Watchdog Timer

Reference Material

Material in this document was drawn from these three sources

1. Watchdog Timer Basic Example, Written by Nicolas Larsen, 10 June 2011
2. ATmega data sheets

   ATmega48PA/88PA/168PA/328P Section 10.8 Watchdog Timer (page 50 / 448)

   ATmega16U4/ATmega32U4 Section 8.2 Watchdog Timer (page 48 / 448)
3. Standard C library for AVR-GCC avr-libc wdt.h library

   You can find this wdt.h file in the Arduino\hardware\tools\avr\avr\include\avr folder

The Basics

-  The watchdog timer watches over the operation of the system. This may include preventing runaway code or in our C example, a lost communications link.

- The watchdog timer operates independent of the CPU, peripheral subsystems, and even the clock of the MCU.

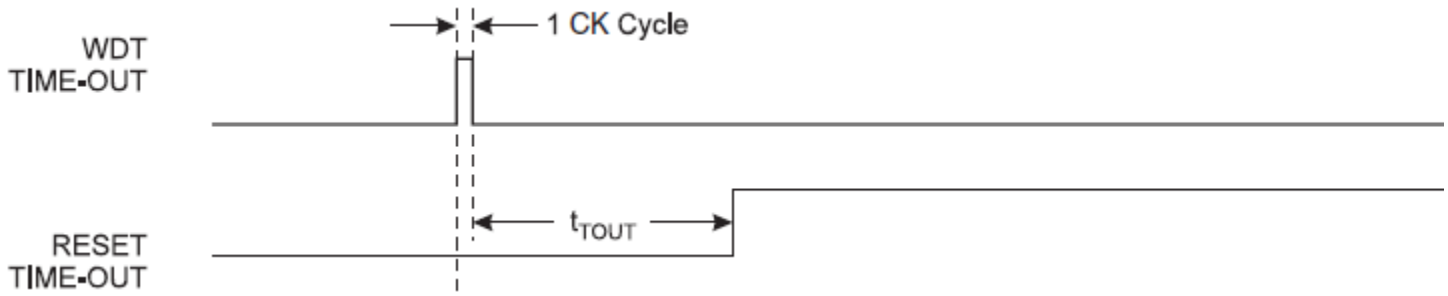
- To keep the watchdog happy you must feed  it a `wdr` (watchdog reset) assembly instruction before a predefined timeout period expires.

- The timeout period is defined by a ~128KHz watchdog timer clock and a 4-bit prescaler.

Watchdog Timer Reset

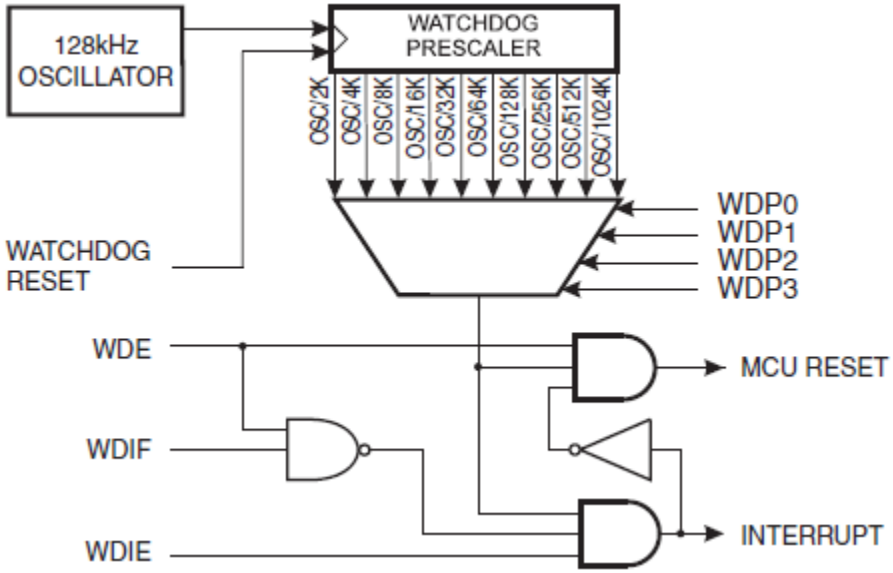
In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued. ATmega328P Datasheet Section 10.8.2 Overview.

