/\*

u8g\_state.c

backup and restore hardware state

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.

state callback: backup env U8G\_STATE\_MSG\_BACKUP\_ENV

device callback: DEV\_MSG\_INIT

state callback: backup u8g U8G\_STATE\_MSG\_BACKUP\_U8G

state callback: restore env U8G\_STATE\_MSG\_RESTORE\_ENV

state callback: backup env U8G\_STATE\_MSG\_BACKUP\_ENV

state callback: retore u8g U8G\_STATE\_MSG\_RESTORE\_U8G

DEV\_MSG\_PAGE\_FIRST or DEV\_MSG\_PAGE\_NEXT

state callback: restore env U8G\_STATE\_MSG\_RESTORE\_ENV

\*/

#include <stddef.h>

#include "u8g.h"

void u8g\_state\_dummy\_cb(uint8\_t msg)

{

/\* the dummy procedure does nothing \*/

}

void u8g\_SetHardwareBackup(u8g\_t \*u8g, u8g\_state\_cb backup\_cb)

{

u8g->state\_cb = backup\_cb;

/\* in most cases the init message was already sent, so this will backup the \*/

/\* current u8g state \*/

backup\_cb(U8G\_STATE\_MSG\_BACKUP\_U8G);

}

/\*===============================================================\*/

/\* register variable for restoring interrupt state \*/

#if defined(\_\_AVR\_\_)

uint8\_t global\_SREG\_backup;

#endif

/\*===============================================================\*/

/\* AVR \*/

#if defined(\_\_AVR\_\_)

#define U8G\_ATMEGA\_HW\_SPI

/\* remove the definition for attiny \*/

#if \_\_AVR\_ARCH\_\_ == 2

#undef U8G\_ATMEGA\_HW\_SPI

#endif

#if \_\_AVR\_ARCH\_\_ == 25

#undef U8G\_ATMEGA\_HW\_SPI

#endif

#endif

#if defined(U8G\_ATMEGA\_HW\_SPI)

#include <avr/interrupt.h>

static uint8\_t u8g\_state\_avr\_spi\_memory[2];

void u8g\_backup\_spi(uint8\_t msg)

{

if ( U8G\_STATE\_MSG\_IS\_BACKUP(msg) )

{

u8g\_state\_avr\_spi\_memory[U8G\_STATE\_MSG\_GET\_IDX(msg)] = SPCR;

}

else

{

uint8\_t tmp = SREG;

cli();

SPCR = 0;

SPCR = u8g\_state\_avr\_spi\_memory[U8G\_STATE\_MSG\_GET\_IDX(msg)];

SREG = tmp;

}

}

#elif defined (U8G\_RASPBERRY\_PI)

#include <stdio.h>

void u8g\_backup\_spi(uint8\_t msg) {

printf("u8g\_backup\_spi %d\r\n",msg);

}

#elif defined(ARDUINO) && defined(\_\_SAM3X8E\_\_) // Arduino Due, maybe we should better check for \_\_SAM3X8E\_\_

#include "sam.h"

struct sam\_backup\_struct

{

uint32\_t mr;

uint32\_t sr;

uint32\_t csr[4];

} sam\_backup[2];

void u8g\_backup\_spi(uint8\_t msg)

{

uint8\_t idx = U8G\_STATE\_MSG\_GET\_IDX(msg);

if ( U8G\_STATE\_MSG\_IS\_BACKUP(msg) )

{

sam\_backup[idx].mr = SPI0->SPI\_MR;

sam\_backup[idx].sr = SPI0->SPI\_SR;

sam\_backup[idx].csr[0] = SPI0->SPI\_CSR[0];

sam\_backup[idx].csr[1] = SPI0->SPI\_CSR[1];

sam\_backup[idx].csr[2] = SPI0->SPI\_CSR[2];

sam\_backup[idx].csr[3] = SPI0->SPI\_CSR[3];

}

else

{

SPI0->SPI\_MR = sam\_backup[idx].mr;

SPI0->SPI\_CSR[0] = sam\_backup[idx].csr[0];

SPI0->SPI\_CSR[1] = sam\_backup[idx].csr[1];

SPI0->SPI\_CSR[2] = sam\_backup[idx].csr[2];

SPI0->SPI\_CSR[3] = sam\_backup[idx].csr[3];

}

}

#else

void u8g\_backup\_spi(uint8\_t msg)

{

}

#endif