Pegasus Watchdog Timer Application Note

January 19, 2010

1.1 Watchdog Timer Interface

The watchdog is a timer designed to reset the CPU or generate an interrupt if the system comes to a standstill for any unknown reason. This is useful in applications where the single board computer will be used in an unmanned or standalone situation.

The HiCORE-i6413DVL has a build-in watchdog timer. The super I/O chip, WINBOND® W83627HF, implements it.

The watchdog timer contains a 1-second/minute resolution down counter. The down counter can be programmed within the range from 1 to 255 seconds/minutes. Writing any new non-zero value to the down counter will cause the watchdog timer to reload and start to count down the new value. As the counter reaches zero, the system will be reset or an interrupt generated, which is determined by the time-out event configuration.

1.2 Configuring the watchdog timer

The system accesses the super I/O chip through Intel® LPC (Low Pin Count) interface. A fixed I/O ranged 2E~2Fh supports index access for super I/O configures.

The I/O chip, W83627HF uses compatible PnP protocol to access configuration registers for setting up variable configurations. In W83627HF, there are eleven logic devices. Each logic device has its own configuration registers (above CR30). Host can access those registers by writing an appropriate logic device number into device select register CR7. Watchdog timer shares the same logic number 8 with GPIO port 2.

To program watchdog timer or other W83627HF configuration registers, the following configuration sequence must be followed:

(1) Enter the extended function mode

To place the chip into the extended function mode, two successive writes of 0x87 must be applied to Extended Function Enable Registers (EFERs, i.e. 0x2E).

The following example is encoded with Turbo C 2.0. Symbol Superio_Config_Port is the address of EFER and must be predefined as a constant 0x2E.

```
/*****************************************************
 *   Enter Logic Device Program Mode
 *****************************************************/
int Superio_Enter_Config(void)
{
     outp(Superio_Config_Port, 0x87);
     outp(Superio_Config_Port, 0x87);
     return    0;
}
```

Configure the configuration registers

The chip selects the logical device and activates the desired logical devices through Extended Function Index Register (EFIR) and Extended Function Data Register (EFDR). EFIR is located at the same address as EFER (0x2E), and EFDR is located at address (EFIR+1).

```
#define NEWIODELAY() asm OUT 0EBh,AL   /* delay for I/O access */

int Superio_Set_Reg(int RegInx,  int ReGEal)
{
     outp(Superio_Config_Port, RegInx);
     NEWIODELAY();
     outp(Superio_Config_Port+1, ReGEal);
     NEWIODELAY();
     return    0;
}

int Superio_Get_Reg(int RegInx)
{
     int  ReGEal;
    outp(Superio_Config_Port, RegInx);
    NEWIODELAY();
    ReGEal = inp(Superio_Config_Port+1);
    NEWIODELAY();
    return    ReGEal;
}
```

First, write the device select register number (0x07) to the EFIR and then write the number of the desired logical device (0x08 for watchdog timer/GPIO port 2) to the EFDR. If accessing the Chip (Global) Control Registers, this step is not required.

Invoking following routine performs a logic device selection for watchdog. The entry parameter LgcDevNum is predefined as 0x8.

```
int Set_Logic_Device(int LgcDevNum)
{
   Superio_Set_Reg(0x7, LgcDevNum);  /* LgcDevNum=8 for watchdog */
   return 0;
}
```

Secondly, write the address of the desired configuration register within the logical device to the EFIR and then write (or read) the desired configuration register through EFDR.
The detail of watchdog configuration register programming will be described in the next paragraph.

Exit the extended function mode.

To exit the extended function mode, one write of 0xAA to EFER is required. Once the chip exits the extended function mode, it is in the normal running mode and is ready to enter the configuration mode.

```
/************************************************************
 *   Exit Logic Device Program Mode
 ************************************************************/
int Superio_Exit_Config(void)
{
     outp(Superio_Config_Port, 0xaa);
     return    0;
}
```

1.3 The detail of watchdog programming

The Watchdog timer output pin, WDTO shares the same physical pin with GPIO24. The status of GPIO24 configuration registers must be programmed to a known value whatever the application configures the watchdog time-out event as system reset or interrupt.

