

ATtiny85 Watchdog

Sources

- [ATtiny85 Datasheet, Chapter 8.4](#)
- [wolles-elektronikkiste.de/en/sleep-modes-and-power-management#Anker4](#) — seems to be the only working example for interrupts ...
- [avr/wdt.h](#)

Description

- The watchdog has its own, independent 128 kHz oscillator (timer) to monitor the MCU operation
- Watchdog modes

WDE	WDIE	Watchdog Timer State	Action on Time-out
0	0	Stopped	None
0	1	Running	Interrupt
1	0	Running	Reset
1	1	Running	Interrupt

- - **Stopped** — default, not active, no power consumption
 - **Interrupt mode** (WDE = 0) — generates an **interrupt** at time-out
 - **Watchdog / Reset mode** (WDE = 1) — generates an MCU **resets** at time-out **unless the counter is reset to zero before time-out** via `wdt_reset()`
 - WDIE = 1
 - **one "last-chance" interrupt** is raised, WDIE is set to 0 by hardware
 - **next time-out causes MCU reset** unless WDIE is set to 1 by interrupt handler
 - WDIE = 0
 - directly resets the MCU — this is the behaviour configured by `wdt_enable(timeout)`
- Disabling the watchdog requires a two-step sequence involving WDCE (Watchdog Change Enable) flag
 - Taken care of by `wdt_disable()`

Using the Watchdog

- [avr/wdt.h](#) ([Doxygen comments](#))

```
// Timeout bit-mask values
#define WDTO_15MS 0
#define WDTO_30MS 1
#define WDTO_60MS 2
#define WDTO_120MS 3
#define WDTO_250MS 4
#define WDTO_500MS 5
#define WDTO_1S 6
#define WDTO_2S 7
#define WDTO_4S 8
#define WDTO_8S 9

#define wdt_enable(timeout) // Will perform a device reset at timeout, e.g. if code entered infinite loop.
                           // User valus like this: wdt_enable(WDTO_500MS)

#define wdt_reset()        // Reset the watchdog timer. When the watchdog timer is enabled,
                           // a call to this instruction is required before the timer expires,
                           // otherwise a watchdog-initiated device reset will occur.

#define wdt_disable()
```

- **wdt_enable(timeout)**
 - Sets up a **true watchdog** that **will** perform an **MCU reset** after the given timeout unless `wdt_reset()` is called before timeout.
- ```
#include <avr/wdt.h> // Include the avr-libc Watchdog timer handling library

void setup() {
```

```

 wdt_enable(WDTO_8S); // Set watchdog timeout to 8 seconds
}

void loop() {
 wdt_reset(); // Reset the watchdog timer to zero before timeout --> call frequently :-)
}

```

- Set WDE = 1 and WDIE = 1, to generate an interrupt in place of an MCU reset:
  - **Reset WDIE = 1 after each "last-chance" interrupt**
  - This option is independent of the WDT control-register name which varies across AVR processors

```

wdt_enable(WDTO_1S);
WDTCSR |= _BV(WDIE);
sei();

```

```

ISR(WDT_vect) {
 WDTCSR |= _BV(WDIE);
}

```

- Alternatively: set WDE = 0 and WDIE = 1:

```

void setup_watchdog(uint8_t timeoutBitmask){
 cli()
 wdt_reset(); // Reset counter to zero
 WDTCSR = (1<<WDIE) | (0<<WDE) | timeoutBitmask; // Enable interrupt, no system reset
 // WDTCSR = (1<<WDIE) | (0<<WDE) | (1<<WDP2) | (1<<WDP1); // Alternative, 1 second
 sei();
}

// Invoke like this
setup_watchdog(WDTO_1S);

```

- **wdt\_disable()**  
[suggested usage](#) (see also: [ATtiny85 Datasheet, p.46](#), Note under "Bit 3 – WDE: Watchdog Enable")

```

MCUSR &= ~_BV(WDRF);
wdt_disable();

```