010 T1PC</td>
</tr>
</tbody>
</table>
Timer Prescale Register

Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

```
PCLK        +1  +1  +1  +1  +1  +1  +1  +1
            32
            60 MHz
```

PC: 0xE0004010

```
00000000000000000000000000000000
```

PR: 0xE000400C

```
000000000000000000000000000000001111011
```

TC: 0xE0004008

```
00000000000000000000000000000000
```
Timer Prescale Register

Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pckls when PR = 1, etc.
**Timer Prescale Register**

**Prescale Counter Register** (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

![Prescale Counter Register Diagram]

- **PCLK**: 60 MHz
- **PR**: 0xE000400C
- **PC**: 0xE0004010
- **TC**: 0xE0004008
Timer Prescale Register

Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

PCLK
60 MHz

PC: 0xE0004010

PR: 0xE000400C

TC: 0xE0004008
Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.
**Timer Prescale Register**

**Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)**

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

**PCLK 60 MHz**

- **PC: 0xE0004010**
  - 00000000000000000000000000000000111011

- **PR: 0xE000400C**
  - 00000000000000000000000000000000111011

- **TC: 0xE0004008**
  - 0000000000000000000000000000000000000000
Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

PCLK

60 MHz

32

PC: 0xE0004010

PR: 0xE000400C

TC: 0xE0004008
Timer Prescale Register

Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.
Timer Prescale Register

Prescale Counter Register (PC: TIMER0 - T0PC: 0xE0004010; TIMER1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

PCLK
60 MHz

PC: 0xE0004010

PR: 0xE000400C

TC: 0xE0004008
# Timer Control Register

Table 124: Timer Control Register (TCR: TIMER0 - T0TCR: 0xE0004004; TIMER1 - T1TCR: 0xE0008004)

<table>
<thead>
<tr>
<th>TCR</th>
<th>Function</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Counter Enable</td>
<td>When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Counter Reset</td>
<td>When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.</td>
<td>0</td>
</tr>
</tbody>
</table>
; function to stop a timer
(define (stop timer)
  (write 0 timer timer-control)) ; disable timer via TxCr

; function to restart a timer
(define (restart timer)
  (write 2 timer timer-control) ; reset timer via TxCr
  (write 1 timer timer-control)) ; enable and start timer via TxCr
Putting it together

; configure timer0
(write 59 timer0 timer-period) ; set timer1 period to 1 us (for 60MHz clk) via T1PR
(restart timer0)
; read the timer count (to check proper operation, read it a few times)
(read timer0 timer-count)
(read timer0 timer-count)
Analog Digital

Image from: http://www.engr.colostate.edu/~dga/me307/lectures.html
A/D Registers

**Control**

**Data**

Table 138: A/D Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCR</td>
<td>A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.</td>
<td>Read/Write</td>
<td>0x0000 0001</td>
<td>0xE003 4000</td>
</tr>
<tr>
<td>ADDR</td>
<td>A/D Data Register. This register contains the ADC’s DONE bit and (when DONE is 1) the 10-bit result of the conversion.</td>
<td>Read/Write</td>
<td>NA</td>
<td>0xE003 4004</td>
</tr>
</tbody>
</table>

ADCR:0xE003 4000

ADDR:0xE003 4004
### A/D Data Register

**A/D Data Register (ADDR - 0xE0034004)**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Name</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>DONE</td>
<td>This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the APCR is written. If the APCR is written while a conversion is still in progress, this bit is set and a new conversion is started.</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>OVERUN</td>
<td>This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the LS bits. In non-FIFO operation, this bit is cleared by reading this register.</td>
<td>0</td>
</tr>
<tr>
<td>29:27</td>
<td>These bits always read as zeroes. They could be used for expansion of the CHN field in future compatible A/D converters that can convert more channels.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>26:24</td>
<td>CHN</td>
<td>These bits contain the channel from which the LS bits were converted.</td>
<td>X</td>
</tr>
<tr>
<td>23:16</td>
<td>These bits always read as zeroes. They allow accumulation of successive A/D values without AND-masking, for at least 256 values without overflow into the CHN field.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>15:6</td>
<td>V/V\textsubscript{3A}</td>
<td>When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin selected by the SEL field, divided by the voltage on the VddA pin. Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on V\textsubscript{SSA}, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on V\textsubscript{3A}. For testing, data written to this field is captured in a shift register that is clocked by the A/D converter clock. The MS bit of this register sources the DIN[1][0] input of the A/D converter, which is used only when TEST1.0 are 10.</td>
<td>X</td>
</tr>
<tr>
<td>5:0</td>
<td>These bits always read as zeroes. They provide compatible expansion room for future, higher-resolution A/D converters.</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
A/D Data Register

A/D Control Register (ADCR - 0xE0034000)

Table 139: A/D Control Register (ADCR - 0xE0034000)

<table>
<thead>
<tr>
<th>ADCR</th>
<th>Name</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>SEL</td>
<td>Selects which of the Ain3:0 (LPC2114/2124) or Ain7:0 (LPC2212/2214) pins is (are) to be sampled and converted. Only bits 3:0 should be set to 1 in the 48 or 64 pin package. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones (1 to 4 ones in the 48 or 64 pin package) can be used. All zeroes is equivalent to 0x01.</td>
<td>0x01</td>
</tr>
<tr>
<td>15:8</td>
<td>CLKDIV</td>
<td>The VPB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.</td>
<td>0</td>
</tr>
<tr>
<td>26:24</td>
<td>START</td>
<td>When the BURST bit is 0, these bits control whether and when an A/D conversion is started: 000: no start (this value should be used when clearing PDN to 0) 001: start conversion now 010: start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/ CAP0.2 011: start conversion when the edge selected by bit 27 occurs on P0.22/CAP0.0/MAT0.0 Note: for choices 100-111 the MAT signal need not be pinned out: 100: start conversion when the edge selected by bit 27 occurs on MAT0.1 101: start conversion when the edge selected by bit 27 occurs on MAT0.3 110: start conversion when the edge selected by bit 27 occurs on MAT1.0 111: start conversion when the edge selected by bit 27 occurs on MAT1.1</td>
<td>000</td>
</tr>
</tbody>
</table>

Download Example Code
Light Detection