# AN10929

## Timers 2, 3 and 4 of P89LPC97x/98x

Rev. 01 — 8 April 2010 Application note

### Document information

| Info | Content |
|---|---|
| **Keywords** | LPC98x, LPC97x, Timers 2, 3 and 4, timer, PWM, Keil MCB900 |
| **Abstract** | This application note describes how to use Timers 2, 3 and 4 in LPC98x/97x. Demo code is also provided. |

### Revision history

| Rev | Date | Description |
|---|---|---|
| 01 | 20100408 | Initial version. |

## Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

# 1. Introduction

The P89LPC98x/97x devices have a total of five timers. Timer 0 and Timer 1 are two general-purpose counter/timers which are upward compatible with the 80C51 Timer 0 and Timer 1. In addition, the P89LPC98x/97x has three external 16-bit timers/counters: Timers 2, 3 and 4. These three timers can also be configured to operate either as timers or event counters.

Two pins for each of timers 2, 3 and 4:

- T2/T3/T4 - Timer/counter 2/3/4 external count input or overflow output.
- T2EX/T3EX/T4EX - Timer/counter 2/3/4 external capture input, which is only available on Timers 2, 3 and 4.

# 2. Operation modes and registers

Timers 2, 3 and 4 have three operating modes:

- Mode 0 - 16-bit Timer/Counter with auto-reload;
- Mode 1 - 16-bit Timer/Counter with Input Capture;
- Mode 2 - 16-bit PWM mode.

In the 'Timer' function, the timer is incremented on every PCLK. In the 'Counter' function, the register is incremented in response to a high-to-low transition on its corresponding count input pin Tx (x = 2, 3 or 4). Since it takes two machine cycles (four CPU clocks) to recognize a high-to-low transition, the maximum count rate is 1/4 of the CPU clock frequency.

The TxCON (x = 2, 3 or 4) register shown in Table 1 is used as the control register for Timers 2, 3 and 4 respectively. The combination of PWMx bit and CP/NRLx bit implements different operation modes shown in Table 1. Operation as a timer or counter is selected by the C/NTx bit: it is cleared for timer operation (input from PCLK) and set for counter operation (input from Tx input pin). When in 16-bit PWM mode, the C/NTx bit must be cleared.

**Table 1. Timer/Counter x Control (TxCON – where x = 2, 3 or 4) bit allocation**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Symbol | PSELx | ENTx | TIENx | PWMx | EXENx | TRx | C/NTx | CP/NRLx |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 2. Operation mode selected (x = 2, 3 or 4)**

| PWMx | CP/NRLx | Modes |
|---|---|---|
| 0 | 0 | 16-bit Timer/Counter with auto-reload |
| 0 | 1 | 16-bit Timer/Counter with Input Capture |
| 1 | 0 | 16-bit PWM mode |

The overflow flag, capture flag and interrupt setting of Timers 2, 3 and 4 can be read or set in the TINTF register. The two bits for each timer are shown in Table 3. The TFx bit is an overflow flag and set by hardware when the Timer/Counter x overflows. The EXFx bit is external flag and set when a capture or reload is triggered by a negative transition on TxEX and EXENx. If the Timer x interrupt is enabled, setting of the EXFx will cause an interrupt.

**Table 3. Timer/Counters Overflow and External Flags (TINTF) bit allocation**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Symbol | - | - | TF4 | EXF4 | TF3 | EXF3 | TF2 | EXF2 |
| Reset | X | X | 0 | 0 | 0 | 0 | 0 | 0 |

Each timer has six data register: THx, TLx, RCAPxH, RCAPxL, PWMDxH, PWMDxL (x = 2, 3 or 4).

THx and TLx are 16-bit counting registers. RCAPxH, RCAPxL are 16-bit reload register in mode 0 and 16-bit capture register in mode 1. In 16-bit PWM mode, the high period value of PWM signal is in the RCAPxH, RCAPxL register and the low period value in PWMDxH, PWMDxL register. A detailed description on these data register is given in the following sections.

## 3. Mode 0: 16-bit Timer/Counter with Auto-reload

Mode 0 configures Timer x as a 16-bit Timer/Counter with auto-reload, as shown in Fig 1 (where x = 2, 3, 4). In this mode, Timer x can be configured as either a timer or counter via C/NTx in TxCON and then programmed to count up (Where x = 2, 3, 4) as shown in Table 4.

