**Listing 1. Procedura obsługi przerwania od układu czasowo-licznikowego i najważniejsze elementy programowego generowania PWM**

namespace **{**

//Vectors for PWM handler callbacks

std**::**list**<**soft\_pwm**\*>** g\_handlers**;**

std**::**atomic**<**bool**>** g\_locked**;**

volatile soft\_pwm**::**value\_type m\_pwm\_cnt**;**

**}**

//! Constructor

soft\_pwm**::**soft\_pwm**(** **)**

**{**

g\_locked **=** true**;**

**if(** g\_handlers**.**empty**()** **)**

**{**

timer\_create**();**

**}**

g\_handlers**.**push\_back**(** this **);**

g\_locked **=** false**;**

**}**

//! Destructor

soft\_pwm**::~**soft\_pwm**()**

**{**

g\_locked **=** true**;**

g\_handlers**.**remove**(** this **);**

**if(** g\_handlers**.**empty**()** **)**

**{**

timer\_destroy**();**

**}**

g\_locked **=** false**;**

**}**

//ISR handler

extern "C" **{**

\_\_attribute\_\_**((**interrupt**,**optimize**(**"-O3"**)))** void tim1\_up\_isr\_vector**()**

**{**

stm32**::**tim\_clear\_it\_pending\_bit**(** TIM1**,** TIM\_IT\_Update **);**

**if(** g\_locked **)** **return;**

**for(** auto h **:** g\_handlers **)**

**{**

h**->**isr\_handler**();**

**}**

**++**m\_pwm\_cnt**;**

**}**

**}**

**Listing 2. Publiczne API klasy soft\_pwm**

// !Replace pin and change pwm level

bool change\_pin**(** const out**&** new\_pin **);**

//! Change PWM bright value

void level**(** value\_type pwm **);**

//! Set fast mode

void fast**(** bool m **);**

**Listing 3. Publiczne metody klasy sterownika wyświetlacza**

/\*\* Put character on the display \*/

int putc**(** char ch **);**

/\*\* Set display position \*/

int setpos**(** int x**,** int **);**

//! Set brightness

void brightness**(** bright\_t val **);**

//! Get brightness

bright\_t brightness**()** const**;**

//! Clear the display

void clear**();**

**Listing 4. Fragmenty funkcji obsługi wyświetlacza**

namespace **{**

//Display digit config

const constexpr out ports**[][**10**]**

**{**

//H1

**{** out**(**out**::**A**,**10**),** out**(**out**::**A**,**11**),** out**(**out**::**A**,**12**)**

**},**

//H2

**{** out**(**out**::**A**,**0**),** out**(**out**::**A**,**1**),** out**(**out**::**A**,**2**),** out**(**out**::**A**,**3**),**

out**(**out**::**A**,**4**),** out**(**out**::**A**,**5**),** out**(**out**::**A**,**6**),** out**(**out**::**A**,**7**),**

out**(**out**::**A**,**8**),** out**(**out**::**B**,**13**)**

**},**

//M1

**{** out**(**out**::**C**,**10**),** out**(**out**::**C**,**11**),** out**(**out**::**C**,**12**),** out**(**out**::**C**,**13**),**

out**(**out**::**D**,**0**),** out**(**out**::**D**,**1**)**

**},**

//M2

**{** out**(**out**::**C**,**0**),** out**(**out**::**C**,**1**),** out**(**out**::**C**,**2**),** out**(**out**::**C**,**3**),**

out**(**out**::**C**,**4**),** out**(**out**::**C**,**5**),** out**(**out**::**C**,**6**),** out**(**out**::**C**,**7**),**

out**(**out**::**C**,**8**),** out**(**out**::**C**,**9**)**

**},**

//DOT lo

**{** out**(**out**::**B**,**15**)** **},**

// Dot hi

**{** out**(**out**::**B**,**14**)** **}**

**};**

//Off port marker

constexpr out port\_off**;**

**}**

// Handle nixie digit

void nixie\_disp**::**handle\_digit**(** int val**,** size\_t pos **)**

**{**

const dev**::**out**&** d **=** val**!=**e\_inval**?**ports**[**pos**][**val**]:**port\_off**;**

//! Wait for fadein fade out

**while** **(** m\_pwm**[**pos**].**change\_pin**(** d **)** **)**

**{**

isix**::**wait\_ms**(** 250 **);**

**}**

**}**