libpynq

(release 5EWC0-2023 version 0.2.6 of 2023-10-24 17:38)

Generated on Tue Oct 24 2023 17:38:12 for libpynq by Doxygen 1.9.8

Tue Oct 24 2023 17:38:12

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Topic Documentation

4.1 ADC library

Enumerations

- enum adc_channel_t {
  ADC0 = ((0x240 / 4) + 1) , ADC1 = ((0x240 / 4) + 9) , ADC2 = ((0x240 / 4) + 6) , ADC3 = ((0x240 / 4) + 15) ,
  ADC4 = ((0x240 / 4) + 5) , ADC5 = ((0x240 / 4) + 13) }

Functions

- bool initialized_adc (void)
- void adc_init (void)
- void adc_destroy (void)
- double adc_read_channel (adc_channel_t channel)
- uint32_t adc_read_channel_raw (adc_channel_t channel)

4.1.1 Detailed Description

Functions to use the Analog to Digital Conversion (ADC) of analog pins (A0..A5 on the PYNQ board).

Note that GPIO numbering (IO_A0..IO_A5) used in gpio.h and pinmap.h is different from A0..A5.

4.1.2 Enumeration Type Documentation

4.1.2.1 adc_channel_t

```
enum adc_channel_t
```

Enumerate the different available ADC channels.

Enumerator

ADC0	ADC channel for pin IO_A0
ADC1	ADC channel for pin IO_A1
ADC2	ADC channel for pin IO_A2
ADC3	ADC channel for pin IO_A3
ADC4	ADC channel for pin IO_A4
ADC5	ADC channel for pin IO_A5

Definition at line 43 of file adc.h.

4.1.3 Function Documentation

4.1.3.1 adc_destroy()

```
void adc_destroy (
            void  ) [extern]
```

De-initialize the ADC library and free up the used memory in the shared memory space.

Definition at line 80 of file adc.c.

Here is the call graph for this function:

4.1.3.2 adc_init()

```
void adc_init (
            void  ) [extern]
```

Initialization of the ADC library.

Definition at line 78 of file adc.c.

Here is the call graph for this function:

4.1.3.3 adc_read_channel()

```
double adc_read_channel (
            adc_channel_t channel ) [extern]
```

Parameters

channel	The channel to read the analog value from. Read ADC channel #channel and return the read out voltage.

Returns

> a value between 0.0 and 3.3V.

Warning

> Fails with program exit when channel is outside valid range or has not been initialized..

Definition at line 87 of file adc.c.

Here is the call graph for this function:

4.1.3.4 adc_read_channel_raw()

```
uint32_t adc_read_channel_raw (
            adc_channel_t channel ) [extern]
```

```
adc_channel_t channel ) [extern]
```

Parameters

channel	The channel to read the analog value from. Read ADC channel #channel and return the raw value.

Returns

a value between 0 and 65535.

Warning

Fails with program exit when channel is outside valid range.

Definition at line 97 of file adc.c.

Here is the call graph for this function:

4.1.3.5 initialized_adc()

```
bool initialized_adc (
            void  ) [extern]
```

Check if ADC has been initialized.

Returns

True when initialized, false otherwise.

Definition at line 57 of file adc.c.

Here is the caller graph for this function:

4.2 ARM MMIO library

Data Structures

- struct arm_shared_t

Typedefs

- typedef struct arm_shared_t arm_shared

Functions

- void ∗ arm_shared_init (arm_shared ∗handle, const uint32_t address, const uint32_t length)
- void arm_shared_close (arm_shared ∗handle)

4.2.1 Detailed Description

Do not use. Low-level functions for MMIO access to the FPGA fabric.

This library gives low-level memory-mapped access to the hardware units in the FPGA.

This is an internal library and should not be directly used.

4.2.2 Typedef Documentation

4.2.2.1 arm_shared

```
typedef struct arm_shared_t arm_shared
```

Object handle.

Definition at line 48 of file arm_shared_memory_system.h.

4.2.3 Function Documentation

4.2.3.1 arm_shared_close()

```
void arm_shared_close (
            arm_shared * handle ) [extern]
```

Parameters

handle	a handle to its internal state.

closes the shared memory region, invalidating the previously accessed pointer.

Definition at line 70 of file arm_shared_memory_system.c.

Here is the caller graph for this function:

4.2.3.2 arm_shared_init()

```
void * arm_shared_init (
            arm_shared * handle,
            const uint32_t address,
            const uint32_t length ) [extern]
```

Parameters

handle	a handle to store it internal state.
address	address to access (should be in the shared memory range).
length	the length of the section to access.

Open a shared memory for reading and writing.

Returns

a pointer to the shared memory region.

Definition at line 32 of file arm_shared_memory_system.c.

Here is the caller graph for this function:

4.3 Audio library

Macros

- #define LINE_IN 0
- #define MIC 1
- #define IIC_SLAVE_ADDR 0x3B
- #define IIC_SCLK_RATE 400000
- #define I2S_DATA_RX_L_REG 0x00
- #define I2S_DATA_RX_R_REG 0x04
- #define I2S_DATA_TX_L_REG 0x08
- #define I2S_DATA_TX_R_REG 0x0C
- #define I2S_STATUS_REG 0x10

Enumerations

- enum audio_adau1761_regs {
  R0_CLOCK_CONTROL = 0x00 , R1_PLL_CONTROL = 0x02 , R2_DIGITAL_MIC_JACK_DETECTION_CONTROL
  = 0x08 , R3_RECORD_POWER_MANAGEMENT = 0x09 ,
  R4_RECORD_MIXER_LEFT_CONTROL_0 = 0x0A , R5_RECORD_MIXER_LEFT_CONTROL_1 = 0x0B ,
  R6_RECORD_MIXER_RIGHT_CONTROL_0 = 0x0C , R7_RECORD_MIXER_RIGHT_CONTROL_1 = 0x0D
  ,
  R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL = 0x0E , R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL
  = 0x0F , R10_RECORD_MICROPHONE_BIAS_CONTROL = 0x10 , R11_ALC_CONTROL_0 = 0x11 ,
  R12_ALC_CONTROL_1 = 0x12 , R13_ALC_CONTROL_2 = 0x13 , R14_ALC_CONTROL_3 = 0x14 ,
  R15_SERIAL_PORT_CONTROL_0 = 0x15 ,
  R16_SERIAL_PORT_CONTROL_1 = 0x16 , R17_CONVERTER_CONTROL_0 = 0x17 , R18_CONVERTER_CONTROL_1
  = 0x18 , R19_ADC_CONTROL = 0x19 ,
  R20_LEFT_INPUT_DIGITAL_VOLUME = 0x1A , R21_RIGHT_INPUT_DIGITAL_VOLUME = 0x1B ,
  R22_PLAYBACK_MIXER_LEFT_CONTROL_0 = 0x1C , R23_PLAYBACK_MIXER_LEFT_CONTROL_1
  = 0x1D ,
  R24_PLAYBACK_MIXER_RIGHT_CONTROL_0 = 0x1E , R25_PLAYBACK_MIXER_RIGHT_CONTROL_1 =
  0x1F , R26_PLAYBACK_LR_MIXER_LEFT_LINE_OUTPUT_CONTROL = 0x20 , R27_PLAYBACK_LR_MIXER_RIGHT_LINE_OUTPUT_CONTROL
  = 0x21 ,
  R28_PLAYBACK_LR_MIXER_MONO_OUTPUT_CONTROL = 0x22 , R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL
  = 0x23 , R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL = 0x24 , R31_PLAYBACK_LINE_OUTPUT_LEFT_VOLUME_CONTROL
  = 0x25 ,
  R32_PLAYBACK_LINE_OUTPUT_RIGHT_VOLUME_CONTROL = 0x26 , R33_PLAYBACK_MONO_OUTPUT_CONTROL
  = 0x27 , R34_PLAYBACK_POP_CLICK_SUPPRESSION = 0x28 , R35_PLAYBACK_POWER_MANAGEMENT
  = 0x29 ,
  R36_DAC_CONTROL_0 = 0x2A , R37_DAC_CONTROL_1 = 0x2B , R38_DAC_CONTROL_2 = 0x2C ,
  R39_SERIAL_PORT_PAD_CONTROL = 0x2D ,
  R40_CONTROL_PORT_PAD_CONTROL_0 = 0x2F , R41_CONTROL_PORT_PAD_CONTROL_1 = 0x30 ,
  R42_JACK_DETECT_PIN_CONTROL = 0x31 , R67_DEJITTER_CONTROL = 0x36 ,
  R58_SERIAL_INPUT_ROUTE_CONTROL = 0xF2 , R59_SERIAL_OUTPUT_ROUTE_CONTROL = 0xF3 ,
  R61_DSP_ENABLE = 0xF5 , R62_DSP_RUN = 0xF6 ,
  R63_DSP_SLEW_MODES = 0xF7 , R64_SERIAL_PORT_SAMPLING_RATE = 0xF8 , R65_CLOCK_ENABLE_0
  = 0xF9 , R66_CLOCK_ENABLE_1 = 0xFA }

Functions

- void audio_init (void)
- void audio_select_input (int input)
- void write_audio_reg (unsigned char u8RegAddr, unsigned char u8Data, int iic_fd)
- void config_audio_pll (void)
- void config_audio_codec (void)
- void select_line_in (void)
- void select_mic (void)
- void deselect (void)
- void audio_bypass (unsigned int audio_mmap_size, unsigned int nsamples, unsigned int volume, int uio_↵ index)
- void audio_record (unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, int uio_↵ index)
- void audio_play (unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, unsigned int volume, int uio_index)
- void audio_repeat_play (unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, unsigned int volume, unsigned int repetitions)
- void audio_generate_tone (unsigned int frequency, uint32_t time_ms, unsigned int volume)

4.3.1 Detailed Description

Low-level audio functions.

mic+ph and line_in can be used as audio input and mic+ph as output.

An example of using this library to play audio from line_in to mic+Ph:

```
#include <libpynq.h>
int main (void)
{
 pynq_init();
 audio_init();
 audio_select_input(MIC);
 while(1) {
   audio_bypass(64*1024, 32*1024, 50, 0);
 }
 deselect();
 pynq_destroy();
 return EXIT_SUCCES;
}
```

4.3.2 Macro Definition Documentation

4.3.2.1 I2S_DATA_RX_L_REG

```
#define I2S_DATA_RX_L_REG 0x00
```

Definition at line 42 of file audio.h.

4.3.2.2 I2S_DATA_RX_R_REG

```
#define I2S_DATA_RX_R_REG 0x04
```

Definition at line 43 of file audio.h.

4.3.2.3 I2S_DATA_TX_L_REG

`#define I2S_DATA_TX_L_REG 0x08`

Definition at line 44 of file audio.h.

4.3.2.4 I2S_DATA_TX_R_REG

`#define I2S_DATA_TX_R_REG 0x0C`

Definition at line 45 of file audio.h.

4.3.2.5 I2S_STATUS_REG

`#define I2S_STATUS_REG 0x10`

Definition at line 46 of file audio.h.

4.3.2.6 IIC_SCLK_RATE

`#define IIC_SCLK_RATE 400000`

Definition at line 39 of file audio.h.

4.3.2.7 IIC_SLAVE_ADDR

`#define IIC_SLAVE_ADDR 0x3B`

Definition at line 36 of file audio.h.

4.3.2.8 LINE_IN

`#define LINE_IN 0`

Definition at line 32 of file audio.h.

4.3.2.9 MIC

`#define MIC 1`

Definition at line 33 of file audio.h.