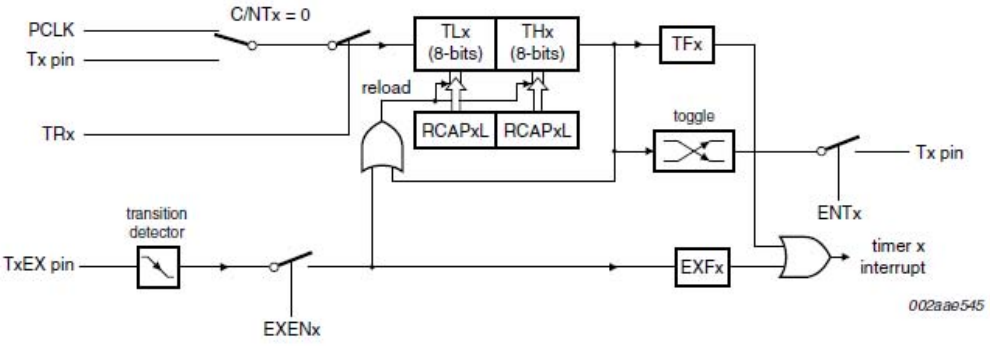



**Fig 1. 16-bit Timer/Counter with Auto-reload**

**Table 4. Mode 0 Operation (x = 2, 3 or 4)**

| PWMx | CP/NRLx | C/NTx | Modes |
|---|---|---|---|
| 0 | 0 | 0 | 16-bit Timer with auto-reload |
| | | 1 | 16-bit Counter with auto-reload |

The 16-bit reloaded value is put in the register RCAPxL and RCAPxH and can be preset by software.

There are two options to trigger the auto-reload that can be selected by the EXENx bit (TxCON.3). When EXENx is cleared, Timer x counts up to 0FFFFH and sets the TFx bit upon overflow. This causes the Timer x registers to be reloaded. When EXENx is set, the 16-bit reload can be triggered by not only the overflow of Timer x but also a 1-to-0 transition at input TxEX. This transition also sets the EXFx bit. It takes two consecutive machine cycles to recognize falling edge on TxEX and another machine cycle to set EXFx.

The Timer x interrupt, if enabled, can be generated when TFx or EXFx is set.

The sample code shown below is a simple example of initializing timer 3 on mode 0 in Keil IDE. In this code, Timer 3 works as a 16-bit Timer with auto-reload. Overflow occurs when timer 3 counts up to 0FFFFH and T3 pin will be toggled. Both the overflow and a falling edge on input T3EX will trigger an auto-reload. The reloaded value is 4000H.

```
1     T3CON =0x48;                    // timer3 mode 0
2     RCAP3H=0x40;                     // initialize timer 3 register
3     RCAP3L=0;
4     T3CON|=0x04;                    // start timer 3
```

## 4. Mode 1: 16-bit Timer/Counter with Input Capture

Mode 1 configures Timer x as a 16-bit Timer/Counter with input capture, as shown in Fig 2 (Where x = 2, 3, 4).

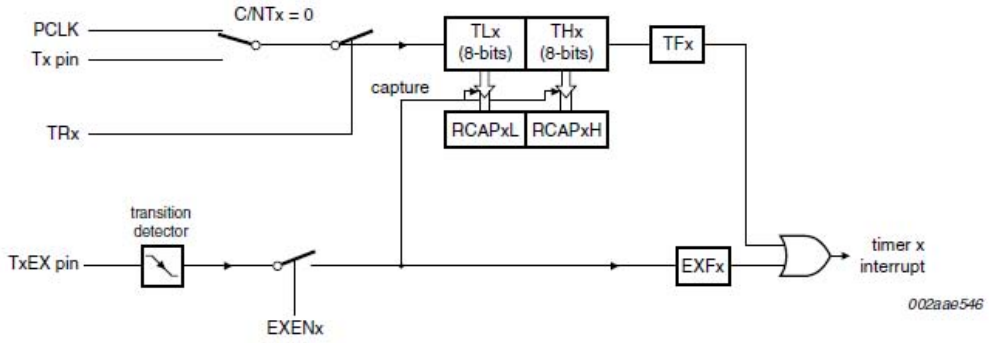



**Fig 2. 16-bit Timer/Counter with Input Capture**

In this mode, TLx and THx registers keep on counting high-to-low transitions at Tx pin or PCLK, regardless of whether the capture event happens or not.

