[...]

Sensing Mode

Interrupts can be triggered by rising or falling edges (transitions between 0 and 1), or a short delay after a transition. You specify what kind of interrupt you want to catch using `EICRA`, the external interrupt control register A. Specifically you use the Interrupt Sense Control bits in that register.

Writing the ISR

ISR is a predefined macro in `interrupt.h`.

Arduino Simplicity

Arduino uses a simplified way of using interrupts:

```
void setup {
    attachInterrupt(0, foo, FALLING);
    […]
}

void foo() {
    // Called when external int 0 falls
}
```

This is much easier, but there are some things you can't do this way. E.g., the ISR allows you to attach to any vectors in the vector table, which you can't do with the Arduino function.

Interrupt Routines

In general, your interrupt routines should run quickly, to minimise the chance of an interrupt happening while you're in a routine. In your interrupt routine you have to decide whether to disable interrupts (in which case you miss any that happen during your routine) or not (and recursively descend into interrupt routines).

Any variable used in an ISR must be declared as `volatile`, since they can change in ways unknown to the compiler. The compiler can't rely on cached copies of the variable (e.g. in a register), and must fetch it again from memory. This is because the compiler can't predict variables changing due to interrupts.

Timers & Counters

Counter registers count up to their maximum value and then overflow and start again. An overflow flag is set when they overflow, and this can be checked manually or used to generate an interrupt.

To use a counter as a timer, it needs to be connected to a clock source. A timer is just a counter whose values are incremented by the clock.

You can clock the AVR timer via an external pin.

The frequency of the clock can vary with the supply voltage, though, slowing down as the voltage lowers. This can be a problem if a battery is depleting.

The watchdog timer can be used to reset your program if it gets into a strange state. Say you have some code that you know should never take more than 200 clock ticks – you can set the watchdog to 200 in between doing it, and if it runs out it means you got stuck, and it'll reboot the program.

Prescaling

Counter counts after some number of clock pulses, instead of every time.

Real Time

If the clock speed is 16MHz, then the clock period will be ~4.1 ms.

To get a specific interval (e.g. one second), you can find the value that the counter hits at that value, and make the microcontroller generate an interrupt when the counter reaches that value.

Timer Prescaling and Clear Timer on Compare (CTC)