/\*

u8g\_dev\_ld7032\_60x32.c

60x32 OLED display

Universal 8bit Graphics Library

Copyright (c) 2011, olikraus@gmail.com

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,

are permitted provided that the following conditions are met:

\* Redistributions of source code must retain the above copyright notice, this list

of conditions and the following disclaimer.

\* Redistributions in binary form must reproduce the above copyright notice, this

list of conditions and the following disclaimer in the documentation and/or other

materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND

CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,

INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR

CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,

STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF

ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

#include "u8g.h"

/\* define width as 64, so that it is a multiple of 8 \*/

#define WIDTH 64

#define HEIGHT 32

#define PAGE\_HEIGHT 8

static const uint8\_t u8g\_dev\_ld7032\_60x32\_init\_seq[] PROGMEM = {

U8G\_ESC\_CS(0), /\* disable chip \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

U8G\_ESC\_DLY(1), /\* delay 1 ms \*/

U8G\_ESC\_RST(1), /\* do reset low pulse with (1\*16)+2 milliseconds \*/

U8G\_ESC\_CS(1), /\* enable chip \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x002, /\* Dot Matrix Display ON/OFF \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x001, /\* ON \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x014, /\* Dot Matrix Display Stand-by ON/OFF \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* ON \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x01a, /\* Dot Matrix Frame Rate \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x004, /\* special value for this OLED from manual \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x01d, /\* Graphics Memory Writing Direction \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* reset default (right down, horizontal) \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x009, /\* Display Direction \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* reset default (x,y: min --> max) \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x030, /\* Display Size X \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* Column Start Output \*/

0x03b, /\* Column End Output \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x032, /\* Display Size Y \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* Row Start Output \*/

0x01f, /\* Row End Output \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x010, /\* Peak Pulse Width Set \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* 0 SCLK \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x016, /\* Peak Pulse Delay Set \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* 0 SCLK \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x012, /\* Dot Matrix Current Level Set \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x050, /\* 0x050 \* 1 uA = 80 uA \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x018, /\* Pre-Charge Pulse Width \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x003, /\* 3 SCLK \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x044, /\* Pre-Charge Mode \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x002, /\* Every Time \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x048, /\* Row overlap timing \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x003, /\* Pre-Charge + Peak Delay + Peak boot Timing \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x03f, /\* VCC\_R\_SEL \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x011, /\* ??? \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x03d, /\* VSS selection \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* 2.8V \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x002, /\* Dot Matrix Display ON/OFF \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x001, /\* ON \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x008, /\* write data \*/

U8G\_ESC\_CS(0), /\* disable chip \*/

U8G\_ESC\_END /\* end of sequence \*/

};

/\* use box commands to set start adr \*/

static const uint8\_t u8g\_dev\_ld7032\_60x32\_data\_start[] PROGMEM = {

U8G\_ESC\_ADR(0), /\* instruction mode \*/

U8G\_ESC\_CS(1), /\* enable chip \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x034, /\* box x start \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000, /\* 0 \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x035, /\* box x end \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x007, /\* \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x037, /\* box y end \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

0x01f, /\* \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

0x036, /\* box y start \*/

U8G\_ESC\_ADR(1), /\* data mode \*/

U8G\_ESC\_END /\* end of sequence \*/

};

static const uint8\_t u8g\_dev\_ld7032\_60x32\_sleep\_on[] PROGMEM = {

U8G\_ESC\_ADR(0), /\* instruction mode \*/

U8G\_ESC\_CS(1), /\* enable chip \*/

/\* ... \*/

U8G\_ESC\_CS(0), /\* disable chip \*/

U8G\_ESC\_END /\* end of sequence \*/

};

static const uint8\_t u8g\_dev\_ld7032\_60x32\_sleep\_off[] PROGMEM = {

U8G\_ESC\_ADR(0), /\* instruction mode \*/

U8G\_ESC\_CS(1), /\* enable chip \*/

/\* ... \*/

U8G\_ESC\_DLY(50), /\* delay 50 ms \*/

U8G\_ESC\_CS(0), /\* disable chip \*/

U8G\_ESC\_END /\* end of sequence \*/

};

uint8\_t u8g\_dev\_ld7032\_60x32\_fn(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t msg, void \*arg)

{

switch(msg)

{

case U8G\_DEV\_MSG\_INIT:

u8g\_InitCom(u8g, dev, U8G\_SPI\_CLK\_CYCLE\_400NS);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ld7032\_60x32\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

u8g\_pb\_t \*pb = (u8g\_pb\_t \*)(dev->dev\_mem);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ld7032\_60x32\_data\_start);

u8g\_WriteByte(u8g, dev, pb->p.page\_y0); /\* y start \*/

u8g\_SetAddress(u8g, dev, 0); /\* instruction mode \*/

u8g\_WriteByte(u8g, dev, 0x008);

u8g\_SetAddress(u8g, dev, 1); /\* data mode \*/

if ( u8g\_pb\_WriteBuffer(pb, u8g, dev) == 0 )

return 0;

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_CONTRAST:

u8g\_SetChipSelect(u8g, dev, 1);

u8g\_SetAddress(u8g, dev, 0); /\* instruction mode \*/

u8g\_WriteByte(u8g, dev, 0x081);

u8g\_WriteByte(u8g, dev, (\*(uint8\_t \*)arg) >> 2);

u8g\_SetChipSelect(u8g, dev, 0);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ld7032\_60x32\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ld7032\_60x32\_sleep\_off);

return 1;

}

return u8g\_dev\_pb8h1\_base\_fn(u8g, dev, msg, arg);

}

U8G\_PB\_DEV(u8g\_dev\_ld7032\_60x32\_sw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ld7032\_60x32\_fn, U8G\_COM\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ld7032\_60x32\_hw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ld7032\_60x32\_fn, U8G\_COM\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ld7032\_60x32\_parallel, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ld7032\_60x32\_fn, U8G\_COM\_PARALLEL);

U8G\_PB\_DEV(u8g\_dev\_ld7032\_60x32\_hw\_usart\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ld7032\_60x32\_fn, U8G\_COM\_HW\_USART\_SPI);