4.3.3 Enumeration Type Documentation

4.3.3.1 audio_adau1761_regs

`enum audio_adau1761_regs`

Enumerator

R0_CLOCK_CONTROL	
R1_PLL_CONTROL	
R2_DIGITAL_MIC_JACK_DETECTION_CONTROL	
R3_RECORD_POWER_MANAGEMENT	
R4_RECORD_MIXER_LEFT_CONTROL_0	
R5_RECORD_MIXER_LEFT_CONTROL_1	
R6_RECORD_MIXER_RIGHT_CONTROL_0	
R7_RECORD_MIXER_RIGHT_CONTROL_1	
R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL	
R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL	
R10_RECORD_MICROPHONE_BIAS_CONTROL	
R11_ALC_CONTROL_0	
R12_ALC_CONTROL_1	
R13_ALC_CONTROL_2	
R14_ALC_CONTROL_3	
R15_SERIAL_PORT_CONTROL_0	
R16_SERIAL_PORT_CONTROL_1	
R17_CONVERTER_CONTROL_0	
R18_CONVERTER_CONTROL_1	
R19_ADC_CONTROL	
R20_LEFT_INPUT_DIGITAL_VOLUME	
R21_RIGHT_INPUT_DIGITAL_VOLUME	
R22_PLAYBACK_MIXER_LEFT_CONTROL_0	
R23_PLAYBACK_MIXER_LEFT_CONTROL_1	
R24_PLAYBACK_MIXER_RIGHT_CONTROL_0	
R25_PLAYBACK_MIXER_RIGHT_CONTROL_1	
R26_PLAYBACK_LR_MIXER_LEFT_LINE_OUTPUT_CONTROL	
R27_PLAYBACK_LR_MIXER_RIGHT_LINE_OUTPUT_CONTROL	
R28_PLAYBACK_LR_MIXER_MONO_OUTPUT_CONTROL	
R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL	
R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL	
R31_PLAYBACK_LINE_OUTPUT_LEFT_VOLUME_CONTROL	
R32_PLAYBACK_LINE_OUTPUT_RIGHT_VOLUME_CONTROL	
R33_PLAYBACK_MONO_OUTPUT_CONTROL	
R34_PLAYBACK_POP_CLICK_SUPPRESSION	
R35_PLAYBACK_POWER_MANAGEMENT	
R36_DAC_CONTROL_0	
R37_DAC_CONTROL_1	
R38_DAC_CONTROL_2	
R39_SERIAL_PORT_PAD_CONTROL	
R40_CONTROL_PORT_PAD_CONTROL_0	
R41_CONTROL_PORT_PAD_CONTROL_1	
R42_JACK_DETECT_PIN_CONTROL	
R67_DEJITTER_CONTROL	
R58_SERIAL_INPUT_ROUTE_CONTROL	
R59_SERIAL_OUTPUT_ROUTE_CONTROL	
R61_DSP_ENABLE	
R62_DSP_RUN	
R63_DSP_SLEW_MODES	

Enumerator

R64_SERIAL_PORT_SAMPLING_RATE	
R65_CLOCK_ENABLE_0	
R66_CLOCK_ENABLE_1	

Definition at line 49 of file audio.h.

4.3.4 Function Documentation

4.3.4.1 audio_bypass()

```
void audio_bypass (
            unsigned int audio_mmap_size,
            unsigned int nsamples,
            unsigned int volume,
            int uio_index )  [extern]
```

Record and play the audio without storing in DRAM.

Parameters

audio_mmap_size	is the address range of the audio codec.
nsamples	is the number of samples to read and output.
uio_index	is the uio index in /dev list.

Definition at line 304 of file audio.c.

Here is the call graph for this function:

4.3.4.2 audio_generate_tone()

```
void audio_generate_tone (
            unsigned int frequency,
            uint32_t time_ms,
            unsigned int volume )  [extern]
```

Definition at line 570 of file audio.c.

Here is the call graph for this function:

4.3.4.3 audio_init()

```
void audio_init (
            void  )  [extern]
```

Initializes the audio register. Sets the sampling frequency. defines several values such as audio record volume and playback volume. output is always played over mic+ph aux output.

Definition at line 72 of file audio.c.

Here is the call graph for this function:

4.3.4.4 audio_play()

```
void audio_play (
            unsigned int audio_mmap_size,
            unsigned int * BufAddr,
            unsigned int nsamples,
            unsigned int volume,
            int uio_index )  [extern]
```

Definition at line 430 of file audio.c.

Here is the call graph for this function:

4.3.4.5 audio_record()

```
void audio_record (
            unsigned int audio_mmap_size,
            unsigned int * BufAddr,
            unsigned int nsamples,
            int uio_index )  [extern]
```

Function to support audio recording without the audio codec controller.

Notice that the buffer has to be twice the size of the number of samples, because both left and right channels are sampled.

Parameters

audio_mmap_size	is the address range of the audio codec.
BufAddr	is the buffer address.
nsamples	is the number of samples.
uio_index	is the uio index in /dev list.

Definition at line 381 of file audio.c.

Here is the call graph for this function:

4.3.4.6 audio_repeat_play()

```
void audio_repeat_play (
            unsigned int audio_mmap_size,
            unsigned int * BufAddr,
            unsigned int nsamples,
            unsigned int volume,
            unsigned int repetitions )  [extern]
```

Function to play one audio fragment for multiple repititions.

Parameters

audio_mmap_size	is the address range of the audio codec.
BufAddr	is the buffer address.
nsamples	is the number of samples.
volume	is the volume of the output.
repetitions	is the number of repitions.

Definition at line 502 of file audio.c.

Here is the call graph for this function:

4.3.4.7 audio_select_input()

```
void audio_select_input (
            int input ) [extern]
```

selects the audio input channel.

Parameters

input	defines the input. Can be 0 LINE_IN or 1 MIC

Warning

> Fails with program exit when input is not valid.

Definition at line 77 of file audio.c.

Here is the call graph for this function:

4.3.4.8 config_audio_codec()

```
void config_audio_codec (
            void ) [extern]
```

Definition at line 174 of file audio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.3.4.9 config_audio_pll()

```
void config_audio_pll (
            void ) [extern]
```

Definition at line 102 of file audio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.3.4.10 deselect()

```
void deselect (
            void ) [extern]
```

Function to deselect input, either LINE_IN, or MIC.

Definition at line 286 of file audio.c.

Here is the call graph for this function:

4.3.4.11 select_line_in()

```
void select_line_in (
            void  ) [extern]
```

Function to select LINE_IN as input.

Definition at line 234 of file audio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.3.4.12 select_mic()

```
void select_mic (
            void  ) [extern]
```

Function to select MIC as input.

Definition at line 257 of file audio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.3.4.13 write_audio_reg()

```
void write_audio_reg (
            unsigned char u8RegAddr,
            unsigned char u8Data,
            int iic_fd ) [extern]
```

Definition at line 90 of file audio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.4 Button library

Macros

- #define BUTTON_NOT_PUSHED 0
- #define BUTTON_PUSHED 1
- #define SWITCH_OFF 0
- #define SWITCH_ON 1

Enumerations

- enum button_index_t {
  BUTTON0 , BUTTON1 , BUTTON2 , BUTTON3 ,
  NUM_BUTTONS }
- enum switches_index_t { SWITCH0 , SWITCH1 , NUM_SWITCHES }

Functions

- void switches_init (void)
- void switches_destroy (void)
- void buttons_init (void)
- void buttons_destroy (void)
- int get_button_state (const int button)
- int wait_until_button_state (const int button, const int state)
- int sleep_msec_button_pushed (const int button, const int msec)
- void sleep_msec_buttons_pushed (int button_states[ ], const int ms)
- int wait_until_button_pushed (const int button)
- int wait_until_button_released (const int button)
- int wait_until_any_button_pushed (void)
- int wait_until_any_button_released (void)
- int get_switch_state (const int switch_num)

4.4.1 Detailed Description

Wrappers to simplify the use of buttons.

- Buttons are numbered 0..NUM_BUTTONS-1, and return values are BUTTON_PUSHED and BUTTON_↩ NOT_PUSHED

- Switches are numbered 0..NUM_SWITCHES-1, and return values are SWITCH_ON and SWITCH_OFF.

- wait_ functions return early, i.e. as soon as the stated condition is true.

- sleep_ functions do not return early, i.e. always wait until the specified number of milliseconds.

An example of how to use this library.
```
#include <libpynq.h>
int main (void)
{
  // initialise all I/O
  pynq_init();
  buttons_init();

  printf("Waiting until button 0 is pushed...\n");
  printf("Waited %d milliseconds\n\n", wait_until_button_pushed(0));
  printf("Waiting until button 0 is released...\n");
  printf("Waited %d milliseconds\n\n", wait_until_button_released(0));

  // clean up after use
  buttons_destroy();
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

Buttons can also be used through GPIO (see gpio.h and pinmap.h). Note that GPIO numbering (IO_BTN0..IO↩ _BTN3) is then used instead of 0..NUM_BUTTONS-1 (BUTTON0..BUTTON3). GPIO return values are GPIO_↩ LEVEL_LOW/HIGH instead of BUTTON_(NOT_)PUSHED.

Switches can also be used through GPIO (see gpio.h and pinmap.h). Note that GPIO numbering (IO_SW0..IO↩ _SW1) is then used instead of 0..NUM_SWITCHES-1 (SWITCH0..SWITCH1). GPIO return values are GPIO_↩ LEVEL_LOW/HIGH instead of SWITCH_ON/OFF.

4.4.2 Macro Definition Documentation

4.4.2.1 BUTTON_NOT_PUSHED

```
#define BUTTON_NOT_PUSHED 0
```

Definition at line 74 of file buttons.h.

4.4.2.2 BUTTON_PUSHED

```
#define BUTTON_PUSHED 1
```

Definition at line 75 of file buttons.h.

4.4.2.3 SWITCH_OFF

```
#define SWITCH_OFF 0
```

Definition at line 76 of file buttons.h.

4.4.2.4 SWITCH_ON

```
#define SWITCH_ON 1
```

Definition at line 77 of file buttons.h.

4.4.3 Enumeration Type Documentation

4.4.3.1 button_index_t

```
enum button_index_t
```

Enum of buttons.

Functions use a button numbered from 0..NUM_BUTTONS-1. Alternatively, you can use BUTTONi instead of just i if you find that clearer.

Enumerator

BUTTON0	
BUTTON1	
BUTTON2	
BUTTON3	
NUM_BUTTONS	

Definition at line 86 of file buttons.h.

4.4.3.2 switches_index_t

```
enum switches_index_t
```

Enum of switches. Functions use a switch numbered from 0..NUM_SWITCHES-1. Alternatively, you can use SWITCHi instead of just i if you find that clearer.

Enumerator

SWITCH0	
SWITCH1	
NUM_SWITCHES	

Definition at line 94 of file buttons.h.

4.4.4 Function Documentation

4.4.4.1 buttons_destroy()

```
void buttons_destroy (
            void  ) [extern]
```

Unitialize the buttons.

Definition at line 50 of file buttons.c.

4.4.4.2 buttons_init()

```
void buttons_init (
            void  ) [extern]
```

Initialise the buttons before they can be used.

Definition at line 39 of file buttons.c.

Here is the call graph for this function:

4.4.4.3 get_button_state()

```
int get_button_state (
            const int button ) [extern]
```

Return the state of the button (BUTTON_(NOT_)PUSHED).

Parameters

button	The button the state of which is returned.

Warning

> Fails with program exit when button is outside valid range.
>
> Fails with program exit when the direction of the button was not set to input (e.g. because buttons_init was not called before).

Definition at line 71 of file buttons.c.

Here is the call graph for this function:

4.4.4.4 get_switch_state()

