Using Watchdog Timer in Vortex86SX SoC

2007-09-28

There are two watchdog timers in Vortex86SX CPU. One is compatible with M6117D watchdog timer and the other is new. The M6117D compatible watchdog timer is called WDT0 and new one is called WDT1.

WDT0

To access WDT0 registers, programmer can use index port 22H and data port 23H. The watchdog timer uses 32.768 kHz frequency source to count a 24-bit counter so the time range is from 30.5u sec to 512 sec with resolution 30.5u sec. When timer times out, a system reset, NMI or IRQ may happen to be decided by BIOS programming.

Index Port 37h	
Bit 7	Reserved.
Bit 6	0: Disable WDT0 1: Enable WDT0 (default)
Bit 5-0	Reserved.
Index Port 3Ch	
Bit 7	0: Read only, Watchdog timer time out event does not happen. 1: Read only, Watchdog timer time out event happens.
Bit 6	Write 1 to reset Watchdog timer.
Index Port 38h	
Bit 7-4	0000:Reserved 0101:IRQ7 1011:IRQ15 0001:IRQ3 0110:IRQ9 1100:NMI 0010:IRQ4 0111:IRQ10 1101:System reset 0011:IRQ5 1001:IRQ12 1110:Reserved 0100:IRQ6 1010:IRQ14 1111:Reserved
Bit 3-0	Reserved.

Index 3Bh, 3Ah, 39h : Counter

	3Bh	3Ah	39h
	D7……D0	D7……D0	D7……D0
Counter	Most SBit ……Least SBit		

Here are steps to setup watchdog timer:

1. Set Bit 6 = 0 to disable the timer.
2. Write the desired counter value to 3Bh, 3Ah, 39h.
3. Set Bit 6 = 1 to enable the timer, the counter will begin to count up.
4. When counter reaches the setting value, the time out will generate signal setting by index 38h bit[7:4]
5. BIOS can read index 3Ch Bit 7 to decide whether the Watchdog timeout event will happen or not.

To clear the watchdog timer counter:

1. Set Bit 6 = 0 to disable timer. This will also clear counter at the same time.

WDT1

WDT1 does not use index and data port to access WDT registers. It uses I/O port 68H~6DH. The time resolution of WDT1 is 30.5 u second. Here are registers information:

WDT1 Control Register

Port 68h	
Bit 7	Reserved.
Bit 6	0: Disable watchdog timer. 1: Enable watchdog timer.
Bit 5-0	Reserved.

WDT1 Signal Select Control Register

Port 69h			
Bit 7-4	0000:Reserved	0101:IRQ7	1011:IRQ15
	0001:IRQ3	0110:IRQ9	1100:NMI
	0010:IRQ4	0111:IRQ10	1101:System reset
	0011:IRQ5	1001:IRQ12	1110:Reserved
	0100:IRQ6	1010:IRQ14	1111:Reserved
Bit 3-0	Reserved.		

WDT1 Control 2 Register

Port	**6Ch**	**6Bh**	**6Ah**
	D7……D0	D7……D0	D7……D0
Counter	Most SBit ……Least SBit		

Resolution is 30.5u second.

WDT1 Status Register

Port 6Dh	
Bit 7	0: WDT1 timeout event does not happen 1: WDT1 timeout event happens (write 1 to clear this flag)
Bit 6-0	Reserved.

WDT1 Reload Register

Port 67h	
Bit 7-0	Write this port to reload WDT1 internal counter. The read data is unknown.

Here are steps to setup WDT1:

1. Write time into register 6Ah-6Ch.
2. Select signal from register 69h.
3. Set register 68h bit 8 to enable WDT1.

To clear the watchdog timer counter:

1. Write any value to register 67H

WDT0 DOS Example

```
#include <stdio.h>
#include <conio.h>

void main()
{
  unsigned char c;
  unsigned int lTime;

  outp(0x22,0x13); // Lock register
  outp(0x23,0xc5); // Unlock config. register

  // 500 mini-second
  lTime = 0x20L * 500L;
  outp(0x22,0x3b);
  outp(0x23,(lTime>>16)&0xff);
  outp(0x22,0x3a);
  outp(0x23,(lTime>> 8)&0xff);
  outp(0x22,0x39);
  outp(0x23,(lTime>> 0)&0xff);

  // Reset system
  outp(0x22,0x38);
  c = inp(0x23);
  c &= 0x0f;
  c |= 0xd0; // Reset system. For example, 0x50 to trigger IRQ7
  outp(0x22,0x38);
  outp(0x23,c);

  // Enable watchdog timer
  outp(0x22,0x37);
  c = inp(0x23);
  c |= 0x40;
  outp(0x22,0x37);
  outp(0x23,c);

  outp(0x22,0x13); // Lock register
  outp(0x23,0x00); // Lock config. register

  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
  {
    outp(0x22,0x13);  // Unlock register
    outp(0x23,0xc5);
    outp(0x22,0x3c);
    unsigned char c = inp(0x23);
    outp(0x22,0x3c);
    outp(0x23,c|0x40);
```

```
    outp(0x22,0x13);  // Lock register
    outp(0x23,0x00);
  }

  printf("System will reboot after 500 milli-seconds.\n");
}
```