The capture function can be enabled or disabled via the EXENx bit (TxCON.3).

If EXENx is cleared, there is no capture operation and Timer x is a 16-bit timer/counter (selected by C/NTx in TxCON). When overflow occurs, TFx bit in the TINTF register will be set. The Timer x interrupt, if enabled, can be generated when TFx or EXFx is set.

If EXENx is set, when a high-to-low transition occurs at external input TxEX, the current value in the Timer x register TLx:THx is captured to the corresponding 16-bit register RCAPxL:RCAPxH. In addition, the transition at TxEX will set EXFx bit in TINTF. EXFx bit, like TFx bit, can generate an interrupt (which vectors to the same location as Timer x overflow interrupt). The Timer x interrupt service routine can check TFx and EXFx to determine which event caused the interrupt.

Since the contents of RCAPxL:RCAPxH registers are not protected, once Timer x interrupt is generated, the value in registers RCAPxL:RCAPxH should be read before new capture event occurs. Otherwise, the next capture will corrupt the contents of capture registers.

## 4.1 Mode 1 demo

This demo gives a simple example of timer 4 on mode 1. The Keil MCB900 board with an LPC982 is used.

In this example, Timer 4 works as a 16-bit Timer with capture. An external falling edge on T4EX (P1.4) is needed to trigger a capture.

The function init() initializes timer 4. PCLK is selected as the clock source. A falling edge on input T4EX will trigger a capture. Since the interrupt is enabled, the transition at T4EX will set EXF4 bit in TINTF and generate an interrupt. TH4 and TL4 are set to zero so timer 4 counts up from zero.

The Interrupt service routine XTimer_ISR() will record the captured content of timer 4. And global variant of ***cap_flag*** will be set when a capture event occurs.

***cap_flag*** is polled in the main function. If set, the captured content will be transmitted through UART interface. The UART interface is configured as 9600 Hz baud rate, 8 bit data, no parity, 1 stop bit. A snippet of the example code is shown below.

```
1    #include <REG98x.h>
2    #include <stdio.h>
3
4    unsigned char cap_datah, cap_datal,cap_flag;
5    ……
6    void main(void)
7    {
8          cap_datah=0;
9          cap_datal=0;
10         cap_flag=0;
11
12         P0M1 = 0x00;          // bi-directional
13         P0M2 = 0x00;
14         P1M1 = 0x10;          // P1.4 input only, others bi-directional
15         P1M2 = 0x00;
16         P2M1 = 0x00;          // bi-directional
17         P2M2 = 0x00;
18
19          SCON   = 0x52;             /* initialize UART                    */
20          BRGR0  = 0xF0;             /* 9600 baud, 8 bit, no parity, 1 stop bit * /
21          BRGR1  = 0x02;
22          BRGCON = 0x03;
23          ……
24         timer4_init();            // set up timer4 mode 1
25
26         while(1)
27         {
28               //if a capture occurs, display the content and clear the flag
29               if (cap_flag==1)
30               {
31                     capture_display();
32                     cap_flag=0;
33               }
34         }
35   }
```

```
36
37   void timer4_init(void)
38   {
39           T4CON =0x29;                    // timer4 in mode 1
40           TH4=0;
41           TL4=0;
42
43           EXTIM=1;                 //enable interrupt of timer 2/3/4
44           EA=1;
45           T4CON|=0x04;             // start timer 4
46   }
47
48   void XTimer_ISR(void) interrupt 11
49   {
50      if (TINTF & 0x10)
51      {
52       cap_datah=RCAP4H;            // record the captured content
53       cap_datal=RCAP4L;
54       cap_flag=1;                  //set the flag
55         }
56      TINTF=0;
57   }
58   ……
```