```
int get_switch_state (
            const int switch_num ) [extern]
```

Returns

> The state of the switch number (1 for on, 0 for off).

Warning

> Fails with program exit when switch is outside valid range.
>
> Fails with program exit when the direction of any switch was not set to input (e.g. because buttons_init was not called before).

Definition at line 217 of file buttons.c.

Here is the call graph for this function:

4.4.4.5 sleep_msec_button_pushed()

```
int sleep_msec_button_pushed (
            const int button,
            const int msec ) [extern]
```

Check if the given button is pushed in msec milliseconds. The function does NOT return early.

Parameters

button	The button of which the state is monitored.
msec	The number of milliseconds to wait.

Returns

> BUTTON_PUSHED or BUTTON_NOT_PUSHED.

Warning

> Fails with program exit when button is outside valid range.
>
> Fails with program exit when the direction of the button was not set to input (e.g. because buttons_init was not called before).

Definition at line 109 of file buttons.c.

Here is the call graph for this function:

4.4.4.6 sleep_msec_buttons_pushed()

```
void sleep_msec_buttons_pushed (
            int button_states[],
            const int ms ) [extern]
```

Check if any button is pushed in msec milliseconds. The function does NOT return early.

Parameters

button_states	The array of button states that are updated with BUTTON_PUSHED or BUTTON_NOT_PUSHED.

Warning

    Fails with program exit when the direction of any button was not set to input (e.g. because buttons_init was not called before).

Definition at line 140 of file buttons.c.

Here is the call graph for this function:

4.4.4.7 switches_destroy()

```
void switches_destroy (
            void  ) [extern]
```

Unitialize the buttons.

Definition at line 65 of file buttons.c.

4.4.4.8 switches_init()

```
void switches_init (
            void  ) [extern]
```

Initialise the switches before they can be used.

Definition at line 56 of file buttons.c.

Here is the call graph for this function:

4.4.4.9 wait_until_any_button_pushed()

```
int wait_until_any_button_pushed (
            void  ) [extern]
```

Wait until any button is not pushed (which may be immediately).

Returns

    Wait until any button is pushed, return the number of the button that was pushed (0..NUM_BUTTONS-1).

Warning

    Fails with program exit when the direction of any button was not set to input (e.g. because buttons_init was not called before).

Definition at line 176 of file buttons.c.

Here is the call graph for this function:

4.4.4.10 wait_until_any_button_released()

```
int wait_until_any_button_released (
            void ) [extern]
```

Wait until the given button is not pushed (which may be immediately).

Returns

Wait until any button is released, return the number of the button that was pushed (0..NUM_BUTTONS-1).

Warning

Fails with program exit when the direction of any button was not set to input (e.g. because buttons_init was not called before).

Definition at line 197 of file buttons.c.

Here is the call graph for this function:

4.4.4.11 wait_until_button_pushed()

```
int wait_until_button_pushed (
            const int button ) [extern]
```

Wait until the given button is pushed (which may be immediately).

Parameters

button	The button of which the state is monitored.

Returns

The number of milliseconds waited until the button was pushed.

Warning

Fails with program exit when button is outside valid range.

Fails with program exit when the direction of the button was not set to input (e.g. because buttons_init was not called before).

Definition at line 166 of file buttons.c.

Here is the call graph for this function:

4.4.4.12 wait_until_button_released()

```
int wait_until_button_released (
            const int button ) [extern]
```

Wait until the given button is not pushed (which may be immediately).

Parameters

button	The button of which the state is monitored.

Returns

> The number of milliseconds waited until the button was released.

Warning

> Fails with program exit when button is outside valid range.
>
> Fails with program exit when the direction of the button was not set to input (e.g. because buttons_init was not called before).

Definition at line 171 of file buttons.c.

Here is the call graph for this function:

4.4.4.13 wait_until_button_state()

```
int wait_until_button_state (
            const int button,
            const int state )  [extern]
```

Wait until the given button is in state (which may be immediately).

Parameters

button	The button of which the state is monitored.
state	The state that is waited for. Must be BUTTON_PUSHED or BUTTON_NOT_PUSHED.

Returns

> The number of milliseconds that was waited.

Warning

> Fails with program exit when button is outside valid range.
>
> Fails with program exit when the direction of the button was not set to input (e.g. because buttons_init was not called before).

Definition at line 83 of file buttons.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.5 Display library

Data Structures

- struct display_t

Macros

- #define DISPLAY_HEIGHT 240
- #define DISPLAY_WIDTH 240

Enumerations

- enum colors {
  RGB_RED = 0xf800 , RGB_GREEN = 0x07e0 , RGB_BLUE = 0x001f , RGB_BLACK = 0x0000 ,
  RGB_WHITE = 0xffff , RGB_GRAY = 0x8c51 , RGB_YELLOW = 0xFFE0 , RGB_CYAN = 0x07FF ,
  RGB_PURPLE = 0xF81F }
- enum directions {
  TEXT_DIRECTION0 = 0 , TEXT_DIRECTION90 = 1 , TEXT_DIRECTION180 = 2 , TEXT_DIRECTION270 =
  3 ,
  NUM_TEXT_DIRECTIONS }

Functions

- void display_init (display_t ∗display)
- void display_destroy (display_t ∗display)
- void displayDrawPixel (display_t ∗display, uint16_t x, uint16_t y, uint16_t color)
- void displayDrawFillRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayFillScreen (display_t ∗display, uint16_t color)
- void displayDrawLine (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDrawRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDrawRectAngle (display_t ∗display, uint16_t xc, uint16_t yc, uint16_t w, uint16_t h, uint16_t angle,
  uint16_t color)
- void displayDrawTriangleCenter (display_t ∗display, uint16_t xc, uint16_t yc, uint16_t w, uint16_t h, uint16_t
  angle, uint16_t color)
- void displayDrawCircle (display_t ∗display, uint16_t x_center, uint16_t y_center, uint16_t r, uint16_t color)
- void displayDrawFillCircle (display_t ∗display, uint16_t x_center, uint16_t y_center, uint16_t r, uint16_t color)
- void displayDrawRoundRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t r,
  uint16_t color)
- uint16_t rgb_conv (uint16_t r, uint16_t g, uint16_t b)
- int displayDrawChar (display_t ∗display, FontxFile ∗fx, uint16_t x, uint16_t y, uint8_t ascii, uint16_t color)
- int displayDrawString (display_t ∗display, FontxFile ∗fx, uint16_t x, uint16_t y, uint8_t ∗ascii, uint16_t color)
- void displaySetFontDirection (display_t ∗display, uint16_t dir)
- void displaySetFontFill (display_t ∗display, uint16_t color)
- void displayUnsetFontFill (display_t ∗display)
- void displaySetFontUnderLine (display_t ∗display, uint16_t color)
- void displayUnsetFontUnderLine (display_t ∗display)
- void displayDisplayOff (display_t ∗display)
- void displayDisplayOn (display_t ∗display)
- void displayBacklightOff (display_t ∗display)
- void displayBacklightOn (display_t ∗display)
- void displayInversionOff (display_t ∗display)
- void displayInversionOn (display_t ∗display)
- void displayDrawTriangle (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t x3,
  uint16_t y3, uint16_t color)
- void display_set_flip (display_t ∗display, bool xflip, bool yflip)

4.5.1 Detailed Description

Wrappers to simplify the use of the TFT LCD display.

Define a display_t display (called the display "handle"), initialise it, and pass this as the first parameter to all functions.

Warning

> All functions fail with program exit if any pixel of the shape that is drawn is outside the display dimensions.

An example of how to use this library.

```
#include <libpynq.h>
int main (void)
{
  // initialise all I/O
  pynq_init();
  display_t display;
  display_init(&display);

  displayFillScreen(&display, RGB_RED);
  // drawing is simple
  displayDrawPixel(&display, 50, 50, RGB_YELLOW);
  displayDrawFillRect(&display, 10, 100, 110, 200, RGB_RED);
  displayDrawCircle(&display, 60, 40, 15, RGB_RED);
  // text is more involved
  FontxFile fx16G[2];
  // the font file must be reachable from the directory
  // from which the executable is run -- see InitFontx
  InitFontx(fx16G, "../../fonts/ILGH16XB.FNT", "");
  GetFontx(fx16G, 0, buffer_fx16G, &fontWidth_fx16G, &fontHeight_fx16G);
  displaySetFontDirection(&display, TEXT_DIRECTION0);
  uint8_t text[] = "hello";
  displayDrawString(&display, fx16G, 15, fontHeight_fx16G * 6, text1,
RGB_WHITE);

  // clean up after use
  display_destroy(&display);
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

4.5.2 Macro Definition Documentation

4.5.2.1 DISPLAY_HEIGHT

#define DISPLAY_HEIGHT 240

Definition at line 83 of file display.h.

4.5.2.2 DISPLAY_WIDTH

#define DISPLAY_WIDTH 240

Definition at line 84 of file display.h.

4.5.3 Enumeration Type Documentation

4.5.3.1 colors

enum colors

Colors that can be used with the display.

Enumerator

RGB_RED	
RGB_GREEN	
RGB_BLUE	
RGB_BLACK	
RGB_WHITE	
RGB_GRAY	
RGB_YELLOW	
RGB_CYAN	
RGB_PURPLE	

Definition at line 89 of file display.h.

4.5.3.2 directions

```
enum directions
```

Enum of directions the text can be printed on on the display.

Enumerator

TEXT_DIRECTION0	
TEXT_DIRECTION90	
TEXT_DIRECTION180	
TEXT_DIRECTION270	
NUM_TEXT_DIRECTIONS	

Definition at line 104 of file display.h.

4.5.4 Function Documentation

4.5.4.1 display_destroy()

```
void display_destroy (
            display_t * display ) [extern]
```

Stop using the display.

Parameters

display	Handle to display.

4.5.4.2 display_init()

```
void display_init (
            display_t * display ) [extern]
```

Initialize the display display.

Parameters

display	Handle to display.

Definition at line 301 of file display.c.

Here is the call graph for this function:

4.5.4.3 display_set_flip()

```
void display_set_flip (
            display_t * display,
            bool xflip,
            bool yflip )
```

Flip the drawing off the screen.

Parameters

display	Handle to display
xflip	Flip in the X direction
yflip	Flip in the Y direction

Definition at line 279 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.5.4.4 displayBacklightOff()

```
void displayBacklightOff (
            display_t * display ) [extern]
```

Turn off the display backlight.

Parameters

display	Handle to display.

Definition at line 1014 of file display.c.

Here is the call graph for this function:

4.5.4.5 displayBacklightOn()

```
void displayBacklightOn (
            display_t * display ) [extern]
```

Turn on the display backlight.

Parameters

display	Handle to display.

Definition at line 1023 of file display.c.

Here is the call graph for this function:

4.5.4.6 displayDisplayOff()

```
void displayDisplayOff (
            display_t * display )  [extern]
```

Turn off the display.

Parameters

display	Handle to display.

Definition at line 403 of file display.c.

Here is the call graph for this function:

4.5.4.7 displayDisplayOn()

```
void displayDisplayOn (
            display_t * display )  [extern]
```

Initialize DISPLAY screen.

Parameters

display	Handle to display.
width	Width of screen in pixels.
height	Height of screen in pixels.
offsetx	Horizontal offset.
offsety	Vertical offset.

Definition at line 410 of file display.c.