WDT1 DOS Example

```
#include <stdio.h>
#include <conio.h>

void main()
{
  unsigned char c;
  unsigned long lTime;

  // 500 mini-second
  lTime = 0x20L * 500L;
  outp(0x6c, (lTime >> 16) & 0xff);
  outp(0x6b, (lTime >>  8) & 0xff);
  outp(0x6a, (lTime >>  0) & 0xff);

  // Reset system. For example, 0x50 to trigger IRQ7
  outp(0x69, 0xd0);

  // Enable watchdog timer
  c = inp(0x68);
  c |= 0x40;
  outp(0x68, c);

  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
    outp(0x67, 0x00);

  printf("System will reboot after 500 milli-seconds.\n");
}
```

WDT0 Linux Example

```
#include <stdio.h>
#include <sys/io.h>

#include <termios.h>
#include <unistd.h>
#include <assert.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>

#define outp(a, b)  outb(b, a)
#define inp(a)     inb(a)

int kbhit(void)
{
```

```
  struct timeval  tv;
  struct termios  old_termios, new_termios;
  int error;
  int count = 0;
  tcgetattr(0, &old_termios);
  new_termios = old_termios;
  new_termios.c_lflag &= ~ICANON;
  new_termios.c_lflag &= ~ECHO;
  new_termios.c_cc[VMIN] = 1;
  new_termios.c_cc[VTIME] = 1;
  error = tcsetattr(0, TCSANOW, &new_termios);
  tv.tv_sec = 0;
  tv.tv_usec = 100;
  select(1, NULL, NULL, NULL, &tv);
  error += ioctl(0, FIONREAD, &count);
  error += tcsetattr(0, TCSANOW, &old_termios);
  return error == 0 ? count : -1;
}

int main(void)
{
  iopl(3);

  unsigned char c;
  unsigned int  lTime;
  outp(0x22, 0x13);   // Lock register
  outp(0x23, 0xc5);   // Unlock config. register

  // 500 mini-second
  lTime = 0x20L * 500L;
  outp(0x22, 0x3b);
  outp(0x23, (lTime >> 16) & 0xff);
  outp(0x22, 0x3a);
  outp(0x23, (lTime >> 8) & 0xff);
  outp(0x22, 0x39);
  outp(0x23, (lTime >> 0) & 0xff);

  // Reset system
  outp(0x22, 0x38);
  c = inp(0x23);
  c &= 0x0f;
  c |= 0xd0;         // Reset system. For example, 0x50 to trigger IRQ7
  outp(0x22, 0x38);
  outp(0x23, c);

  // Enable watchdog timer
  outp(0x22, 0x37);
  c = inp(0x23);
  c |= 0x40;
  outp(0x22, 0x37);
  outp(0x23, c);
  outp(0x22, 0x13);   // Lock register
  outp(0x23, 0x00);   // Lock config. register
  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
  {
    outp(0x22, 0x13); // Unlock register
    outp(0x23, 0xc5);
    outp(0x22, 0x3c);

    unsigned char c = inp(0x23);
```

```
    outp(0x22, 0x3c);
    outp(0x23, c | 0x40);
    outp(0x22, 0x13); // Lock register
    outp(0x23, 0x00);
  }

  printf("System will reboot after 500 milli-seconds.\n");
  return 0;
}
```

