```
#include <avr/interrupt.h>
#include <Wire.h>
#include <Tiny4kOLED.h>

// --- Платкови пинове ---
const byte LED_RUN   = 1;      // PB1 – зеленият диод
const byte BTN_START = 4;      // PB4 – бутон Старт / Пауза
const byte BTN_RESET = 3;      // PB3 – бутон Нулиране (работи само на пауза)

// --- Глобални променливи ---
volatile uint32_t msElapsed = 0;  // натрупани милисекунди от старта
volatile bool     running   = false; // флаг дали таймерът върви

// ограничение срещу многократни натискания (debounce + anti spam)
uint32_t lastStartMs = 0;
uint32_t lastResetMs = 0;
const uint16_t LOCK_MS = 1000;      // минимум 1 сек между две събития

// координати за изписване на минути/секунди върху OLED а
const uint8_t X_MIN = 24;           // X позиция на минутите
const uint8_t X_SEC = X_MIN + 40;   // X позиция на секундите
const uint8_t Y_TOP = 0;            // най горна страница (0 та)

/* ──────────────────────────────────────────────────────
   Функция: updateField()
   - Чисти старите две цифри на дадено поле
   - Изписва новата стойност (с формат "%02u")
*/
void updateField(uint8_t value, uint8_t x) {
  char buf[3];
  oled.setFontX2(FONT8X16P);       // голям шрифт 16×32 px
  oled.setCursor(x, Y_TOP);
  oled.print(F("  "));           // два интервала = зачиства предишното
  snprintf(buf, sizeof buf, "%02u", value);
  oled.setCursor(x, Y_TOP);
  oled.print(buf);                // печатаме новата двойка цифри
}

/* ──────────────────────────────────────────────────────
   Функция: drawTime()
   - Опреснява минутите и секундите само ако са се променили
   - Двоеточието го рисувам веднъж и повече не го пипам
*/
void drawTime(uint8_t mm, uint8_t ss) {
  static uint8_t prevMM = 255, prevSS = 255;  // запомнени стари стойности
  static bool    colonDrawn = false;          // дали вече има двоеточие

  if (mm != prevMM) { updateField(mm, X_MIN); prevMM = mm; }
  if (ss != prevSS) { updateField(ss, X_SEC); prevSS = ss; }

  if (!colonDrawn) {
    oled.setFont(FONT8X16P);                // малък шрифт 8×16 px
    oled.setCursor(X_MIN + 32, 1);          // центрирам „:“ по Y
    oled.print(':');
    colonDrawn = true;
```

```
  }
}

/* ──────────────────────────────────────────────────────────
   setup() – еднократна инициализация
*/
void setup() {
  pinMode(LED_RUN, OUTPUT);
  pinMode(BTN_START, INPUT_PULLUP);
  pinMode(BTN_RESET, INPUT_PULLUP);

  // подкарвам OLED а
  oled.begin();
  oled.clear();
  drawTime(0, 0);    // стартово 00:00
  oled.on();

  // Таймер 1: режим CTC, делител /64 → 1 kHz = 1 ms тик
  TCCR1 = (1 << CTC1) | (1 << CS12) | (1 << CS11) | (1 << CS10);
  OCR1C = (F_CPU / 64000UL) - 1;   // 8 MHz→124, 1 MHz→14
  TIMSK |= (1 << OCIE1A);          // разрешавам прекъсване Compare A
  sei();                           // глобално разрешавам IRQ тата
}

/* ──────────────────────────────────────────────────────────
   loop() – главен цикъл, върти се постоянно
*/
void loop() {
  uint32_t nowMs = millis();   // системният таймер на Arduino ядрото

  // --- бутон Старт / Пауза ---
  static bool pStart = HIGH;           // предходно състояние на входа
  bool s = digitalRead(BTN_START);
  if (pStart && !s && (nowMs - lastStartMs >= LOCK_MS)) {
    running = !running;                // превключвам режима
    lastStartMs = nowMs;
  }
  pStart = s;

  // --- бутон Reset (работи само когато е на пауза) ---
  static bool pReset = HIGH;
  bool r = digitalRead(BTN_RESET);
  if (pReset && !r && !running && (nowMs - lastResetMs >= LOCK_MS)) {
    cli(); msElapsed = 0; sei();       // нулирам таймера атомарно
    lastResetMs = nowMs;
  }
  pReset = r;

  // копирам брояча без риск от прекъсване по средата
  uint32_t msCopy; cli(); msCopy = msElapsed; sei();

  // обновявам дисплея само когато секундата се промени
  static uint32_t prevSec = 0xFFFFFFFF;
  uint32_t tSec = msCopy / 1000;
  if (tSec != prevSec) {
```

```
    prevSec = tSec;
    drawTime(tSec / 60 % 100, tSec % 60);
  }

  // мигане на LED a – 4 Hz, само докато таймерът върви
  static uint16_t blink = 0;
  if (running) {
    if (++blink >= 250) { blink = 0; PINB |= _BV(LED_RUN); } // toggle
  } else {
    digitalWrite(LED_RUN, LOW);
    blink = 0;
  }
}

/* ——————————————————————————————————————
   ISR – прекъсване от Timer 1 на всеки 1 ms
*/
ISR(TIMER1_COMPA_vect) {
  if (running) ++msElapsed;  // броим милисекундите само когато таймерът е активен
}
```