Here is the call graph for this function:

4.5.4.8 displayDrawChar()

```
int displayDrawChar (
            display_t * display,
            FontxFile * fx,
```

```
            uint16_t x,
            uint16_t y,
            uint8_t ascii,
            uint16_t color ) [extern]
```

Draws a character on the given coordinates of the display.

Parameters

display	Handle to display.
fx	Pointer to font-file that is used for drawing the text.
x	The x-coordinate of the text on the display.
y	The y-coordinate of the text on the display.
ascii	The ascii character to draw.
color	The 16-bit color value to write.

Returns

The x-value of the next character to be printed on the display.

Warning

The font-file path must be valid from the directory in which the executable is called, otherwise the error message "cannot get font from font file" will be thrown. Absolute paths (starting with /) are safe. See documentation for InitFontx.

Definition at line 782 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.5.4.9 displayDrawCircle()

```
void displayDrawCircle (
            display_t * display,
            uint16_t x_center,
            uint16_t y_center,
            uint16_t r,
            uint16_t color ) [extern]
```

Draw a circle without infill on the display.

Parameters

display	Handle to display.
x_center	X-coordinate of the center of the circle.
y_center	Y-coordinate of the center of the circle.
r	The radius of the circle in pixels.
color	The 16-bit color value to write.

Definition at line 621 of file display.c.

Here is the call graph for this function:

4.5.4.10 displayDrawFillCircle()

```
void displayDrawFillCircle (
            display_t * display,
            uint16_t x_center,
            uint16_t y_center,
            uint16_t r,
            uint16_t color )  [extern]
```

Draw a circle with infill on the display.

Parameters

display	Handle to display.
x_center	X-coordinate of the center of the circle.
y_center	Y-coordinate of the center of the circle.
r	The radius of the circle in pixels.
color	The 16-bit color value to write.

Definition at line 662 of file display.c.

Here is the call graph for this function:

4.5.4.11 displayDrawFillRect()

```
void displayDrawFillRect (
            display_t * display,
            uint16_t x1,
            uint16_t y1,
            uint16_t x2,
            uint16_t y2,
            uint16_t color )  [extern]
```

Draw a filled rectangle to the display.

Parameters

display	Handle to display.
x1	The X coordinate of the top-left corner of the rectangle.
y1	The Y coordinate of the top-left corner of the rectangle.
x2	The X coordinate of the bottom-right corner of the rectangle.
y2	The Y coordinate of the bottom-right corner of the rectangle.
color	The 16-bit color value to write.

Definition at line 361 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.5.4.12 displayDrawLine()

```
void displayDrawLine (
            display_t * display,
            uint16_t x1,
            uint16_t y1,
            uint16_t x2,
            uint16_t y2,
            uint16_t color )  [extern]
```

Draw a line from two coordinates.

Parameters

display	Handle to display.
x1	Starting x-coordinate of line.
y1	Starting y-coordinate of line.
x2	Ending x-coordinate of line.
y2	Ending y-coordinate of line.
color	The 16-bit color value to write.

Definition at line 425 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.5.4.13 displayDrawPixel()

```
void displayDrawPixel (
            display_t * display,
            uint16_t x,
            uint16_t y,
            uint16_t color )  [extern]
```

Draw a single pixel to the display.

Parameters

display	Handle to display.
x	The X coordinate of the pixel.
y	The Y coordinate of the pixel.
color	The 16-bit color value to write.

Definition at line 317 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.5.4.14 displayDrawRect()

```
void displayDrawRect (
            display_t * display,
```

```
            uint16_t x1,
            uint16_t y1,
            uint16_t x2,
            uint16_t y2,
            uint16_t color )  [extern]
```

Draw a filled rectangle.

Parameters

display	Handle to display.
x1	Top-left x-coordinate of rectangle.
y1	Top-left y-coordinate of rectangle.
x2	Bottom-right x-coordinate of rectangle.
y2	Bottom-right y-coordinate of rectangle.
color	The 16-bit color value to write.

Definition at line 478 of file display.c.

Here is the call graph for this function:

4.5.4.15   displayDrawRectAngle()

```
void displayDrawRectAngle (
            display_t * display,
            uint16_t xc,
            uint16_t yc,
            uint16_t w,
            uint16_t h,
            uint16_t angle,
            uint16_t color )  [extern]
```

Draws a rectangle with rounded corners at a specified angle on the display.

Parameters

display	Handle to display.
xc	X-coordinate of the center of the rectangle.
yc	Y-coordinate of the center of the rectangle.
w	Width of the rectangle.
h	Height of the rectangle.
angle	Angle of rotation in degrees.
color	The 16-bit color value to write.

Definition at line 496 of file display.c.

Here is the call graph for this function:

4.5.4.16   displayDrawRoundRect()

```
void displayDrawRoundRect (
            display_t * display,
```

```
        uint16_t x1,
        uint16_t y1,
        uint16_t x2,
        uint16_t y2,
        uint16_t r,
        uint16_t color ) [extern]
```

Draw a rectangle with rounded angles.

Parameters

display	Handle to display.
x1	Top-left x-coordinate of rectangle.
y1	Top-left y-coordinate of rectangle.
x2	Bottom-right x-coordinate of rectangle.
y2	Bottom-right y-coordinate of rectangle.
r	The radius of the circle that is used for the edges.
color	The 16-bit color value to write.

Definition at line 708 of file display.c.

Here is the call graph for this function:

4.5.4.17 displayDrawString()

```
int displayDrawString (
        display_t * display,
        FontxFile * fx,
        uint16_t x,
        uint16_t y,
        uint8_t * ascii,
        uint16_t color ) [extern]
```

Function to draw a string on the display.

Parameters

display	Handle to display.
fx	Pointer to font-file that is used for drawing the text.
x	The x-coordinate of the text on the display.
y	The y-coordinate of the text on the display.
ascii	The ascii characters to draw.
color	The 16-bit color value to write.

Returns

The x or y coordinate of the next character, depending on the orientation of the display.

Warning

The font-file path must be valid from the directory in which the executable is called, otherwise the error message "cannot get font from font file" will be thrown. Absolute paths (starting with /) are safe. See documentation for InitFontx.

Definition at line 951 of file display.c.

Here is the call graph for this function:

4.5.4.18 displayDrawTriangle()

```
void displayDrawTriangle (
            display_t * display,
            uint16_t x1,
            uint16_t y1,
            uint16_t x2,
            uint16_t y2,
            uint16_t x3,
            uint16_t y3,
            uint16_t color )  [extern]
```

Draw a triangle without infill between the three given points in the given color.

Parameters

display	Handle to display.
x1	The first X-coordinate of the triangle.
y1	The first Y-coordinate of the triangle.
x2	The second X-coordinate of the triangle.
y2	The second Y-coordinate of the triangle.
x3	The third X-coordinate of the triangle.
y3	The third Y-coordinate of the triangle.
color	The 16-bit color value to write.

Definition at line 553 of file display.c.

Here is the call graph for this function:

4.5.4.19 displayDrawTriangleCenter()

```
void displayDrawTriangleCenter (
            display_t * display,
            uint16_t xc,
            uint16_t yc,
            uint16_t w,
            uint16_t h,
            uint16_t angle,
            uint16_t color )  [extern]
```

Draws a triangle at a specified angle on the display.

Parameters

display	Handle to display.
xc	X-coordinate of the center of the rectangle.
yc	Y-coordinate of the center of the rectangle.
w	Width of the rectangle.
h	Height of the rectangle.
angle	Angle of rotation in degrees.
color	The 16-bit color value to write.

Definition at line 580 of file display.c.

Here is the call graph for this function:

4.5.4.20 displayFillScreen()

```
void displayFillScreen (
          display_t * display,
          uint16_t color )  [extern]
```

Fill entire display with a single color using the ldcDrawFillRect function.

Parameters

display	Handle to display.
color	Fill color in RGB format.

Definition at line 417 of file display.c.

Here is the call graph for this function:

4.5.4.21 displayInversionOff()

```
void displayInversionOff (
          display_t * display )  [extern]
```

Turn off inversion of the colors.

Parameters

display	Handle to display.

Definition at line 1032 of file display.c.

Here is the call graph for this function:

4.5.4.22 displayInversionOn()

```
void displayInversionOn (
          display_t * display )  [extern]
```

Turn on inversion of the colors.

Parameters

display	Handle to display.

Definition at line 1039 of file display.c.

Here is the call graph for this function:

4.5.4.23 displaySetFontDirection()

```
void displaySetFontDirection (
            display_t * display,
            uint16_t dir )  [extern]
```

Changes the direction the characters will be printed.

Parameters

display	Handle to display.
dir	The direction to set the font in the display handle.

Definition at line 982 of file display.c.

4.5.4.24 displaySetFontFill()

```
void displaySetFontFill (
            display_t * display,
            uint16_t color )  [extern]
```

Enables the _font_fill and sets the _font_fill_color in the display handle.

Parameters

display	Handle to display.
color	The fill-color the font should have

Definition at line 989 of file display.c.

4.5.4.25 displaySetFontUnderLine()

```
void displaySetFontUnderLine (
            display_t * display,
            uint16_t color )  [extern]
```

Turns on _font_underline in the display handle and sets the _font_underline_color to the specified color.

Parameters

display	Handle to display.
color	The 16-bit color value to write.

Definition at line 999 of file display.c.

4.5.4.26 displayUnsetFontFill()

```
void displayUnsetFontFill (
            display_t * display ) [extern]
```

Sets the _font_fill parameter to false in the display handle, turns off the font fill.

Parameters

display	Handle to display.

Definition at line 997 of file display.c.

4.5.4.27 displayUnsetFontUnderLine()

```
void displayUnsetFontUnderLine (
            display_t * display ) [extern]
```

Turns off _font_underline in the display handle.

Parameters

display	Handle to display.

Definition at line 1007 of file display.c.

4.5.4.28 rgb_conv()

```
uint16_t rgb_conv (
            uint16_t r,
            uint16_t g,
            uint16_t b ) [extern]
```

RGB conversion for generating a color.

Parameters

r	Red value, 5 least significant bits.
g	Green value, 6 least significant bits.
b	Blue value, 5 least significant bits.

Definition at line 778 of file display.c.

4.6 Font library

Data Structures

- struct FontxFile

Typedefs

- typedef struct _IO_FILE FILE

Functions

- void AaddFontx (FontxFile ∗fx, const char ∗path)
- void InitFontx (FontxFile ∗fxs, const char ∗f0, const char ∗f1)
- bool OpenFontx (FontxFile ∗fx)
- void CloseFontx (FontxFile ∗fx)
- void DumpFontx (FontxFile ∗fxs)
- uint8_t GetFontWidth (FontxFile ∗fx)
- uint8_t GetFontHeight (FontxFile ∗fx)
- bool GetFontx (FontxFile ∗fxs, uint8_t ascii, uint8_t ∗pGlyph, uint8_t ∗pw, uint8_t ∗ph)
- void Font2Bitmap (uint8_t ∗fonts, uint8_t ∗line, uint8_t w, uint8_t h, uint8_t inverse)
- void UnderlineBitmap (uint8_t ∗line, uint8_t w, uint8_t h)
- void ReversBitmap (uint8_t ∗line, uint8_t w, uint8_t h)
- void ShowFont (uint8_t ∗fonts, uint8_t pw, uint8_t ph)
- void ShowBitmap (uint8_t ∗bitmap, uint8_t pw, uint8_t ph)
- uint8_t RotateByte (uint8_t ch)

4.6.1 Detailed Description

Do not use. Low-level library to work with bitmap fonts on the display.