WDT1 Linux Example

```
#include <stdio.h>
#include <sys/io.h>

#include <termios.h>
#include <unistd.h>
#include <assert.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>

#define outp(a, b)  outb(b, a)
#define inp(a)     inb(a)

int kbhit(void)
{
  struct timeval  tv;
  struct termios  old_termios, new_termios;
  int error;
  int count = 0;
  tcgetattr(0, &old_termios);
  new_termios = old_termios;
  new_termios.c_lflag &= ~ICANON;
  new_termios.c_lflag &= ~ECHO;
  new_termios.c_cc[VMIN] = 1;
  new_termios.c_cc[VTIME] = 1;
  error = tcsetattr(0, TCSANOW, &new_termios);
  tv.tv_sec = 0;
  tv.tv_usec = 100;
  select(1, NULL, NULL, NULL, &tv);
  error += ioctl(0, FIONREAD, &count);
  error += tcsetattr(0, TCSANOW, &old_termios);
  return error == 0 ? count : -1;
}

int main(void)
{
  iopl(3);

  unsigned char c;
  unsigned long lTime;

  // 500 mini-second
  lTime = 0x20L * 500L;
  outp(0x6c, (lTime >> 16) & 0xff);
  outp(0x6b, (lTime >> 8) & 0xff);
  outp(0x6a, (lTime >> 0) & 0xff);
```

```
  // Reset system. For example, 0x50 to trigger IRQ7
  outp(0x69, 0xd0);

  // Enable watchdog timer
  c = inp(0x68);
  c |= 0x40;
  outp(0x68, c);
  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
    outp(0x67, 0x00);
  printf("System will reboot after 500 milli-seconds.\n");
  return 0;
}
```

WDT0 Windows CE Example

```
#include "stdafx.h"

unsigned char inportb(int addr)
{
  __asm
  {
    push edx
    mov edx, DWORD PTR addr
    in al, dx
    and eax, 0xff
    pop edx
  }
}

void outportb(int addr, unsigned char val)
{
  __asm
  {
    push edx
    mov edx, DWORD PTR addr
    mov al, BYTE PTR val
    out dx, al
    pop edx
  }
}

void main(void)
{
  unsigned char c;
  unsigned int lTime;

  outp(0x22,0x13); // Lock register
  outp(0x23,0xc5); // Unlock config. register

  // 500 mini-second
  lTime = 0x20L * 500L;
  outp(0x22,0x3b);
  outp(0x23,(lTime>>16)&0xff);
  outp(0x22,0x3a);
  outp(0x23,(lTime>> 8)&0xff);
  outp(0x22,0x39);
  outp(0x23,(lTime>> 0)&0xff);
```

```
  // Reset system
  outp(0x22,0x38);
  c = inp(0x23);
  c &= 0x0f;
  c |= 0xd0; // Reset system. For example, 0x50 to trigger IRQ7
  outp(0x22,0x38);
  outp(0x23,c);

  // Enable watchdog timer
  outp(0x22,0x37);
  c = inp(0x23);
  c |= 0x40;
  outp(0x22,0x37);
  outp(0x23,c);

  outp(0x22,0x13); // Lock register
  outp(0x23,0x00); // Lock config. register

  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
  {
    outp(0x22,0x13);  // Unlock register
    outp(0x23,0xc5);
    outp(0x22,0x3c);
    unsigned char c = inp(0x23);
    outp(0x22,0x3c);
    outp(0x23,c|0x40);
    outp(0x22,0x13);  // Lock register
    outp(0x23,0x00);
  }

  printf("System will reboot after 500 milli-seconds.\n");
}
```

WDT1 Windows CE Example

```
#include "stdafx.h"

unsigned char inportb(int addr)
{
  __asm
  {
    push edx
    mov edx, DWORD PTR addr
    in al, dx
    and eax, 0xff
    pop edx
  }
}

void outportb(int addr, unsigned char val)
{
  __asm
  {
    push edx
    mov edx, DWORD PTR addr
    mov al, BYTE PTR val
    out dx, al
    pop edx
  }
```

```
}

void main(void)
{
  unsigned char c;
  unsigned long lTime;

  // 500 mini-second
  lTime = 0x20L * 500L;
  outp(0x6c, (lTime >> 16) & 0xff);
  outp(0x6b, (lTime >>  8) & 0xff);
  outp(0x6a, (lTime >>  0) & 0xff);

  // Reset system. For example, 0x50 to trigger IRQ7
  outp(0x69, 0xd0);

  // Enable watchdog timer
  c = inp(0x68);
  c |= 0x40;
  outp(0x68, c);

  printf("Press any key to stop trigger timer.\n");
  while(!kbhit())
    outp(0x67, 0x00);

  printf("System will reboot after 500 milli-seconds.\n");

}
```

Technical Support

For more technical support, please visit http://www.dmp.com.tw/tech or mail to tech@dmp.com.tw.