**Listing 1. Funkcje zapisu do wyświetlacza HEM6432-03 komend i danych obrazu**

void lcdHEM6432\_SendCmd**(**uint8\_t **\***pCommandBuf**,** uint8\_t nByte**)**

**{**

 i2c\_DataTypeDef lcdI2C**;** // i2c data structure

 lcdI2C**.**device\_addr **=** LCD\_I2C\_ADDR**;** // LCD i2c address (i.e. 0x7E)

 lcdI2C**.**data\_adr **=** LCD\_COMMAND\_REG**;** // LCD i2c command register (i.e. 0x00)

 lcdI2C**.**data\_size **=** nByte**;** // number of bytes to send

 lcdI2C**.**pdata **=** pCommandBuf**;** // command buffer address

 i2c\_WriteData**(&**lcdI2C**);** // send command(s) to LCD

**}**

void lcdHEM6432\_SendData**(**uint8\_t **\***pDataBuf**,** uint8\_t nByte**)**

**{**

 i2c\_DataTypeDef lcdI2C**;** // i2c data structure

 lcdI2C**.**device\_addr **=** LCD\_I2C\_ADDR**;** // LCD i2c address (i.e. 0x7E)

 lcdI2C**.**data\_adr **=** LCD\_DATA\_REG**;** // LCD i2c data register (i.e. 0x40)

 lcdI2C**.**data\_size **=** nByte**;** // number of bytes to send

 lcdI2C**.**pdata **=** pDataBuf**;** // display data buffer address

 i2c\_WriteData**(&**lcdI2C**);** // send display data to LCD

**}**

**Listing 2. Komendy i funkcja inicjalizacji wyświetlacza HEM6432-03**

// HEM6432-03 LCD module full initialization data

// (define LCD\_UPSIDE\_DOWN if LCD module orientation is inverted)

static const uint8\_t lcdInitData**[]=**

**{**

 LCD\_BIAS\_HIGH**,** // bias select: 1/5 bias @ 1/33 duty

#ifndef LCD\_UPSIDE\_DOWN // LCD module normal orientation

 LCD\_SEG\_DIR\_NORMAL**,** // segment direction: normal

 LCD\_COM\_DIR\_REV**,** // COM direction: reversed

#else // LCD module upside down orientation

 LCD\_SEG\_DIR\_REV**,** // segment direction: reversed

 LCD\_COM\_DIR\_NORMAL**,** // COM direction: normal

#endif

 LCD\_START\_LINE**,** // start line number: line 0

 LCD\_RR\_5\_5**,** // resistor regulation ratio: 5.5

 LCD\_EV**,** // electronic volume level (2B command!)

 LCD\_CONTRAST **&** LCD\_EV\_MSK**,** // EV / contrast

 LCD\_BOOSTER**,** // booster level (2B command!)

 BOOSTER\_LVL\_5X**,** // booster level 5x

 LCD\_PWR\_CONTROL**|**PWR\_VB**|**PWR\_VR**|**PWR\_VF**,**// power control: voltage booster ON,

 // voltage regulator ON,

 // voltage follower ON

 LCD\_DISPLAY\_NORMAL**,** // display mode: normal

 LCD\_DISPLAY\_ON // display ON

**};**

void lcdHEM6432\_Init**(**void**)**

**{**

 /// I2C and DWT configuration

 // Driver assumes that i2c peripheral and DWT unit are already configured

 // by peripheral drivers and running

 /// LCD\_RST pin configuration

 // - enable GPIO clock

 RCC**->**APB2ENR **|=** LCD\_RST\_GPIO\_CLK**;**

 // - clear GPIO CNF and MODE bit fields

 LCD\_RST\_GPIO**->**CRH **&=** **(~((**uint64\_t**)(**GPIO\_CRL\_CNF0 **|** GPIO\_CRL\_MODE0**)** **<<** 4**\***LCD\_RST\_PIN**)** **>>** 32**);**

 LCD\_RST\_GPIO**->**CRL **&=** **(~((**uint64\_t**)(**GPIO\_CRL\_CNF0 **|** GPIO\_CRL\_MODE0**)** **<<** 4**\***LCD\_RST\_PIN**)** **>>** 0**);**

 // - set LCD\_RST pin as 2MHz general purpose push-pull output (i.e. conf.=0x0,

 // mode=0x2), high level

 LCD\_RST\_GPIO**->**BSRR **=** **(**1 **<<** LCD\_RST\_PIN**);**

 LCD\_RST\_GPIO**->**CRH **|=** **(((**uint64\_t**)(**GPIO\_CRL\_MODE0\_1**)** **<<** 4**\***LCD\_RST\_PIN**)** **>>** 32**);**

 LCD\_RST\_GPIO**->**CRL **|=** **(((**uint64\_t**)(**GPIO\_CRL\_MODE0\_1**)** **<<** 4**\***LCD\_RST\_PIN**)** **>>** 0**);**

 /// LCD hardware reset

 // LCD reset ought to be performed within 30ms after power supply voltage

 // becomes stable. Otherwise the display may operate in abnormal way. LCD

 // reset LOW pulse should last at least 1us and LCD should be reset within

 // 2us after rising edge of the reset pulse, but recommended delay times

 // should be longer than 5us and 5us respectively.