It provides functionality for loading and manipulating font files, rendering fonts and bitmaps to the screen, and performing various transformations on bitmaps. The library also includes a struct, FontxFile, which represents a font file and contains metadata about the font.

This is an internal library and should not be directly used.

4.6.2 Typedef Documentation

4.6.2.1 FILE

```
typedef struct _IO_FILE FILE
```

Definition at line 23 of file fontx.h.

4.6.3 Function Documentation

4.6.3.1 AaddFontx()

```
void AaddFontx (
            FontxFile * fx,
            const char * path )
```

Adds a font file to the given FontxFile structure.

Parameters

fx	Pointer to the FontxFile structure
path	Path to the font file

4.6.3.2 CloseFontx()

```
void CloseFontx (
            FontxFile * fx )
```

Closes the font file.

Parameters

fx	Pointer to the FontxFile structure

Definition at line 67 of file fontx.c.

4.6.3.3 DumpFontx()

```
void DumpFontx (
            FontxFile * fxs )
```

Dumps the font data stored in the FontxFile structure.

Parameters

fxs	Pointer to the FontxFile structure

Definition at line 74 of file fontx.c.

4.6.3.4 Font2Bitmap()

```
void Font2Bitmap (
            uint8_t * fonts,
            uint8_t * line,
            uint8_t w,
            uint8_t h,
            uint8_t inverse )
```

Converts a font data buffer into a bitmap.

Parameters

fonts	Pointer to the font data buffer
line	Pointer to the bitmap buffer
w	Width of the bitmap in pixels
h	Height of the bitmap in pixels
inverse	If true, the bitmap will be inverted

Definition at line 135 of file fontx.c.

Here is the call graph for this function:

4.6.3.5 GetFontHeight()

```
uint8_t GetFontHeight (
            FontxFile * fx )
```

Gets the height of a character in the font.

Parameters

fx	Pointer to the FontxFile structure

Returns

The height of a character in pixels

4.6.3.6 GetFontWidth()

```
uint8_t GetFontWidth (
            FontxFile * fx )
```

Gets the width of a character in the font.

Parameters

fx	Pointer to the FontxFile structure

Returns

The width of a character in pixels

4.6.3.7 GetFontx()

```
bool GetFontx (
            FontxFile * fxs,
            uint8_t ascii,
            uint8_t * pGlyph,
            uint8_t * pw,
            uint8_t * ph )
```

Gets the glyph data for the specified ASCII character.

Parameters

fxs	Pointer to the FontxFile structure

Parameters

ascii	ASCII value of the character to get the glyph for
pGlyph	Pointer to the buffer to store the glyph data
pw	Pointer to the variable to store the width of the glyph
ph	Pointer to the variable to store the height of the glyph

Returns

True if the glyph was found, false otherwise

Definition at line 98 of file fontx.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.6.3.8   InitFontx()

```
void InitFontx (
            FontxFile * fxs,
            const char * f0,
            const char * f1 )
```

Initializes the given FontxFile structure with the specified font files.

Parameters

fxs	Pointer to the FontxFile structure
f0	Path to the 8x16 font file
f1	Path to the 16x16 font file

Definition at line 17 of file fontx.c.

Here is the call graph for this function:

4.6.3.9   OpenFontx()

```
bool OpenFontx (
            FontxFile * fx )
```

Opens the font file and reads the font data into the FontxFile structure.

Parameters

fx	Pointer to the FontxFile structure

Returns

True if the font file was opened successfully, false otherwise

Warning

> The font-file path must be valid from the directory in which the executable is called, otherwise the error message "cannot get font from font file" will be thrown. Absolute paths (starting with /) are safe.

Definition at line 22 of file fontx.c.

Here is the caller graph for this function:

4.6.3.10 ReversBitmap()

```
void ReversBitmap (
            uint8_t * line,
            uint8_t w,
            uint8_t h )
```

Reverses the bits in each byte of a bitmap.

Parameters

line	Pointer to the bitmap buffer
w	Width of the bitmap in pixels
h	Height of the bitmap in pixels

Definition at line 181 of file fontx.c.

4.6.3.11 RotateByte()

```
uint8_t RotateByte (
            uint8_t ch )
```

Rotates a byte by 90 degrees.

Parameters

ch	Byte to be rotated

Returns

> The rotated byte

Definition at line 234 of file fontx.c.

Here is the caller graph for this function:

4.6.3.12 ShowBitmap()

```
void ShowBitmap (
            uint8_t * bitmap,
```

```
uint8_t pw,
uint8_t ph )
```

Displays a bitmap on the screen.

Parameters

bitmap	Pointer to the bitmap buffer
pw	Width of the font in pixels
ph	Height of the font in pixels

Definition at line 211 of file fontx.c.

4.6.3.13 ShowFont()

```
void ShowFont (
            uint8_t * fonts,
            uint8_t pw,
            uint8_t ph )
```

Displays a font on the screen.

Parameters

fonts	Pointer to the font buffer
pw	Width of the font in pixels
ph	Height of the font in pixels

Definition at line 192 of file fontx.c.

4.6.3.14 UnderlineBitmap()

```
void UnderlineBitmap (
            uint8_t * line,
            uint8_t w,
            uint8_t h )
```

Adds an underline to a bitmap.

Parameters

line	Pointer to the bitmap buffer
w	Width of the bitmap in pixels
h	Height of the bitmap in pixels

Definition at line 169 of file fontx.c.

4.7 GPIO library

Enumerations

- enum gpio_direction_t { GPIO_DIR_INPUT = 0 , GPIO_DIR_OUTPUT = 1 }
- enum gpio_level_t { GPIO_LEVEL_LOW = 0 , GPIO_LEVEL_HIGH = 1 }

Functions

- void gpio_init (void)
- void gpio_destroy (void)
- void gpio_reset_pin (const io_t pin)
- void gpio_set_direction (const io_t pin, const gpio_direction_t direction)
- gpio_direction_t gpio_get_direction (const io_t pin)
- void gpio_set_level (const io_t pin, const gpio_level_t level)
- gpio_level_t gpio_get_level (const io_t pin)
- void gpio_reset (void)
- bool gpio_is_initialized (void)

4.7.1 Detailed Description

Functions for General Purpose I/O (GPIO) access to leds, buttons, (analog) pins, etc.

All functions use the IO pin number (io_t) from 0..IO_NUM_PINS-1.

The LED and button libraries are built on top of this library, but do not expose the full functionality of this library. Use this library when that is required. Also see the I/O switchbox (switchbox.h) and pin mapping (pinmap.h).

In particular, be aware that the numbering used in the high-level libraries is different from the underlying GPIO numbering.

- The button library uses 0..3 or BUTTON0..BUTTON3, and 0..1 or SWITCH0..SWITCH1, whereas GPIO uses IO_BTN0..IO_BTN3 and IO_SW0..IO_SW1.

- The LED library uses 0..3 or LED0..LED1 for green LEDs whereas GPIO uses IO_LD0..IO_LD3. It uses 0..1 or COLOR_LED0..COLOR_LED1 and the three color components (RGB) whereas GPIO uses IO_LD4/5↩R/G/B.

- The PWM library uses 0..5 or PWM0..PWM5, whereas GPIO uses SWB_PWM0..SWB_PWM5.

- The UART library uses 0..1 or UART0..UART1, whereas GPIO uses SWB_UART0..SWB_UART1.

- The ADC library is slightly different. It uses ADC0..ADC5 (these are non-consecutive numbers), whereas GPIO uses IO_A0..IO_A5 (which are consecutive).

An example of using this library to turn LED0 on:
```
#include <libpynq.h>
int main (void)
{
  gpio_init();
  // set pin A0 to be an input pin and read from it
  gpio_set_direction(IO_A0, GPIO_DIR_INPUT);
  gpio_level_t c = gpio_get_level(IO_A0);
  // alternatively, set A0 to be an output pin and write to it
  gpio_set_direction(IO_A0, GPIO_DIR_OUTPUT);
  gpio_set_level(IO_A0, GPIO_LEVEL_LOW);
  sleep_msec(100);
  gpio_set_level(IO_A0, GPIO_LEVEL_HIGH);

  // set LED 0 as output
  gpio_set_direction(IO_LD0, GPIO_DIR_OUTPUT);
  // turn LED 0 on
  gpio_set_level(IO_LD0, GPIO_LEVEL_HIGH);
  sleep_msec(1000);
  leds_destroy(); // turn LEDs off
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

4.7.2 Enumeration Type Documentation

4.7.2.1 gpio_direction_t

enum gpio_direction_t

Enumerate the direction state (input/output) of the pin

Enumerator

GPIO_DIR_INPUT	The IO pin is an input.
GPIO_DIR_OUTPUT	The IO pin is an output.

Definition at line 88 of file gpio.h.

4.7.2.2 gpio_level_t

`enum gpio_level_t`

Enumerate the signal level.

Enumerator

GPIO_LEVEL_LOW	A low signal
GPIO_LEVEL_HIGH	A high signal

Definition at line 98 of file gpio.h.

4.7.3 Function Documentation

4.7.3.1 gpio_destroy()

```
void gpio_destroy (
            void ) [extern]
```

De-initialize the GPIO library. This releases the memory map and memory allocated by gpio_init.

Definition at line 47 of file gpio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.7.3.2 gpio_get_direction()

```
gpio_direction_t gpio_get_direction (
            const io_t pin ) [extern]
```

Returns the direction the set pin is initialized in.

Parameters

pin	The IO pin to read the direction set in the shared memory system on the ARM processor.

Warning

> Fails with program exit when pin is outside valid range.

Definition at line 95 of file gpio.c.

Here is the caller graph for this function:

4.7.3.3 gpio_get_level()

```
gpio_level_t gpio_get_level (
            const io_t pin )  [extern]
```

Return the level of the IO pin.

Parameters

pin	The IO pin to read it state.

Returns

the output level of pin.

Warning

Fails with program exit when pin is outside valid range.

Definition at line 118 of file gpio.c.

Here is the caller graph for this function:

4.7.3.4 gpio_init()

```
void gpio_init (
            void  )  [extern]
```

Initializes the GPIO library.

Definition at line 40 of file gpio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.7.3.5 gpio_is_initialized()

```
bool gpio_is_initialized (
            void  )  [extern]
```

Check if gpio library is initialized.

Returns

true if initialize, false if not.

Definition at line 35 of file gpio.c.

Here is the caller graph for this function:

4.7.3.6 gpio_reset()

```
void gpio_reset (
            void ) [extern]
```

Reset all IO pins.

Definition at line 62 of file gpio.c.

Here is the caller graph for this function:

4.7.3.7 gpio_reset_pin()

```
void gpio_reset_pin (
            const io_t pin ) [extern]
```

Function is currently a no-op placeholder for arduino compatibility.

Parameters

pin	The IO pin to reset.

Warning

> Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 55 of file gpio.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.7.3.8 gpio_set_direction()

```
void gpio_set_direction (
            const io_t pin,
            const gpio_direction_t direction ) [extern]
```

Set the IO pin as in input or output.

Parameters

pin	The IO pin to modify direction for.
direction	The direction to set on the pin.

Warning

> Fails with program exit when pin or direction is outside valid range.

Definition at line 81 of file gpio.c.

Here is the caller graph for this function:

4.7.3.9 gpio_set_level()

```
void gpio_set_level (
            const io_t pin,
            const gpio_level_t level ) [extern]
```

Set the level of the output IO pin. If the pin is configured as input, this function does nothing.

Parameters

pin	The IO pin to modify direction for.
level	The level to set on the pin.

Warning

Fails with program exit when pin is outside valid range.

Definition at line 104 of file gpio.c.

Here is the caller graph for this function:

4.8 IIC library

Enumerations

- enum iic_index_t { IIC0 = 0 , IIC1 = 1 , NUM_IICS = 2 }

Functions

- void iic_init (const iic_index_t iic)
- void iic_destroy (const iic_index_t iic)
- bool iic_read_register (const iic_index_t iic, const uint8_t addr, const uint8_t reg, uint8_t ∗data, uint16_↩
  t length)
- bool iic_write_register (const iic_index_t iic, const uint8_t addr, const uint8_t reg, uint8_t ∗data, uint16_t
  length)
- bool iic_set_slave_mode (const iic_index_t iic, const uint8_t addr, uint32_t ∗register_map, const uint32_t
  rm_length)
- void iic_slave_mode_handler (const iic_index_t iic)
- void iic_reset (const iic_index_t iic)

4.8.1 Detailed Description

Functions to use the Inter-Integrated Circuit (IIC).

High-level functions to read/write to clients connected to the two integrated IIC modules. Before sending and receiving bytes the IIC2 must be connect to some I/O pins through the switchbox, e.g.

