XMOS

AN02045: Using the watchdog on xcore

Publication Date: 2025/6/5
Document Number: XM-015337-AN v1.0.0

IN THIS DOCUMENT

xcore processors have a *watchdog* that, when enabled, will automatically reset the device after a set amount of time. The purpose of the watchdog is to set it up from a core real-time loop in the system. If the real-time loop halts due to an error, then the watchdog will reset the device back to a known state.

1 Watchdog configuration

The watchdog supervises the entire node, that is, the switch, the two tiles connected to the switch (on *xcore.ai*), the PLL, etc. The watchdog is set by writing to three configuration registers in the node.

The registers that need to be set are:

- **WATCHDOG_PRESCALER_WRAP**: this register sets the granularity of watchdog ticks. The watchdog is driven by the input to the PLL, and the pre-scaler divides the input clock to provide a watchdog-clock. This register is typically written once on setting up the watchdog, and should be set to the desired divider value minus 1.
  For example, if the input clock to the chip is 24 MHz, and the pre-scaler wrap is set to 23 (24 minus one), then the watchdog will be clocked at 1 MHz.
- **WATCHDOG_COUNT**. This register is set up to count how many clock ticks until the chip should be reset. It should initially be set to a safe, high, value. Once the watchdog is enabled, it should be reset to a cut-off value inside a critical loop.
  For example, suppose that the inner loop should execute at 10 kHz (100 us), and the pre-scaler is set up so that the watchdog clocks at 1 MHz, then the watchdog count could be to, say, 120 so that if the next iteration does not happen within 120 us the chip will be reset.
- **WATCHDOG_CFG**: this register enables the watchdog to be enabled. Two bits should be written in the watchdog, one to set the counter going, and one to enable reset when the counter gets to 0.

2 Setting the watchdog

The example program flashes the LEDs (marked “0 1 2 3”) connected to port 4C with a periodicity of 5 Hz. This port is named `PORT_LEDS` in the *XK-EVK-XU316* XN file.

After 10 flashes it simulates an error by waiting 200 us rather than 100 us.

The code to set up the watchdog timer is shown below:

```
    unsigned id = get_local_tile_id();
    write_sswitch_reg(id, XS1_SSWITCH_WATCHDOG_PRESCALER_WRAP_NUM, 23999); // 1 ms ticks
    write_sswitch_reg(id, XS1_SSWITCH_WATCHDOG_COUNT_NUM, 0xFFF);
    int enable = 0;
    enable = XS1_WATCHDOG_COUNT_ENABLE_SET(enable, 1);
    enable = XS1_WATCHDOG_TRIGGER_ENABLE_SET(enable, 1);  // Comment out to not trigger
    write_sswitch_reg(id, XS1_SSWITCH_WATCHDOG_CFG_NUM, enable);
```

In this example the code to perform the time critical operation is trivial: the LEDs are flashed at a frequency of 5 Hz by waiting for 100 ms on each toggle.

```
        port_out(leds, led_value);
        led_value = ~led_value;
        time += 10000000;   // wait for 100 ms
        hwtimer_set_trigger_time(tmr, time);
        (void) hwtimer_get_time(tmr);
```

After switching it off 10 times and off 10 times a timing failure is simulated by waiting for an extra 100 ms:

```
        cnt++;
        if (cnt == 20) {
            time += 10000000;    // Wait for an extra 100 ms
        }
```

3 Example application

3.1 Building the example

This section assumes you have downloaded and installed the XMOS XTC tools (see *README* for required version). Installation instructions can be found here.

Be sure to pay attention to the section Installation of required third-party tools.

The application uses the xcommon-cmake build system as bundled with the XTC tools.

The `an02045` software zip-file should be downloaded and unzipped to a chosen directory.

To configure the build run the following from an XTC command prompt:

```
cd an02045
cd app_an02045
cmake -G "Unix Makefiles" -B build
```

All required dependencies are included in the software download, however, if any are missing it is at this configure step that they will be downloaded by the build system.

Finally, the application binaries can be built using `xmake`:

```
xmake -j -C build
```

This command will cause a binaries (.xe files) to be generated in relevant subdirectories of the *app_an02045/bin* directory, one for each of the build configurations.

3.2 Running the example

From a XTC command prompt run the following command from the *an02045/app_an02045* directory:

```
xrun --io ./bin/app_an02045.xe
```

It should be observed that the LEDs marked “0 1 2 3” all flash 10 times, and then stop. The reason for this is that the watchdog has intervened and reset the device.

Note: Since, on reset, the device will execute any program in flash it is recommended that, before performing the previous experiment, the flash be erased using the command:

```
xflash --target XK-EVK-XU316 --erase
```

In a realistic, deployment situation, the program would run from flash. The binary can be programmed into flash with the following command:

```
xflash ./bin/app_an02045.xe
```

It should be observed that the LEDs flashes 10 times and a small, irregular delay occurs, whereupon the LEDs flashes 10 times again, and another delay. This shows that on intervention by the watchdog the device resets, reboots, and the program re-starts.

4 Interference between the watchdog and the tools

Once a binary that enables the watchdog has been flashed, it may be difficult to flash or run programs, as the watchdog may kill the debugger connection.

In this case, the following environment variables should be set and the flash erased to remove the program:

```
export XDBG_WATCHDOG_DISABLE=1
xflash --target XK-EVK-XU316 --erase
unset XDBG_WATCHDOG_DISABLE
```

Setting this environment variable will avoid the interference.

5 Debugging with the watchdog enabled

It is difficult to debug programs with a watchdog; on reset it will lose connection with the host. For this reason, the watchdog can be changed to count but not reset the device. For this purpose, comment out the following line:

```
    enable = XS1_WATCHDOG_TRIGGER_ENABLE_SET(enable, 1);  // Comment out to not trigger
```

For debugging, the `XS1_SSWITCH_WATCHDOG_COUNT_NUM` register can be read the counter should be observed to be counting down to zero.