/\*

Special pin usage:

U8G\_PI\_I2C\_OPTION additional options

U8G\_PI\_A0\_STATE used to store the last value of the command/data register selection

U8G\_PI\_SET\_A0 1: Signal request to update I2C device with new A0\_STATE, 0: Do nothing, A0\_STATE matches I2C device

U8G\_PI\_SCL clock line (NOT USED)

U8G\_PI\_SDA data line (NOT USED)

U8G\_PI\_RESET reset line (currently disabled, see below)

Protocol:

SLA, Cmd/Data Selection, Arguments

The command/data register is selected by a special instruction byte, which is sent after SLA

The continue bit is always 0 so that a (re)start is equired for the change from cmd to/data mode

\*/

#include "u8g.h"

#if defined(U8G\_RASPBERRY\_PI)

#include <wiringPi.h>

#include <wiringPiI2C.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#define I2C\_SLA 0x3c

#define I2C\_CMD\_MODE 0x000

#define I2C\_DATA\_MODE 0x040

#if defined(U8G\_WITH\_PINLIST)

uint8\_t u8g\_com\_raspberrypi\_ssd\_start\_sequence(u8g\_t \*u8g)

{

/\* are we requested to set the a0 state? \*/

if ( u8g->pin\_list[U8G\_PI\_SET\_A0] == 0 )

return 1;

/\* setup bus, might be a repeated start \*/

if ( u8g\_i2c\_start(I2C\_SLA) == 0 )

return 0;

if ( u8g->pin\_list[U8G\_PI\_A0\_STATE] == 0 )

{

if ( u8g\_i2c\_send\_mode(I2C\_CMD\_MODE) == 0 )

return 0;

}

else

{

if ( u8g\_i2c\_send\_mode(I2C\_DATA\_MODE) == 0 )

return 0;

}

u8g->pin\_list[U8G\_PI\_SET\_A0] = 0;

return 1;

}

uint8\_t u8g\_com\_raspberrypi\_ssd\_i2c\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

switch(msg)

{

case U8G\_COM\_MSG\_INIT:

u8g\_i2c\_init(u8g->pin\_list[U8G\_PI\_I2C\_OPTION]);

u8g\_SetPIOutput(u8g, U8G\_PI\_RESET);

u8g\_SetPIOutput(u8g, U8G\_PI\_A0);

break;

case U8G\_COM\_MSG\_STOP:

break;

case U8G\_COM\_MSG\_RESET:

break;

case U8G\_COM\_MSG\_CHIP\_SELECT:

u8g->pin\_list[U8G\_PI\_A0\_STATE] = 0;

u8g->pin\_list[U8G\_PI\_SET\_A0] = 1; /\* force a0 to set again, also forces start condition \*/

if ( arg\_val == 0 )

{

/\* disable chip, send stop condition \*/

u8g\_i2c\_stop();

}

else

{

/\* enable, do nothing: any byte writing will trigger the i2c start \*/

}

break;

case U8G\_COM\_MSG\_WRITE\_BYTE:

//u8g->pin\_list[U8G\_PI\_SET\_A0] = 1;

if ( u8g\_com\_raspberrypi\_ssd\_start\_sequence(u8g) == 0 )

return u8g\_i2c\_stop(), 0;

if ( u8g\_i2c\_send\_byte(arg\_val) == 0 )

return u8g\_i2c\_stop(), 0;

// u8g\_i2c\_stop();

break;

case U8G\_COM\_MSG\_WRITE\_SEQ:

//u8g->pin\_list[U8G\_PI\_SET\_A0] = 1;

if ( u8g\_com\_raspberrypi\_ssd\_start\_sequence(u8g) == 0 )

return u8g\_i2c\_stop(), 0;

{

register uint8\_t \*ptr = (uint8\_t \*)arg\_ptr;

while( arg\_val > 0 )

{

if ( u8g\_i2c\_send\_byte(\*ptr++) == 0 )

return u8g\_i2c\_stop(), 0;

arg\_val--;

}

}

// u8g\_i2c\_stop();

break;

case U8G\_COM\_MSG\_WRITE\_SEQ\_P:

//u8g->pin\_list[U8G\_PI\_SET\_A0] = 1;

if ( u8g\_com\_raspberrypi\_ssd\_start\_sequence(u8g) == 0 )

return u8g\_i2c\_stop(), 0;

{

register uint8\_t \*ptr = (uint8\_t \*)arg\_ptr;

while( arg\_val > 0 )

{

if ( u8g\_i2c\_send\_byte(u8g\_pgm\_read(ptr)) == 0 )

return 0;

ptr++;

arg\_val--;

}

}

// u8g\_i2c\_stop();

break;

case U8G\_COM\_MSG\_ADDRESS: /\* define cmd (arg\_val = 0) or data mode (arg\_val = 1) \*/

u8g->pin\_list[U8G\_PI\_A0\_STATE] = arg\_val;

u8g->pin\_list[U8G\_PI\_SET\_A0] = 1; /\* force a0 to set again \*/

#ifdef OLD\_CODE

if ( i2c\_state != 0 )

{

u8g\_i2c\_stop();

i2c\_state = 0;

}

if ( u8g\_com\_raspberrypi\_ssd\_start\_sequence(arg\_val) == 0 )

return 0;

/\* setup bus, might be a repeated start \*/

/\*

if ( u8g\_i2c\_start(I2C\_SLA) == 0 )

return 0;

if ( arg\_val == 0 )

{

i2c\_state = 1;

if ( u8g\_i2c\_send\_byte(I2C\_CMD\_MODE) == 0 )

return 0;

}

else

{

i2c\_state = 2;

if ( u8g\_i2c\_send\_byte(I2C\_DATA\_MODE) == 0 )

return 0;

}

\*/

#endif

break;

}

return 1;

}

#else /\* defined(U8G\_WITH\_PINLIST) \*/

uint8\_t u8g\_com\_raspberrypi\_ssd\_i2c\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr) {

return 1;

}

#endif /\* defined(U8G\_WITH\_PINLIST) \*/

#endif