```
switchbox_set_pin(IO_PMODA3, SWB_IIC0_SCL);
switchbox_set_pin(IO_PMODA4, SWB_IIC0_SDA);
```

The Pmod A pins (see pinmap.h) are good because they have 2K2 pull-up resistors built in. IICs can be routed through the switch box (e.g. SWB_IIC0_SCL/SDA, see switchbox.h).

After that, an example of how to use this library for the MASTER.

```
#include <libpynq.h>
int main (void)
{
  // initialise all I/O
  pynq_init();
  iic_init(IIC0);
  uint32_t i;
  // you can use multiple slaves, here only one is shown
  uint32_t slave_address = 0x70;
  // read out all registers of the slave
  for (int reg=0; reg < 32; reg++) {
    if (iic_read_register(IIC0, slave_address, reg, (uint8_t *) &i, 4)) { // 4 bytes
      printf("register[%d]=error\n",reg);
    } else {
      printf("register[%d]=%d\n",reg,i);
    }
  }
  // clean up after use
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

An example of how to use this library for the SLAVE.

```
int main(void)
{
  // this is the address by which this slave is reached by the master
  // different slaves must have different addresses
  const uint32_t my_slave_address = 0x70;
  // this array contains 32 registers that can be written & read by the master
  // the slave can of course modify the values of the registers
  uint32_t my_register_map[32] =
      {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32};
  const uint32_t my_register_map_length = sizeof(my_register_mmap)/sizeof(uint32_t);

  pynq_init();
  iic_init(IIC0);
  iic_reset(IIC0);
  iic_set_slave_mode(IIC0, my_slave_address, &(my_register_map[0]), my_register_map_length);
  while (1) {
    // the slave mode handler must be run regularly to react to the master
    iic_slave_mode_handler(IIC0);
    // insert your own code here, to do whatever the slave needs to do;
    // make sure that you execute the slave mode handler regularly enough though
    sleep_msec(10);
  }
  iic_destroy(IIC0);
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

4.8.2 Enumeration Type Documentation

4.8.2.1 iic_index_t

```
enum iic_index_t
```

Enum of IICs. Functions use a switch numbered from 0..NUM_IICS-1.

Enumerator

IIC0	
IIC1	
NUM_IICS	

Definition at line 104 of file iic.h.

4.8.3 Function Documentation

4.8.3.1 iic_destroy()

```
void iic_destroy (
            const iic_index_t iic )  [extern]
```

Close the shared memory handle for the specified IIC index.

Parameters

uart	The IIC index to remove from the shared memory space.

Warning

> Fails with program exit if the IIC channel is outside valid range.

Definition at line 124 of file iic.c.

Here is the call graph for this function:

4.8.3.2 iic_init()

```
void iic_init (
            const iic_index_t iic )  [extern]
```

Initialize the IIC specified by the index with a shared memory handle and a buffer size of 4096 bytes.

Parameters

uart	The IIC index to initialize.

Warning

> Fails with program exit if the IIC channel is outside valid range or when the shared memory system has not been instantiated.

Definition at line 108 of file iic.c.

Here is the call graph for this function:

4.8.3.3 iic_read_register()

```
bool iic_read_register (
            const iic_index_t iic,
            const uint8_t addr,
            const uint8_t reg,
            uint8_t * data,
            uint16_t length )  [extern]
```

Parameters

iic	The IIC index to initialize.
addr	The IIC address of the client to access.
reg	The clients register address.
data	Buffer where the register content is stored. [out]
length	The amount of data to read.

Reads the content of the register into data.

Returns

> 0 if successful, 1 on error

Definition at line 327 of file iic.c.

Here is the call graph for this function:

4.8.3.4 iic_reset()

```
void iic_reset (
            const iic_index_t iic )  [extern]
```

Parameters

iic	The IIC index of the hardware to use. Return the IIC module into its default mode. This way it can be used as master.

Definition at line 314 of file iic.c.

4.8.3.5 iic_set_slave_mode()

```
bool iic_set_slave_mode (
            const iic_index_t iic,
            const uint8_t addr,
            uint32_t * register_map,
            const uint32_t rm_length )  [extern]
```

Definition at line 135 of file iic.c.

4.8.3.6 iic_slave_mode_handler()

```
void iic_slave_mode_handler (
            const iic_index_t iic ) [extern]
```

Parameters

iic	The IIC index of the hardware to use.

This handles requests that came in to the IIC unit when it is in slave mode.

Definition at line 302 of file iic.c.

4.8.3.7 iic_write_register()

```
bool iic_write_register (
            const iic_index_t iic,
            const uint8_t addr,
            const uint8_t reg,
            uint8_t * data,
            uint16_t length ) [extern]
```

Parameters

iic	The IIC index to initialize.
addr	The IIC address of the client to access.
reg	The clients register address.
data	Buffer where new the register content is stored.
length	The amount of data to write.

Writes data to register.

Returns

0 if successful, 1 on error

Definition at line 344 of file iic.c.

Here is the call graph for this function:

4.9 Interrupt library

Functions

- int gpio_interrupt_init (void)
- void gpio_ack_interrupt (void)
- void verify_interrupt_request (const io_t pin)
- void gpio_print_interrupt (void)
- void gpio_enable_interrupt (const io_t pin)
- void gpio_disable_interrupt (const io_t pin)
- void gpio_disable_all_interrupts (void)
- uint64_t gpio_get_interrupt (void)
- uint8_t ∗ gpio_get_interrupt_pins (uint8_t ∗positions)
- void gpio_wait_for_interrupt (const io_t pin)

4.9.1 Detailed Description

Functions for interrupt handling.

An example of using this library

```
#include <libpynq.h>
int main (void)
{
  gpio_init(void);
  gpio_reset(void);
  switchbox_init(void);
  switchbox_reset(void);
  gpio_set_direction(IO_LD0, GPIO_DIR_OUTPUT);
  // initialize the interrupt
  gpio_interrupt_init(void);
  gpio_enable_interrupt(IO_BTN0);
  gpio_set_direction(IO_LD0, GPIO_DIR_OUTPUT);
  while(1) {
    gpio_wait_for_interrupt(64); //Wait untill an interupt arrives
    uint8_t* interruptPin = gpio_get_interrupt_pins(void);
    if (interruptPin[0] == IO_BTN0) {
      printf("interrupt on IO_BTN0, turning on IO_LD0\n");
      gpio_set_level(IO_LD0, 1);
    } else {
      printf("interrupt on pin %d\n",interruptPin[0]);
      gpio_set_level(IO_LD0, 0);
    }
    gpio_ack_interrupt(void);
  }
  gpio_destroy(void);
  switchbox_destroy(void);
  return EXIT_SUCCESS;
}
```

4.9.2 Function Documentation

4.9.2.1 gpio_ack_interrupt()

```
void gpio_ack_interrupt (
            void  )  [extern]
```

acknowledges the raised interrupts and resets the interrupt word. Allows new interrupts to occur on the previously triggered pins.

Definition at line 91 of file interrupt.c.

Here is the call graph for this function:

4.9.2.2 gpio_disable_all_interrupts()

```
void gpio_disable_all_interrupts (
            void  )  [extern]
```

Disables all interrupts from being raised.

Definition at line 77 of file interrupt.c.

Here is the call graph for this function:

4.9.2.3 gpio_disable_interrupt()

```
void gpio_disable_interrupt (
            const io_t pin )  [extern]
```

Disables interrupts from occuring on the specific pin. Hereafter, the pin will not trigger an interrupt.

Parameters

pin	to be disabled from obtianing interrupts

Definition at line 72 of file interrupt.c.

Here is the call graph for this function:

4.9.2.4 gpio_enable_interrupt()

```
void gpio_enable_interrupt (
            const io_t pin ) [extern]
```

enables a specific pin to raise interrupts.

Parameters

pin	to raise interrupts

Definition at line 59 of file interrupt.c.

Here is the call graph for this function:

4.9.2.5 gpio_get_interrupt()

```
uint64_t gpio_get_interrupt (
            void ) [extern]
```

Returns

the 64 bits on which interrupts are indicated by a one. The bits are in accordance with the pins described in pinmap.h

Definition at line 83 of file interrupt.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.9.2.6 gpio_get_interrupt_pins()

```
uint8_t * gpio_get_interrupt_pins (
            uint8_t * positions ) [extern]
```

Gets all pins on which an interrupt occurred.

Returns

a pointer to an array of maximum 64 intergers. The integers correspond to pins with a pending interrupt.

Definition at line 160 of file interrupt.c.

Here is the call graph for this function:

4.9.2.7 gpio_interrupt_init()

```
int gpio_interrupt_init (
            void ) [extern]
```

Enables interrupts to be set and read.

Definition at line 48 of file interrupt.c.

4.9.2.8 gpio_print_interrupt()

```
void gpio_print_interrupt (
            void ) [extern]
```

prints the current interrupt word

Definition at line 117 of file interrupt.c.

Here is the call graph for this function:

4.9.2.9 gpio_wait_for_interrupt()

```
void gpio_wait_for_interrupt (
            const io_t pin ) [extern]
```

Waits untill an interrupt occurs on the specified pin or if the value of pin is larger than 63, if any interrupt has occurred.

Parameters

pin	The pin on which an interrupt should occur

Definition at line 138 of file interrupt.c.

Here is the call graph for this function:

4.9.2.10 verify_interrupt_request()

```
void verify_interrupt_request (
            const io_t pin ) [extern]
```

Checks for error in enabled pin. Terminates the process if the pin is not enabled.

Parameters

pin	indicates a specific pin or if larger than 63, if any interrupt pin is enabled

Definition at line 96 of file interrupt.c.

Here is the caller graph for this function:

4.10 LED library

Macros

- #define NUM_LED_COLORS 3 /∗ # colors per color LED (RGB) ∗/
- #define NUM_LEDS (NUM_GREEN_LEDS + NUM_COLOR_LEDS)
- #define LED_OFF 0
- #define LED_ON 255

Enumerations

- enum green_led_index_t {
  LED0 , LED1 , LED2 , LED3 ,
  NUM_GREEN_LEDS }
- enum color_led_index_t { COLOR_LED0 , COLOR_LED1 , NUM_COLOR_LEDS }

Functions

- void leds_init_onoff (void)
- void green_leds_init_pwm (void)
- void color_leds_init_pwm (void)
- void leds_destroy (void)
- void green_led_onoff (const int led, const int onoff)
- void green_led_on (const int led)
- void green_led_off (const int led)
- void color_led_red_onoff (const int onoff)
- void color_led_green_onoff (const int onoff)
- void color_led_blue_onoff (const int onoff)
- void color_led_onoff (const int red_onoff, const int green_onoff, const int blue_onoff)
- void color_led_on (void)
- void color_led_off (void)

4.10.1 Detailed Description

Wrappers to simplify the use of LEDs.

- Green LEDs are numbered 0 to NUM_GREEN_LEDS-1.

- Only color LED 0 is used.

- The color LED has three components R, G, B that can be set independently to mix to a color.

LEDs can be used in three modes:

1. on/off mode for all green LEDs and all color LEDs

2. PWM mode for green LEDs (PWM0..PWM3 are rounted to green LEDs 0..3)

3. PWM mode for color LED 0 (PWM0..PWM3 are routed to color LED 0)

An example of how to use this library.