When the Watchdog Reset (`wdr`) instruction is encountered (pun intended), it generates a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}.



The Watchdog Timer System

- To configure the watchdog timer you define the timeout period by setting the pre-scale value, and define what action is to be taken if a timeout occurs.



- Configuration bits are found in the WDTCSR – Watchdog Timer Control Register.

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

Define the Timeout Period

- The WDP3..0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is running. The different prescaling values and their corresponding time-out periods are shown here.

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s

One your own…

How many flip-flops are needed to implement the watchdog prescaler?

Hint: How many bits are needed to generate the longest delay with an input clock frequency of 128KHz?

Define Action on Timeout

- The Watchdog always on (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed (WDTON = 0) the System Reset mode bit (WDE) and mode bit (WDIE) are locked to 1 and 0 respectively. Arduino / ATmega 328P fuse settings.
- The Arduino ATmega328P bootloader sets the fuse to unprogrammed WDTON = 1, which means you can program the action to be taken by setting or clearing the WDE and WDIE bits as shown in the following table.

WDTON[1]	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	x	x	System Reset Mode	Reset

- Watchdog Timer is in Interrupt and System Reset Mode – When the interrupt occurs the hardware automatically clears the WDIE bit, thus putting the watchdog timer into the "System Reset" mode, as defined in the table (WDTON = 1, WDIE = 0, WDE = 1). At the next timeout, a reset is generated.

Assembly Code Example

- The following assembly code example Starts the watchdog timer in System Reset Mode with a timeout period of ~0.5 seconds

```
WDT_Prescaler_Change:
cli                         ; Turn off global interrupt
wdr                         ; Give yourself some time
lds r16, WDTCSR             ; Start timed sequence
ori r16, (1<<WDCE) | (1<<WDE)
sts WDTCSR, r16
; -- You have four cycles to set the new values from here --
; Set mode and new prescaler(time-out) value = 64K cycles (~0.5 s)
ldi r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
sts WDTCSR, r16
; -- Finished setting new values, used 2 cycles --
sei                         ; Turn on global interrupt
ret
```

Assembly and C code Examples Turning Off the Watchdog Timer.

Assembly Code Example

```
WDT_off:
  cli             ; Turn off global interrupt
  wdr             ; Reset Watchdog Timer
  in r16, MCUSR   ; Clear WDRF in MCUSR
  andi r16, (0xff & (0<<WDRF))
  out MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional time-out[1]
  lds r16, WDTCSR
  ori r16, (1<<WDCE) | (1<<WDE)
  sts WDTCSR, r16
  ldi r16, (0<<WDE) ; Turn off WDT
  sts WDTCSR, r16
  sei ; Turn on global interrupt
  ret
```

C Code Example

```
void WDT_off(void)
{
  cli();                // disable interrupts
  wdt_reset();          // included in avr/wdt.h library, assembly instruction wdr
  MCUSR &= ~(1<<WDRF); // Clear WDRF in MCUSR
  // Write logical one to WDCE and WDE
  // Keep old prescaler setting to prevent unintentional time-out[1]
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  WDTCSR = 0x00;        // Turn off WDT
}
```

[1] If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the Watchdog Timer will stay enabled. If the code is not set up to handle the Watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the Watchdog System Reset Flag(WDRF) and the WDE control bit in the initialisation routine, even if the Watchdog is not in use.

3DoT C Code Watchdog Object

- The 3DoT Watchdog object has only one public methods

```
void watchdogSetup(uint8_t);
```

- The 3DoT Watchdog object has three private methods

```
void watchdogOff();
void throwError(uint16_t);
```

- The 3DoT Watchdog object has three read-only private properties

```
uint8_t _prescaler;
uint8_t _mode;
uint8_t _counter;
```

- Let's take a closer look at the `watchdogSetup` method

3DoT C Code Operation

- In the `watchdogSetup` C program on the next page the ATmega328P WDTCSR register may be configured to operate the watchdog timer in the “Interrupt and System Reset” or “Interrupt” mode with a programmable delay from 1 to 8 seconds.

