This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

Using hardware and software to make new stuff

Search

type, hit enter

Categories

- 3D Printing (1)
- Affinity Photo (1)
- Aide-memoir (1)
- Analog (1)
- Ansible (3)
- Dates to Remember (3)
- Docker (1)
- Electronics (144)
- ESP8266 (12)
- FPGA (4)
- Garden (1)
- General (8)
- Home Built CPU (6)
- Informatica (1)
- Internet of Things (8)
- iOS (2)
- KiCad (4)
- MSP430 (2)
- Netduino (49)
- NuttX (9)
- Photography (3)
- Pico (10)
- Raspberry Pi (17)
- Silverlight (6)
- Software Development (88)
- STM32 (5)
- STM8 (54)
- Tips (5)

Archives

- July 2024
- June 2024
- May 2024
- April 2024
- March 2024
- February 2024
- January 2024
- December 2023
- November 2023
- October 2023

This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

- April 2023
- February 2020
- January 2020
- July 2019
- February 2018
- July 2017
- June 2017
- April 2017
- March 2017
- February 2017
- October 2016
- September 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- January 2016
- December 2015
- November 2015
- October 2015
- August 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- January 2013
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- June 2012
- May 2012

This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

- October 2011
- September 2011
- July 2011
- June 2011
- May 2011
- April 2011
- March 2011
- February 2011
- January 2011
- December 2010
- April 2010

STM8S Independent Watchdog

The Independent Watchdog (IWDG) allows the developer to detect software failures and reset the STM8S microcontroller restarting the application. This post demonstrates how this feature works and how the developer can change the time window for the watchdog.

Watchdogs offer the application developer a simple to use method of determining if an application has entered a state which means that it is locked or taking too long to perform an operation.

The first task is to define what we mean by *too long*. This will vary from one application to another but the key point is that this should be longer than the longest operation the application will perform plus a margin. Once we have this time period defined we know that the application must reset the IWDG before the time period has elapsed.

Hardware

Independent Watchdog Key Register – IWDG_KR

The Key Register controls the operation of the IWDG. The allowed values are as follows:

Value	Description
0x55	Enable writing to the IWDG_PR and IWDG_RLR registers. These registers are write protected by default.
0xAA	Reset the IWDG counter to the reload value. This value must be written to the key register periodically in order to prevent the watchdog from resetting the microcontroller.
0xCC	Enable and start the IWDG.
0x00	Disable writing to the IWDG_PR and IWDG_RLR registers.

The final value in the table is not present in revision 9 of *RM0016 – Reference Manual* but does appear in the header files for the *STD Peripheral Library*.

In researching this post a key point in the documentation was missed, namely **IWDG_PR and IWDG_RLR can only be written when the IWDG has actually been enabled**. This means that the IWDG needs to be running before the prescaler and reload registers can be loaded with the values required by the application. If the IWDG is not running then the system will resort to the default reset values for these registers. This gives a watchdog window of 16ms. It also means that the application has 16ms to set the desired time window through the IWDG_PR and IWDG_RLR registers before the microcontroller is reset.

Independent Watchdog Prescaler – IWDG_PR and Independent Watchdog Reload Register – IWDG_RLR

This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

Prescaler Divider	Prescaler Value (IWDG_PR)	Period when RL = 0	Period when RL = 0xFF
4	0	62.5 us	15.90 ms
8	1	125 us	31.90 ms
16	2	250 us	63.70 ms
32	3	500 us	127 ms
64	4	1.00 ms	255 ms
128	5	2.00 ms	510 ms
256	6	4.00 ms	1.02 s

The programming reference for the STM8S gives the following formula for determining the exact reset period:

$T = 2 \times T_{LSI} \times P \times R$

where:

T	Timeout period
T_{LSI}	$1 / f_{LSI}$
P	$2^{(IWDG\_PR + 2)}$
R	IWDG_RLR + 1

Additionally, the time between the last reset of the key register (i.e. writing 0xAA to IWDG_KR) is $T + 6 \times T_{LSI}$.