```
#include <libpynq.h>
int main (void)
{
  // initialise all I/O
  gpio_reset();
  leds_init_onoff();

  for (int led = 0; led < NUM_GREEN_LEDS; led++)
    green_led_on(led);
  sleep_msec(500);
  for (int led = 0; led < NUM_GREEN_LEDS; led++)
    green_led_off(led);

  // clean up after use
  leds_destroy(); // switches all leds off
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

LEDs can also be used through GPIO (see gpio.h and pinmap.h). Note that GPIO numbering (IO_LD0..IO_LD3) is then used instead of 0..NUM_GREEN_LEDS-1 (LED0..LED3). In the PWM mode for color LED 0, SWB_↩
PWM0..SWB_PWM3 are routed to color LED 0 (GPIO IO_LD4R, IO_LD4G, IO_LD4B).

4.10.2 Macro Definition Documentation

4.10.2.1 LED_OFF

#define LED_OFF 0

Definition at line 102 of file leds.h.

4.10.2.2 LED_ON

#define LED_ON 255

Definition at line 103 of file leds.h.

4.10.2.3 NUM_LED_COLORS

#define NUM_LED_COLORS 3 /* # colors per color LED (RGB) */

Definition at line 100 of file leds.h.

4.10.2.4 NUM_LEDS

#define NUM_LEDS (NUM_GREEN_LEDS + NUM_COLOR_LEDS)

Definition at line 101 of file leds.h.

4.10.3 Enumeration Type Documentation

4.10.3.1 color_led_index_t

enum color_led_index_t

Enum of color LEDs. Functions for color LEDs use a led number from 0..NUM_COLOR_LEDS-1. Alternatively, you can use COLOR_LEDi instead of just i if you find that clearer.

Enumerator

COLOR_LED0	
COLOR_LED1	
NUM_COLOR_LEDS	

Definition at line 94 of file leds.h.

4.10.3.2 green_led_index_t

enum green_led_index_t

Enum of green LEDs. Functions for green LEDs use a led number from 0..NUM_GREEN_LEDS-1. Alternatively, you can use LEDi instead of just i if you find that clearer.

Enumerator

LED0	
LED1	
LED2	
LED3	
NUM_GREEN_LEDS	

Definition at line 80 of file leds.h.

4.10.4 Function Documentation

4.10.4.1 color_led_blue_onoff()

```
void color_led_blue_onoff (
            const int onoff )  [extern]
```

Switches on/off the blue component of color LED 0.

Parameters

onoff	If the LEDs are in onoff mode then onoff must be either LED_ON or LED_OFF. If the LEDs are in one of the PWM modes then onoff must be 0..255.

Warning

Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 195 of file leds.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.10.4.2 color_led_green_onoff()

```
void color_led_green_onoff (
            const int onoff )  [extern]
```

Switches on/off the green component of color LED 0.

Parameters

onoff	If the LEDs are in onoff mode then onoff must be either LED_ON or LED_OFF. If the LEDs are in one of the PWM modes then onoff must be 0.255.

Warning

> Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 172 of file leds.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.10.4.3 color_led_off()

```
void color_led_off (
            void )  [extern]
```

Set color LED 0 to black. Same as color_led_onoff(LED_OFF, LED_OFF, LED_OFF).

Warning

> Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 226 of file leds.c.

Here is the call graph for this function:

4.10.4.4 color_led_on()

```
void color_led_on (
            void )  [extern]
```

Set color LED 0 to white. Same as color_led_onoff(LED_ON, LED_ON, LED_ON).

Warning

> Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 225 of file leds.c.

Here is the call graph for this function:

4.10.4.5 color_led_onoff()

```
void color_led_onoff (
            const int red_onoff,
            const int green_onoff,
            const int blue_onoff )  [extern]
```

Switches on/off the red/green/blue components of color LED 0.

Parameters

onoff	If the LEDs are in onoff mode then ∗_onoff must be either LED_ON or LED_OFF. If the LEDs are in one of the PWM modes then ∗_onoff must be 0.255.

Warning

> Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 218 of file leds.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.10.4.6 color_led_red_onoff()

```
void color_led_red_onoff (
            const int onoff )  [extern]
```

Switches on/off the red component of color LED 0.

Parameters

onoff	If the LEDs are in onoff mode then onoff must be either LED_ON or LED_OFF. If the LEDs are in one of the PWM modes then onoff must be 0.255.

Warning

> Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 148 of file leds.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.10.4.7 color_leds_init_pwm()

```
void color_leds_init_pwm (
            void )  [extern]
```

Initialize the color LEDs for use with variable intensity. The LED intensity can range from 0.255.

Warning

> Fails with program exit when LEDs have already been to another mode.

Definition at line 79 of file leds.c.

Here is the call graph for this function:

4.10.4.8 green_led_off()

```
void green_led_off (
            const int led )  [extern]
```

Same as green_led_onoff(led, LED_OFF). Works in all modes.

Parameters

led	The green LED.

Warning

Fails with program exit when led is outside valid range.

Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 147 of file leds.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.10.4.9 green_led_on()

```
void green_led_on (
            const int led )  [extern]
```

Same as green_led_onoff(led, LED_ON). Works in all modes.

Parameters

led	The green LED.

Warning

Fails with program exit when led is outside valid range.

Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 146 of file leds.c.

Here is the call graph for this function:

4.10.4.10 green_led_onoff()

```
void green_led_onoff (
            const int led,
            const int onoff )  [extern]
```

Parameters

led	The green LED.
onoff	If the LEDs are in onoff mode then onoff must be either LED_ON or LED_OFF. If the LEDs are in one of the PWM modes then onoff must be 0.255.

Warning

Fails with program exit when led is outside valid range.

Fails with program exit when LEDs were not initialized with the correct mode.

Definition at line 117 of file leds.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.10.4.11 green_leds_init_pwm()

```
void green_leds_init_pwm (
            void  ) [extern]
```

Initialize the green LEDs for use with variable intensity. The LED intensity can range from 0.255.

Warning

Fails with program exit when LEDs have already been to another mode.

Definition at line 58 of file leds.c.

Here is the call graph for this function:

4.10.4.12 leds_destroy()

```
void leds_destroy (
            void  ) [extern]
```

Unitialize the LEDs, such that the mode of the LEDs can be changed. Switch all lEDs off.

Definition at line 96 of file leds.c.

Here is the call graph for this function:

4.10.4.13 leds_init_onoff()

```
void leds_init_onoff (
            void  ) [extern]
```

Initialize the green LEDs for on/off use.

Warning

Fails with program exit when LEDs have already been to another mode.

Definition at line 37 of file leds.c.

Here is the call graph for this function:

4.11 Logging library

Macros

- #define pynq_info(...) pynq_log(LOG_LEVEL_INFO, LOG_DOMAIN, __FUNCTION__, __LINE__, __VA_↩
ARGS__)
- #define pynq_warning(...) pynq_log(LOG_LEVEL_WARNING, LOG_DOMAIN, __FUNCTION__, __LINE↩
__, __VA_ARGS__)
- #define pynq_error(...)

Typedefs

- typedef enum LogLevel LogLevel

Enumerations

- enum LogLevel { LOG_LEVEL_INFO , LOG_LEVEL_WARNING , LOG_LEVEL_ERROR , NUM_LOG_LEVELS
}

Functions

- void pynq_log (const LogLevel level, char const ∗domain, char const ∗location, unsigned int lineno, char const
∗fmt,...)

4.11.1 Detailed Description

Functions for error handling and logging.
```
#include <log.h>

int main (void)
{
   pynq_log("Print my information message");
   pynq_warning("Print my warning message");
   pynq_error("Failed on error");
   return EXIT_SUCCESS;
}
```

Or with a custom log domain
```
#include <log.h>

#undef LOG_DOMAIN
#define LOG_DOMAIN "MyApp"

int main ( int argc, char **argv)
{
   pynq_log("Print my information message");
   pynq_warning("Print my warning message");
   pynq_error("Failed on error");
   return EXIT_SUCCESS;
}
```

4.11.2 Macro Definition Documentation

4.11.2.1 pynq_error

```
#define pynq_error(
              ...  )
```
Value:
```
  do {                                                             \
     pynq_log(LOG_LEVEL_ERROR, LOG_DOMAIN, __FUNCTION__, __LINE__,  \
              __VA_ARGS__);                                         \
     for (;;)                                                       \
       ;                                                            \
  } while (0)
```

Parameters

...	

Wrapper around pynq_log to print error messages. This expects LOG_DOMAIN to be set.

Definition at line 118 of file log.h.

4.11.2.2 pynq_info

```
#define pynq_info(
            ... )   pynq_log(LOG_LEVEL_INFO, LOG_DOMAIN, __FUNCTION__, __LINE__, __VA_ARGS↩
__)
```

Parameters

...	

Wrapper around pynq_log to print info messages. This expects LOG_DOMAIN to be set.

Definition at line 100 of file log.h.

4.11.2.3 pynq_warning

```
#define pynq_warning(
            ... )   pynq_log(LOG_LEVEL_WARNING, LOG_DOMAIN, __FUNCTION__, __LINE__, __VA_↩
ARGS__)
```

Parameters

...	

Wrapper around pynq_log to print warning messages. This expects LOG_DOMAIN to be set.

Definition at line 109 of file log.h.

4.11.3 Typedef Documentation

4.11.3.1 LogLevel

```
typedef enum LogLevel LogLevel
```

4.11.4 Enumeration Type Documentation

4.11.4.1 LogLevel

```
enum LogLevel
```

Enumerator

LOG_LEVEL_INFO	Informational messages.
LOG_LEVEL_WARNING	Warning messages
LOG_LEVEL_ERROR	Error messages
NUM_LOG_LEVELS	Number of log levels

Definition at line 65 of file log.h.

4.11.5 Function Documentation

4.11.5.1 pynq_log()

```
void pynq_log (
            const LogLevel level,
            char const * domain,
            char const * location,
            unsigned int lineno,
            char const * fmt,
             ...  )
```

Parameters

level	The LogLevel of this mssage.
domain	The log domain.
fmt	The format string.
location	The location string of the message origin.
lineno	The line number of the message origin.
...	The arguments to the format string.

Print log messages with loglevel WARNING and higher. Messages of level ERROR will result in an abort().

Environment DEBUG will print out level LOG_LEVEL_INFO Environment FATAL_WARNING will abort after a warning.

Definition at line 52 of file log.c.