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

- To configure the WDT a `0x10` WATCHDOG_SETUP command packet is sent with one of the following arguments

```
0x00        Watchdog Off
0x4E        1 sec interrupt and system reset mode
0x4F        2 sec
0x68        4 sec
0x69        8 sec
0x46        1 sec interrupt mode
0x47        2 sec
0x60        4 sec
0x61        8 sec
```

- If one of these arguments is not sent the program sends a 0x0E “Exception” packet with a 0x06 “Watchdog timeout out of range” code. To put this in perspective, here are all the Exception codes and what they mean.

```
0x0E Exception Codes
     High        low order byte
     01          Start byte 0xA5 expected
     02          Packet length out of range 1 - 20
     03          LRC checksum error
     04          Undefined command decoder FSM state
     05          Array out of range i >= 23
     06          Watchdog timeout out of range
```

3DoT C Code Operation Example

- If programmed for 8 second “Interrupt and Reset” Mode and a `wdr` command is not generated within the timeout period, an Interrupt will occur at T+8 seconds and the Reset at T+16.

- When the interrupt occurs T+8 seconds…
  - the hardware automatically clears the WDIE bit, thus putting the watchdog timer into the "System Reset" mode, as defined in the table (WDTON = 1, WDIE = 0, WDE = 1).
  - a 0x0B “Emergency” packet with 0x0100 code is sent

```
0x0B Emergency Codes
High  Low order byte
01    00  Watchdog timeout
```

  - After this interrupt, at any time (up to T+16) you can reset the timer, turn it off, change modes, etc.

3DoT Watchdog C Code Example

```
/*
 * Watchdog Timer Interrupt
 */
ISR(WDT_vect) //
{
  // Safe 3DoT
  motorDriver.motors_safe();
  wdtPacket.sendPacket(EMERGENCY_ID,WATCHDOG_TIMEOUT); // send EMERGENCY_ID with WATCHDOG_TIMEOUT
}

/*
 * Watchdog Setup
 */
void Watchdog::watchdogSetup(uint8_t mode_prescaler)
{
  _prescaler = ((mode_prescaler & 1<<WDP3)>>2) | (mode_prescaler & 0x07); // extract prescaler WDP3..WDP0
  _mode = (((1<<WDIE) & mode_prescaler)>>5) | (((1<<WDE) & mode_prescaler)>>3);  // extract mode WDIE:WDE
  if (mode_prescaler == 0x00){
    watchdogOff();    // turn off watchdog timer
  }
  else if ((WDTO_1S <= _prescaler) && (_prescaler <= WDTO_8S)) // only allowable prescale values
  {
    cli();              // __disable_interrupt();
    wdt_reset();        // __watchdog_reset(); included in avr/wdt.h library, assembly instruction wdr
    // enter Watchdog Configuration mode
    // keep old prescaler setting to prevent unintentional time-out
    WDTCSR |= (1<<WDCE) | (1<<WDE);
    // Interrupt and System Reset mode (see Table 10-1) plus Prescaler (see Table 10-2)
    // timed instuction (4 cycles max)
    WDTCSR = mode_prescaler;
    sei();             // __enable_interrupt();
  }
  else
  {
    throwError(word(0x06,mode_prescaler)); // send 0x0E with code 0x06 plus undefined argument
  }
}
```

3DoT Watchdog Demonstration

- Plug in an Arduino UNO
- Launch and Configure CoolTerm
- Launch arxrobot_firmware_3DoT
- Normal Operation

```
A5 02 10 69 DE          Set watchdog interrupt for 8 sec
A5 01 11 B5             Ping (repeat at a frequency of less than 0.125 seconds)
CA 01 11 DA             Pong
A5 02 10 00 B7          Turn Watchdog Off
```

- Timeout Example

```
A5 02 10 4E 59          Set watchdog interrupt for 1 sec
CA 03 0B 01 00          Emergency Code 0B, Watchdog timeout 0100
CA 03 06 00 63 AC       Read and transmit sensor values after restart
CA 03 02 00 00 CB
```

- Timeout Prescaler out-of-range

```
A5 02 10 62 D5          ATmega reserved
CA 03 0E 06 62 A3
      ↓   |  ↓
exception |  argument
error     ↓
watchdog timeout out of range
```