## 4.2 Demo setup

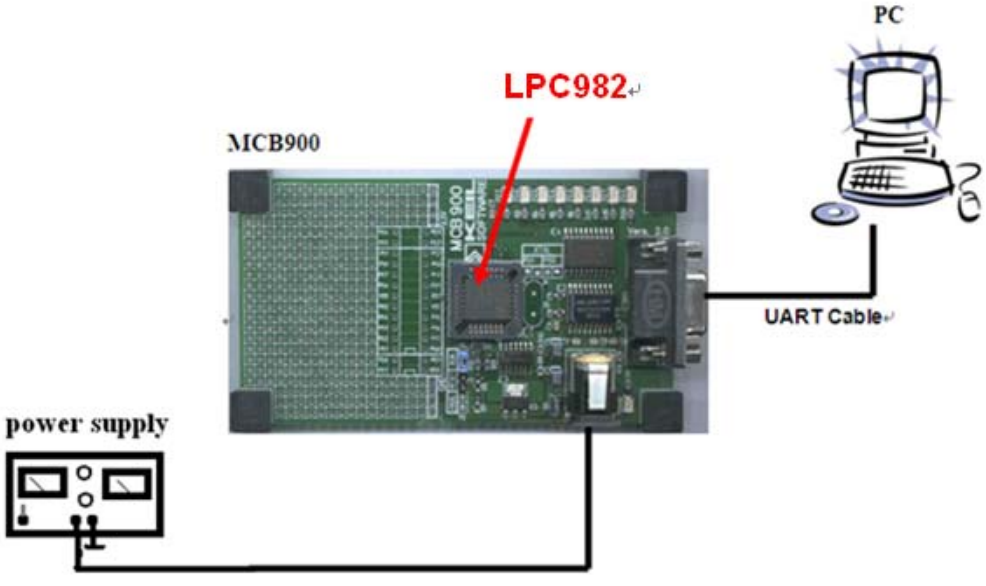



Fig 3. Demo setup

### 4.3 Output information

In this example, an external falling edge on T4EX (P1.4) is needed to trigger a capture event. If a capture occurs, the captured content will be transmitted through UART interface.

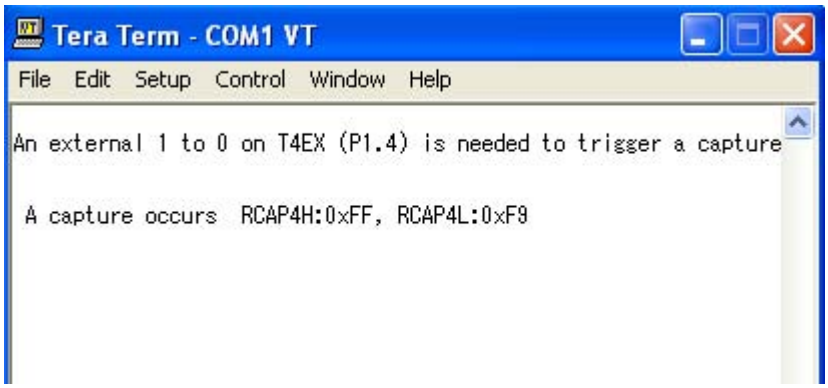



**Fig 4. Output information**

## 5. Mode 2: 16-bit PWM mode

In PWM mode, the corresponding timer can be configured as a PWM generator shown in Fig 5. To enter PWM mode correctly, C/NTx, CP/NRLx bit in TxCON register should be clear and PWMx bit should be set.

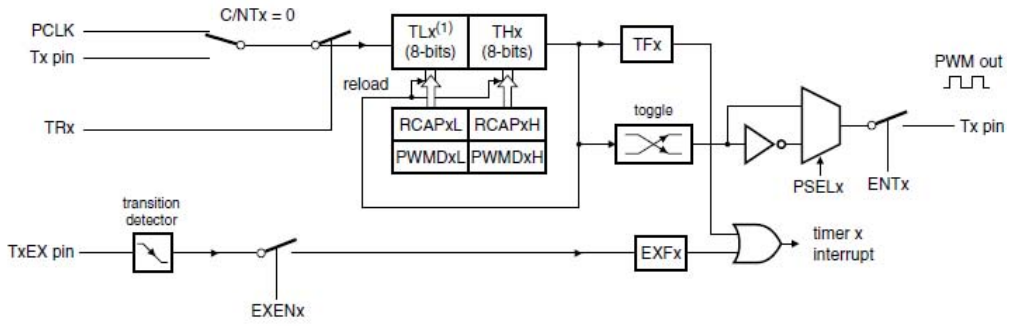



**Fig 5. 16-bit PWM mode**

The registers of RCAPxH ,RCAPxL and PWMDxH, PWMDxL are used to adjust the duty cycle of PWM waveform through the following formula:

*Duty Cycle = RCAPxH:RCAPxL / (RCAPxH:RCAPxL + PWMDxH:PWMDxL)* *(1)*

The ENTx bit in TxCON should be set to output PWM signal. In addition, PWM output has a polarity control function. When control register bit PSELx is set, PWM will output an inversed waveform.