Software

So lets look at how we can implement this:

```
1  //
2  //  This program demonstrates how to use the Independent Watchdog timer
3  //  on the STM8S microcontroller in order to detect software failures and
4  //  reset the microcontroller.
5  //
6  //  This software is provided under the CC BY-SA 3.0 licence.  A
7  //  copy of this licence can be found at:
8  //
9  //  http://creativecommons.org/licenses/by-sa/3.0/legalcode
10 //
11 #include <iostm8S105c6.h>
12 #include <intrinsics.h>
13
14 //-------------------------------------------------------------------------
15 //
16 //  Setup the system clock to run at 16MHz using the internal oscillator.
17 //
18 void InitialiseSystemClock()
19 {
20     CLK_ICKR = 0;                       //  Reset the Internal Clock Register
21     CLK_ICKR_HSIEN = 1;                 //  Enable the HSI.
22     CLK_ECKR = 0;                       //  Disable the external clock.
23     while (CLK_ICKR_HSIRDY == 0);       //  Wait for the HSI to be ready for
24     CLK_CKDIVR = 0;                     //  Ensure the clocks are running at
25     CLK_PCKENR1 = 0xff;                 //  Enable all peripheral clocks.
26     CLK_PCKENR2 = 0xff;                 //  Ditto.
```

This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

```
    CLK_SWCR_SWEN = 1;                  //  Enable switching.
    while (CLK_SWCR_SWBSY != 0);        //  Pause while the clock switch is b
}

//-------------------------------------------------------------------------
//
//  Initialise the ports.
//
//  Configure all of Port D for output.
//
void InitialisePorts()
{
    PD_ODR = 0;             //  All pins are turned off.
    PD_DDR = 0xff;          //  All pins are outputs.
    PD_CR1 = 0xff;          //  Push-Pull outputs.
    PD_CR2 = 0xff;          //  Output speeds up to 10 MHz.
}
```

The code above should be pretty familiar to anyone who has been following The Way of the Register series of posts. This sets up the system clock to run at 16MHz and sets port D as an output port.

```
//-------------------------------------------------------------------------
//
//  Initialise the Independent Watchdog (IWDG)
//
void InitialiseIWDG()
{
    IWDG_KR = 0xcc;         //  Start the independent watchdog.
    IWDG_KR = 0x55;         //  Allow the IWDG registers to be programmed.
    IWDG_PR = 0x02;         //  Prescaler is 2 => each count is 250uS
    IWDG_RLR = 0x04;        //  Reload counter.
    IWDG_KR = 0xaa;         //  Reset the counter.
}
```

We follow this up by setting up the IWDG. The prescaler and counter values should give a 1ms window.

This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

```
int main()
{
    //
    //  Initialise the system.
    //
    __disable_interrupt();
    InitialiseSystemClock();
    InitialisePorts();
    InitialiseIWDG();
    __enable_interrupt();
    __halt();
}
```

The main program configures the system and then halts the microcontroller. Without the watchdog this would the end of the application. With the watchdog we should see the microcontroller continuously resetting. This can be verified by connecting an oscilloscope to Port D. If we do this we see the output on the port rise as the microcontroller is reset and the *InitialisePorts* method is called.

Now replace the *__halt();* statement with the following code:

```
while (1)
{
    IWDG_KR = 0xaa;
}
```

This code continuously writes the *I am alive* message to the IWDG key register. Doing this forces the IWDG to reset the counters with the reload values and start again. Running this application we see a flat line following the initial reset of the microcontroller. The flat line indicates that the microcontroller is not being reset.

Conclusion

The Independent Watchdog provides a simple method for detecting software failures merely writing a reset value into the key register.

Earlier the point was made that it appears that the IWDG must be enabled (and hence running) before the prescaler and reload registers can be modified. This point is critical as the microcontroller will resort to the reset values otherwise.

Although not demonstrated here it is possible for the application to determine if the application has been started through the application of power to the microcontroller or because of a reset by the IWDG. This can be determined by checking the value in *RST_SR_IWDGF*. This will be set to 1 if the IWDG has caused the microcontroller to be reset or 0 otherwise. This is left as an exercise for the reader.

Download

The source code for this post is available for download.

Tags: Electronics, Software Development, STM8, The Way of the Register

Saturday, June 21st, 2014 at 2:35 pm • Electronics, Software Development, STM8 • RSS 2.0 feed Both comments and pings are currently closed.

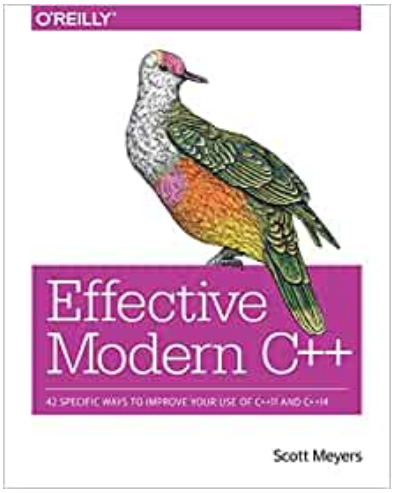
Comments are closed.

This website uses cookies to improve your experience and to gather page view statistics. This site does not collect user information. Accept & Close **Read More**

- The Way of the Register
- Making an IR Remote Control
- Weather Station Project
- NuttX and Raspberry Pi PicoW

Support This Site

Currently Reading



© 2010 - 2024 Mark Stevens