Using the watchdog timer with the ATTiny1614

Using the watchdog timer (WDT) library, declare a variable holding the value for the timeout period, then enable the WDT and keep resetting it using functions from the library.

```
#include <avr/wdt.h> // Include the watchdog timer library

/* Define watchdog timer reset period
0x0 OFF    -
0x1 8CLK   0.008s
0x2 16CLK  0.016s
0x3 32CLK  0.031s
0x4 64CLK  0.063s
0x5 128CLK 0.125s
0x6 256CLK 0.25s
0x7 512CLK 0.5s
0x8 1KCLK  1s
0x9 2KCLK  2s
0xA 4KCLK  4s
0xB 8KCLK  8s
other - Reserved
*/
const uint8_t reset_timeout = 0xB; //8-bit unsigned integer, maximum timeout is 8 seconds

void setup() {
  // Configure watchdog timer
  //wdt_enable(reset_timeout); //Enable watchdog timer to reset MCU after reset_timer seconds,
unless timer is reset before timeout period
 }

void loop() {
  // Reset the watchdog timer
  wdt_reset();  //Reset watchdog timer to avoid resetting the MCU
}
```

Using the watchdog timer (WDT) library, enable the WDT using one of several built-in macros and keep resetting it using functions from the library.

```
#include <avr/wdt.h> // Include the watchdog timer library

void setup() {
 // Configure watchdog timer
    wdt_enable(WDTO_2S); // Valid options are: WDTO_15MS, WDTO_30MS, WDTO_60MS,
//WDTO_120MS, WDTO_250MS, WDTO_500MS, WDTO_1S & WDTO_2S.  Note that WDTO_4S and
//WDTO_8S, although defined in wdt.h do not work with the ATTiny1614.
}

void loop() {
  // Reset the watchdog timer
  wdt_reset();  //Reset watchdog timer to avoid resetting the MCU
```

```
}
```

Don’t use the watchdog timer (WDT) library, use instead functions from the iotn1614.h library

```
/*  Available watchdog timer reset periods
   WDT_PERIOD_OFF_gc = (0x00<<0),  /* Off */
   WDT_PERIOD_8CLK_gc = (0x01<<0),  /* 8 cycles (8ms) */
   WDT_PERIOD_16CLK_gc = (0x02<<0),  /* 16 cycles (16ms) */
   WDT_PERIOD_32CLK_gc = (0x03<<0),  /* 32 cycles (32ms) */
   WDT_PERIOD_64CLK_gc = (0x04<<0),  /* 64 cycles (64ms) */
   WDT_PERIOD_128CLK_gc = (0x05<<0),  /* 128 cycles (0.128s) */
   WDT_PERIOD_256CLK_gc = (0x06<<0),  /* 256 cycles (0.256s) */
   WDT_PERIOD_512CLK_gc = (0x07<<0),  /* 512 cycles (0.512s) */
   WDT_PERIOD_1KCLK_gc = (0x08<<0),  /* 1K cycles (1.0s) */
   WDT_PERIOD_2KCLK_gc = (0x09<<0),  /* 2K cycles (2.0s) */
   WDT_PERIOD_4KCLK_gc = (0x0A<<0),  /* 4K cycles (4.1s) */
   WDT_PERIOD_8KCLK_gc = (0x0B<<0)  /* 8K cycles (8.2s) */
*/

void setup() {
   // Configure watchdog timer
   _PROTECTED_WRITE(WDT.CTRLA,WDT_PERIOD_8KCLK_gc); // no window, 8 seconds
 }

void loop() {
/*
  // Reset the watchdog timer
  __asm__ __volatile__ ("wdr"::);
}
```