```
int unitim(boolean inputswitch, int &state, unsigned long &start_time, int &set_reset, unsigned long &puls_now,
unsigned long millis_1, unsigned long millis_2, unsigned long pulses, int out_max, int mode, boolean fast)
 {
      boolean input;
      unsigned long now_time;
      unsigned long millis_3;
      millis_3 = millis_2;

      //convert periode lenght and %-dutycycle to two timer values in oscillator mode "8"
      if (mode == 8 || mode == 12 || mode == 14) {
        millis_2 = millis_1 * millis_2 / 100;
          millis_1 = millis_1 - millis_2;
      }

      //a front-end for overriding the input signal. Used in fixed length counter modes "11" to "14"
      if (mode == 11 || mode == 12 || mode == 13 || mode == 14) {
        switch (set_reset) {
         case 0://standby and ready
            input = inputswitch; //reflects the real input
            if (inputswitch == true) {
              set_reset = 1;
              puls_now = 0;
            }
            break;
         case 1: //wait here as long as we are counting up
            input = true; //force the input high as long as we count
            if (puls_now >= pulses) {
              input = false;
              set_reset = 2;
              puls_now = 0;
            }
            break;
         case 2: //finished and on hold in mode 13 and 14...
            input = false;
            puls_now = 0;
            if (mode == 13 || mode == 14) {
              if (inputswitch == false){ //...until inputswitch is FALSE
                set_reset = 0;
              }
            }
            else {
              set_reset = 0;
            }
            break;
         default:
            set_reset = 0;
            break;
        }
      }
      else {
          input = inputswitch;
          set_reset = 0;
          puls_now = 0;
      }

      // this is the state-switching mechanism that produces the timers/pulses

      if (fast == true){
        now_time = micros();
      }
      else{
        now_time = millis();
      }
```

```
    switch (state) {
       case 0:
        if (input == true) {
          state = 1;
          start_time = now_time; //set starttime to now_time = millis() OR micros()
        }
        break;
       case 1: //evaluate on-timer in these modes compared to now_time = millis() OR micros()
        if (now_time >= (start_time + millis_1) || mode == 6 || mode == 2 || mode == 0) {
          state = 2; //ON-timer run out, or skip ON-timer
        }
        if (input == false && (mode == 7 || mode == 5 || mode == 10)) { //evaluate input in these modes
          state = 0; //retrig during ON-timer
        }
        break;
       case 2: //stay here as long as input is true, and not in any osc. mode, incl. counters
        if (input == false || mode == 4 || mode == 8 || mode == 11 || mode == 12 || mode == 13 || mode == 14) {
          state = 3;
          start_time = now_time; //set starttime to now_time = millis() OR micros()
          if (mode == 11 || mode == 12 || mode == 13 || mode == 14) {
            puls_now = puls_now + 1; //increment the counter if in any countermode
          }
        }
        break;
       case 3: //evaluate off-timer in these modes compared to now_time = millis() OR micros()
        if (now_time >= (start_time + millis_2) || mode == 5 || mode == 1 || mode == 0) {
          state = 0; //OFF-timer run out, or skip OFF-timer
        }
        if (input == true && (mode == 7 || mode == 6 || mode == 9)) { //evaluate input in these modes
          state = 2; //retrig during OFF-timer
        }
        break;
       default:
        state = 0;
        break;
    }

    //force to min/max via forced states in oscillator mode "8" (for STP), and mode "12"
    if  ((mode == 8 || mode == 12 || mode == 14) && millis_3 == 100) {
      state = 3;
    }
    if  ((mode == 8 || mode == 12 || mode == 14) && millis_3 == 0) {
      state = 0;
    }

    //output according to state
    if (state == 2 || state == 3) {
      return out_max;
    }
    else {
      return 0;
    }
}
```