(2) Configure watchdog time-out event

The watchdog can be configured as system reset output or generate an interrupt if the system comes to a standstill for any unknown reason. If it is set as system interrupt, the following lines, predefinitions must be implemented.

```
#define   WDTIRQMod
/* Select time-out event IRQ number (0 to 15, 2 for SMI) */
#define   IRQSource 5/*eg. Select Watchdog interrupt connect to IRQ5*/
```

The counter resolution of the watchdog timer should be predefined too. WDTCntMod=0 is for 1-sec resolution and 1 for 1-min resolution.

```
#define   WDTCntMod 0    /* 0 -- 1-second resolution */
                          /* 1 -- 1-minute resolution */
```

The following is an example to initialize the watchdog.

```
int ConfigWDT(void)
{
    int  iRetVal;

    /* Enter super I/O chip configuration mode */
    Superio_Enter_Config();

    /* Select logic device 8, watchdog to configure */
    Set_Logic_Device(0x8);

    /* Configure GPIO24 as output pin */
    iRetVal = Superio_Get_Reg(0xf0);
    iRetVal &= ~0x10;  /* clear bit4, GPIO24 as output */
    Superio_Set_Reg(0xf0, iRetVal);

    /* Configure GPIO24 output LOW level */
    iRetVal = Superio_Get_Reg(0xf1);
    iRetVal &= ~0x10;  /* clear bit4, GPIO24 output 0 */
    Superio_Set_Reg(0xf1, iRetVal);

    /* Configure GPIO24 output non-inversion */
    iRetVal = Superio_Get_Reg(0xf2);
    iRetVal &= ~0x10;  /* clear bit4, GPIO24 non-inversion */
    Superio_Set_Reg(0xf2, iRetVal);
#ifdef   WDTIRQMod

   /* Select GPIO/WDTO pin as GPIO */
   iRetVal = Superio_Get_Reg(0x2b);
   iRetVal |= 0x10;     /* Set bit4,Select GPIO/WDTO pin as GPIO */
   Superio_Set_Reg(0x2b, iRetVal);

   /* Note: Application should provide an interrupt service */
   /* Select time-out event IRQ number (0 to 15, 2 for SMI) */
   Superio_Set_Reg(0xf7, IRQSource);

#else

   /* Select GPIO/WDTO pin as WDTO */
   iRetVal = Superio_Get_Reg(0x2b);
   iRetVal &= ~0x10;    /* Clear bit4, Select GPIO/WDTO pin as WDTO */
   Superio_Set_Reg(0x2b, iRetVal);

#endif

   /* Select watchdog timer count mode (sec/min) */
   iRetVal = Superio_Get_Reg(0xf5);
   iRetVal &= ~0x8;          /* Count mode config bit */
   if (WDTCntMod)
      iRetVal |= 0x8;
   Superio_Set_Reg(0xf5, iRetVal);

   /* Set watchdog time-out value, disabled */
   Superio_Set_Reg(0xf6, 0);

   /* Exit super I/O chip configuration mode */
   Superio_Exit_Config();

   return 0;
}
```

Enable/Refresh the watchdog timer

The following codes show how to refresh the watchdog timer. It must be invoked at least once every cycle in application. The entry parameter iTimOutVal is ranged 0x1 to 0xff. The watchdog timer should be initialized before it is enabled or refreshed.

```
int RefWDT(int iTimOutVal)
{
    /* Enter super I/O chip configuration mode */
    Superio_Enter_Config();

    /* Select logic device 8 to configure */
    Set_Logic_Device(0x8);

    /* Set watchdog time-out value, disabled */
    Superio_Set_Reg(0xf6, iTimOutVal);

    /* Exit super I/O chip configuration mode */
    Superio_Exit_Config();

    return    0;
}
```

Disable the watchdog timer

Invoke `RefWDT()` with parameter `iTimOutVal=0` will disable the watchdog timer.

```
    RefWDT(0x0);
```

How to check watchdog timer status

If the watchdog is configured as a system time-out reset. Bypass the section.

If the watchdog time-out event is configured as a system interrupt, the application program should handle the preset IRQ and provide an interrupt service routine. Following routine shows how to check if the generated interrupt is required from watchdog.

```
int ChkWdtIrq(void)
{
    int    iRetVal;

    /* Enter super I/O chip configuration mode */
    Superio_Enter_Config();

    /* Select logic device 8 to configure */
    Set_Logic_Device(0x8);

    /* Check watchdog timer status */
    iRetVal = Superio_Get_Reg(0xf7);
    iRetVal &= 0x10;    /* Check bit4, 1 - Time-out event occurred */

    /* Exit super I/O chip configuration mode */
    Superio_Exit_Config();

     if   (iRetVal)
          return    1;   /* Watchdog time-out occurred */
    return    0;        /* Watchdog timer counting */
}
```