Fig 6 illustrates the Tx pin output when timer x operating in PWM mode. We can see that both the frequency and the duty cycle of PWM waveform can be adjusted according to the application requirements.


**Fig 6. PWM signal output**

Its structure is similar to auto-reload mode, except that:

- TFx (where x = 2, 3 or 4) is set and cleared in hardware;
- The high period of the TFx is in RCAPxH:RCAPxL, and the low period of TFx is PWMDxH:PWMDxL. They should be between 1 and 65536;
- Only loading RCAPxH:RCAPxL with 0000h will force the Tx pin low, and only loading PWMDxH:PWMDxL with 0000h will force the Tx pin high. Loading both RCAPxH:RCAPxL and PWMDxH:PWMDxL with 0000h will force the Tx pin high.

Interrupts can still be enabled on the low to high transition of TFx and TFx and can be cleared in software like in other modes.

## 5.1 PWM mode demo

This demo gives examples on how to generate PWM signal with Timers 2, 3 and 4. The Keil MCB900 board with a LPC982 is used.

In this example, Timer 2 works as a 16-bit PWM mode.

The function timer2_init() gives the initialization of timer 2. The RCAP2H, RCAP2L, PWMD2H and PWMD2L registers are initialized to 1.

In the main function, the high period and the low period of PWM waveform vary simultaneously. Measuring T2 (P0.3) pin with an oscilloscope, it will display the varied PWM waveform.

```
1    void Timer2_init(void)
2    {
3         T2CON =0x40;               // timer2:  PWM mode
4         RCAP2H=1;                   // initialize the high period and low period
5         RCAP2L=1;
6         PWMD2H=1;
7         PWMD2L=1;
8
9          T2CON|=0x14;               // start timer 2
10   }
11
12   void main(void)
13   {
14        P0M1 = 0x00;          // PushPull via P0.3
15        P0M2 = 0x08;
16
17        Timer2_init();        // set up timer2 in PWM  mode
18
```

```
19          while(1)
20          {
21                for(n=1; n<99;n++)    // vary high period and low period of PWM
22                {
23                      PWMD2H=n;
24                      PWMD2L=n;
25                      RCAP2H=n;
26                      RCAP2L=n;
27                      msec(15);
28                }
29                msec(100);
30                for(n=99; n>1; n--)
31                {
32                      PWMD2H=n;
33                      PWMD2L=n;
34                      RCAP2H=n;
35                      RCAP2L=n;
36                      msec(15);
37                }
38                msec(100);
39          }
40    }
```

This demo setup is the same as mode 1 shown in Fig 3.

## 5.2 PWM output information

The PWM demo gives a PWM signal output on T2 (P0.3) pin using Timer 2. The high period and the low period of PWM waveform vary simultaneously. Using an oscilloscope, the varied PWM waveform can be viewed by monitoring T2 (P0.3) pin. Fig 7 gives the snapshot of observed PWM signal. It shows how the frequency of PWM varies. Fig 8 gives the zoom in of Fig 7 in some time slot; one is low frequency PWM and the other is high frequency PWM.