 LCD\_RST\_GPIO**->**BRR **=** **(**1 **<<** LCD\_RST\_PIN**);** // set LCD\_RST line low

 dwt\_Delay**(**10**);** // wait for 10us

 LCD\_RST\_GPIO**->**BSRR **=** **(**1 **<<** LCD\_RST\_PIN**);** // set LCD\_RST line high

 dwt\_Delay**(**10**);** // wait for 10us

 /// LCD initialization

 // LCD controller initialization code is stored in lcdInitData[] table.

 // Initialization phase, counting from hardware reset to sending Power Control

 // command ought to be completed within 5ms!

 lcdHEM6432\_SendCmd**((**uint8\_t **\*)**lcdInitData**,** **sizeof(**lcdInitData**));**

**}**

**Listing 3. Funkcja ustawienia wskaźnika zapisu na wybranej pozycji w pamięci obrazu**

#define LCD\_X\_SIZE 64 // LCD horizontal size (i.e. 64 columns)

#define LCD\_Y\_SIZE 32 // LCD vertical size (i.e. 32 lines)

#define LCD\_PAGE\_SIZE 8 // LCD DDRAM page size (i.e. 8 lines)

#ifndef LCD\_UPSIDE\_DOWN // LCD module normal orientation

 #define LCD\_COL\_OFFSET 0x20 // LCD DDRAM column address offset

#else // LCD module upside down orientation

 #define LCD\_COL\_OFFSET 0x24 // LCD DDRAM column address offset

#endif

void lcdHEM6432\_GoToXY**(**uint8\_t Xpos**,** uint8\_t Ypos**)**

**{**

 uint8\_t lcdPosBuf**[**3**];** // cursor position buffer

 // Check x coordinate limit (i.e. 0 <= x <= 63)

 **if** **(**Xpos **>=** LCD\_X\_SIZE**)** Xpos **=** LCD\_X\_SIZE**-**1**;**

 // Convert Xpos to column number

 Xpos **+=** LCD\_COL\_OFFSET**;**

 // Store 'Set column MSN address' command code with MSN column number

 lcdPosBuf**[**0**]** **=** LCD\_COL\_ADDR\_MSN **|** **(**Xpos **>>** 4**);**

 // Store 'Set column LSN address' command code with LSN column number

 lcdPosBuf**[**1**]** **=** LCD\_COL\_ADDR\_LSN **|** **(**Xpos **&** 0x0F**);**

 // Check y coordinate limit (i.e. 0 <= y <= 3)

 **if** **(**Ypos **>=** LCD\_Y\_SIZE**/**LCD\_PAGE\_SIZE**)** Ypos **=** LCD\_Y\_SIZE**/**LCD\_PAGE\_SIZE**-**1**;**

 // Store 'Set page address' command code with page number

 lcdPosBuf**[**2**]** **=** LCD\_PAGE\_ADDR **|** Ypos**;**

 // Send command to LCD

 lcdHEM6432\_SendCmd**(**lcdPosBuf**,** 3**);**

**}**

**Listing 4. Funkcja wyświetlania pojedynczego znaku na aktualnie wybranej pozycji kursora**

void lcdHEM6432\_PutChar**(**uint8\_t char\_code**)**

**{**

 uint8\_t i**;** // loop counter

 uint8\_t lcdCharPatBuf**[**6**];** // character pattern buffer

 // Check character code and convert it to the printable one

 **if** **((**char\_code **<** ' '**)** **||** **(**char\_code **>** '~'**))** char\_code **=** ' '**;**

 // Convert character code to char patterns ROM table index

 char\_code **-=** ' '**;**

 // Copy character pattern to buffer

 **for** **(**i **=** 0**;** i **<** 5**;** i**++)**

 **{**

 lcdCharPatBuf**[**i**]** **=** LCDFontROM**[**char\_code**][**i**];**

 **}**

 // Add one pixel space between characters

 lcdCharPatBuf**[**5**]** **=** 0x00**;**

 // Send character pattern to LCD

 lcdHEM6432\_SendData**(**lcdCharPatBuf**,** 6**);**

**}**