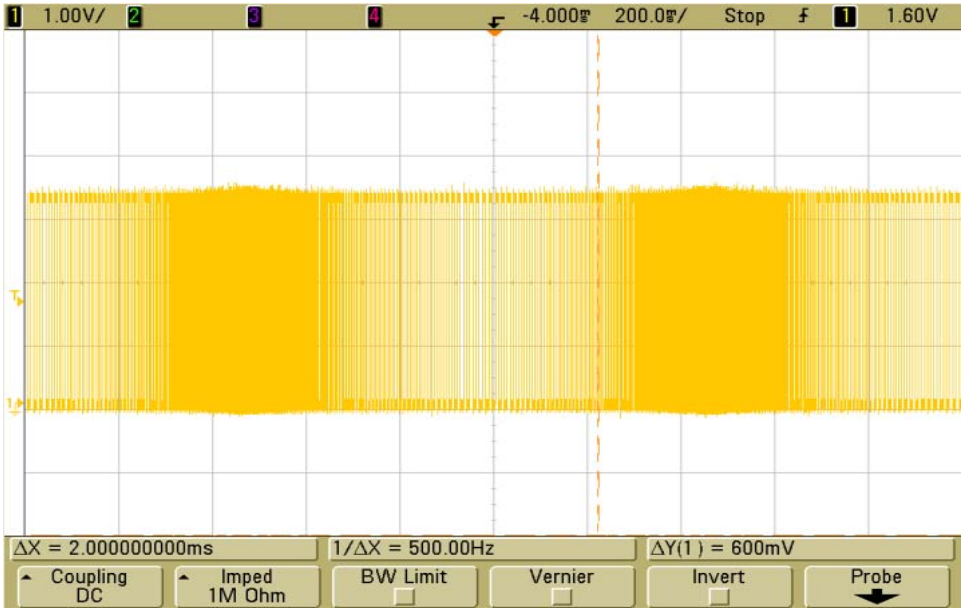
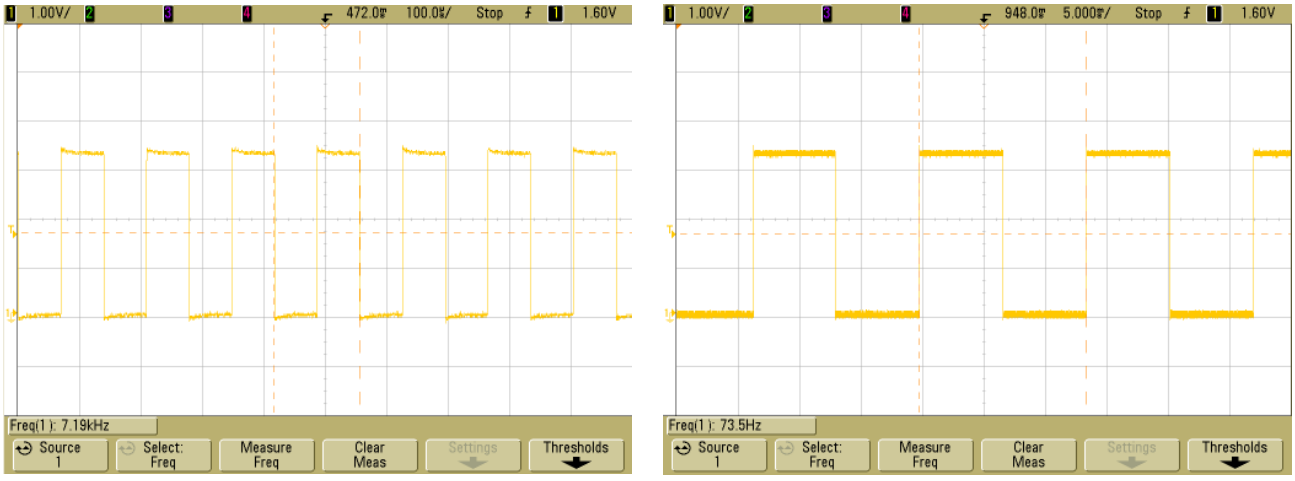



**Fig 7. Snapshot of PWM output**




a. Low frequency PWM

b. High frequency PWM

**Fig 8. Zoom in PWM output**

## 6. Reference

[1] LPC98x User Manual (UM10346）

[2] LPC98x Data Sheet

# 7. Legal information

## 7.1 Definitions

**Draft —** The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

## 7.2 Disclaimers

**Limited warranty and liability —** Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes —** NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use —** NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications —** Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on a weakness or default in the customer application/use or the application/use of customer's third party customer(s) (hereinafter both referred to as "Application"). It is customer's sole responsibility to check whether the NXP Semiconductors product is suitable and fit for the Application planned. Customer has to do all necessary testing for the Application in order to avoid a default of the Application and the product. NXP Semiconductors does not accept any liability in this respect.

**Export control —** This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

## 7.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

## 8. List of figures

# 9. List of tables

## 10. Contents





For more information, please visit: http://www.nxp.com
For sales office addresses, please send an please send an email to:
salesaddresses@nxp.com

**Date of release: 8 April 2010**
**Document identifier: AN10929_1**