4.12 I/O pin mapping

Macros

- #define NUM_ANALOG_REFERENCE_PINS 14 /∗ # analog reference pins ∗/
- #define NUM_ANALOG_IN_PINS 6 /∗ # analog input pins ∗/
- #define IO_PMODA1 IO_RBPI07
- #define IO_PMODA2 IO_RBPI29
- #define IO_PMODA3 IO_RBPI27
- #define IO_PMODA4 IO_RBPI28
- #define IO_PMODA7 IO_RBPI31
- #define IO_PMODA8 IO_RBPI26
- #define PIN_CHECK(pin)

Enumerations

- enum io_t {
  IO_AR0 = 0 , IO_AR1 = 1 , IO_AR2 = 2 , IO_AR3 = 3 ,
  IO_AR4 = 4 , IO_AR5 = 5 , IO_AR6 = 6 , IO_AR7 = 7 ,
  IO_AR8 = 8 , IO_AR9 = 9 , IO_AR10 = 10 , IO_AR11 = 11 ,
  IO_AR12 = 12 , IO_AR13 = 13 , IO_A0 = 14 , IO_A1 = 15 ,
  IO_A2 = 16 , IO_A3 = 17 , IO_A4 = 18 , IO_A5 = 19 ,
  IO_SW0 = 20 , IO_SW1 = 21 , IO_BTN0 = 22 , IO_BTN1 = 23 ,
  IO_BTN2 = 24 , IO_BTN3 = 25 , IO_LD0 = 26 , IO_LD1 = 27 ,
  IO_LD2 = 28 , IO_LD3 = 29 , IO_AR_SCL = 31 , IO_AR_SDA = 30 ,
  IO_LD4B = 32 , IO_LD4R = 33 , IO_LD4G = 34 , IO_LD5B = 35 ,
  IO_LD5R = 36 , IO_LD5G = 37 , IO_RBPI40 = 38 , IO_RBPI37 = 39 ,
  IO_RBPI38 = 40 , IO_RBPI35 = 41 , IO_RBPI36 = 42 , IO_RBPI33 = 43 ,
  IO_RBPI18 = 44 , IO_RBPI32 = 45 , IO_RBPI10 = 46 , IO_RBPI27 = 47 ,
  IO_RBPI28 = 48 , IO_RBPI22 = 49 , IO_RBPI23 = 50 , IO_RBPI24 = 51 ,
  IO_RBPI21 = 52 , IO_RBPI26 = 53 , IO_RBPI19 = 54 , IO_RBPI31 = 55 ,
  IO_RBPI15 = 56 , IO_RBPI16 = 57 , IO_RBPI13 = 58 , IO_RBPI12 = 59 ,
  IO_RBPI29 = 60 , IO_RBPI08 = 61 , IO_RBPI07 = 62 , IO_RBPI05 = 63 ,
  IO_NUM_PINS = 64 }

Variables

- char ∗const pin_names [64]

4.12.1 Detailed Description

Definitions of I/O pin numbers and names for the switchbox and GPIO.

For example, when calling a function, use IO_AR0 to specify analog reference pin AR0. Specifically, symbolic pin names are prefixed with IO_ because they are used as inputs to switchbox functions, but the pin name when printed omits the IO_.

4.12.2 Macro Definition Documentation

4.12.2.1 IO_PMODA1

```
#define IO_PMODA1 IO_RBPI07
```

6 RaspberryPi headers are also accessible via Pmod A.

Definition at line 150 of file pinmap.h.

4.12.2.2 IO_PMODA2

```
#define IO_PMODA2 IO_RBPI29
```

Definition at line 151 of file pinmap.h.

4.12.2.3 IO_PMODA3

```
#define IO_PMODA3 IO_RBPI27
```

Definition at line 152 of file pinmap.h.

4.12.2.4 IO_PMODA4

```
#define IO_PMODA4 IO_RBPI28
```

Definition at line 153 of file pinmap.h.

4.12.2.5 IO_PMODA7

```
#define IO_PMODA7 IO_RBPI31
```

Definition at line 154 of file pinmap.h.

4.12.2.6 IO_PMODA8

```
#define IO_PMODA8 IO_RBPI26
```

Definition at line 155 of file pinmap.h.

4.12.2.7 NUM_ANALOG_IN_PINS

```
#define NUM_ANALOG_IN_PINS 6 /* # analog input pins */
```

Definition at line 43 of file pinmap.h.

4.12.2.8 NUM_ANALOG_REFERENCE_PINS

```
#define NUM_ANALOG_REFERENCE_PINS 14 /* # analog reference pins */
```

Definition of the number of I/O pins we have for each category.

Definition at line 42 of file pinmap.h.

4.12.2.9 PIN_CHECK

```
#define PIN_CHECK(
            pin )
```

Value:
```
  do {                                                              \
    if (pin >= IO_NUM_PINS) {                                       \
      pynq_error("pin %u is invalid, must be 0..%u-1.\n", pin, IO_NUM_PINS);  \
    }                                                               \
  } while (0);
```

macro that checks if the pin number is valid, throws an error if not.

Definition at line 160 of file pinmap.h.

4.12.3 Enumeration Type Documentation

4.12.3.1 io_t

```
enum io_t
```

Enumerator

IO_AR0	Analog reference pins (Arduino header).
IO_AR1	
IO_AR2	
IO_AR3	
IO_AR4	
IO_AR5	
IO_AR6	
IO_AR7	
IO_AR8	
IO_AR9	
IO_AR10	
IO_AR11	
IO_AR12	
IO_AR13	
IO_A0	Analog input pins (Arduino header).
IO_A1	
IO_A2	
IO_A3	
IO_A4	
IO_A5	
IO_SW0	Switch input pins.
IO_SW1	
IO_BTN0	Button input pins.
IO_BTN1	
IO_BTN2	
IO_BTN3	
IO_LD0	LED output pins.
IO_LD1	
IO_LD2	
IO_LD3	
IO_AR_SCL	I2C pins.
IO_AR_SDA	
IO_LD4B	The RGB adresses for IO_LD4 and IO_LD5.
IO_LD4R	
IO_LD4G	
IO_LD5B	
IO_LD5R	
IO_LD5G	
IO_RBPI40	The RaspberryPi header-pin indexing.
IO_RBPI37	
IO_RBPI38	
IO_RBPI35	
IO_RBPI36	
IO_RBPI33	
IO_RBPI18	
IO_RBPI32	
IO_RBPI10	
IO_RBPI27	
IO_RBPI28	

Enumerator

IO_RBPI22	
IO_RBPI23	
IO_RBPI24	
IO_RBPI21	
IO_RBPI26	
IO_RBPI19	
IO_RBPI31	
IO_RBPI15	
IO_RBPI16	
IO_RBPI13	
IO_RBPI12	
IO_RBPI29	
IO_RBPI08	
IO_RBPI07	
IO_RBPI05	
IO_NUM_PINS	

Definition at line 45 of file pinmap.h.

4.12.4 Variable Documentation

4.12.4.1 pin_names

```
char* const pin_names[64]  [extern]
```

Pin names.

Definition at line 24 of file pinmap.c.

4.13 PWM library

Enumerations

- enum pwm_index_t {
  PWM0 , PWM1 , PWM2 , PWM3 ,
  PWM4 , PWM5 , NUM_PWMS }

Functions

- bool pwm_initialized (const int pwm)
- void pwm_init (const int pwm, const uint32_t period)
- void pwm_destroy (const int pwm)
- void pwm_set_duty_cycle (const int pwm, const uint32_t duty)
- void pwm_set_period (const int pwm, const uint32_t period)
- uint32_t pwm_get_period (const int pwm)
- uint32_t pwm_get_duty_cycle (const int pwm)
- void pwm_set_steps (const int pwm, const uint32_t steps)
- uint32_t pwm_get_steps (const int pwm)

4.13.1 Detailed Description

Functions to use Pulse Width Modulation (PWM).

Each of the 6 PWM channels (numbered 0..NUM_PWMS-1) can be linked to any mappable pin (e.g. green or color LEDs, buttons).

PWM can also be used through GPIO (see gpio.h and pinmap.h). Note that GPIO numbering (SWB_PWM0..SWB↩
_PWM5) is then used instead of 0..NUM_PWMS-1 (PWM0..PWM5).

4.13.2 Enumeration Type Documentation

4.13.2.1 pwm_index_t

enum pwm_index_t

Enum of PWM channels.

All functions use a PWM channel from 0..NUM_PWMS-1. Alternatively, you can use PWMi instead of just i if you find that clearer.

Enumerator

PWM0	
PWM1	
PWM2	
PWM3	
PWM4	
PWM5	
NUM_PWMS	

Definition at line 47 of file pwm.h.

4.13.3 Function Documentation

4.13.3.1 pwm_destroy()

```
void pwm_destroy (
            const int pwm )  [extern]
```

Removes the instantiated shared memory system of the PWM channel.

Parameters

pwm	The PWM channel to destroy.

Warning

> Fails with program exit if pwm is outside valid range.

Definition at line 72 of file pwm.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.13.3.2 pwm_get_duty_cycle()

```
uint32_t pwm_get_duty_cycle (
              const int pwm ) [extern]
```

Gets the duty cycle of the specified PWM channel.

Parameters

pwm	The PWM channel.

Returns

The duty cycle of the specified PWM channel.

Warning

Fails with program exit if pwm is outside valid range or if pwm has not been initialized.

Definition at line 78 of file pwm.c.

Here is the call graph for this function:

4.13.3.3 pwm_get_period()

```
uint32_t pwm_get_period (
              const int pwm )
```

Returns the period of a certain PWM channel.

Parameters

pwm	The PWM channel.

Returns

The period of the specified PWM channel as an uint32_t.

Warning

Fails with program exit if pwm is outside valid range or if pwm has not been initialized.

Definition at line 83 of file pwm.c.

Here is the call graph for this function:

4.13.3.4 pwm_get_steps()

```
uint32_t pwm_get_steps (
            const int pwm )  [extern]
```

Get the number of steps a certain channel has taken so far.

Parameters

pwm	PWM channel.

Returns

The number of steps that have been taken; 0 is off and -1 is continous.

Warning

Fails with program exit if pwm is outside valid range or if pwm has not been initialized.

Definition at line 98 of file pwm.c.

Here is the call graph for this function:

4.13.3.5 pwm_init()

```
void pwm_init (
            const int pwm,
            const uint32_t period )  [extern]
```

Initializes the PWM channel with the specified period.

Parameters

pwm	the PWM channel to initialize.
period	The period to set for the PWM channel.

Warning

Fails with program exit if pwm is outside valid range.

Definition at line 61 of file pwm.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.13.3.6 pwm_initialized()

```
bool pwm_initialized (
            const int pwm )  [extern]
```

Checks if the channel index is initialized.

Parameters

pwm	The PWM channel

Returns

True if initialized, false if not

Warning

Fails with program exit if pwm is outside valid range.

Definition at line 38 of file pwm.c.

4.13.3.7 pwm_set_duty_cycle()

```
void pwm_set_duty_cycle (
            const int pwm,
            const uint32_t duty )  [extern]
```

Sets the duty cycle for the specified PWM channel.

Parameters

pwm	The PWM channel.
duty	The duty cycle to set for the PWM channel.

Warning

Fails with program exit if pwm is outside valid range or if pwm has not been initialized.

Definition at line 93 of file pwm.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.13.3.8 pwm_set_period()

```
void pwm_set_period (
            const int pwm,
            const uint32_t period )  [extern]
```

Sets the period for the specified PWM channel.

Parameters

pwm	The PWM channel.
period	The period to set for the PWM channel.

Warning

> Fails with program exit if pwm is outside valid range or if pwm has not been initialized.

Definition at line 88 of file pwm.c.

Here is the call graph for this function:

4.13.3.9   pwm_set_steps()

```
void pwm_set_steps (
            const int pwm,
            const uint32_t steps ) [extern]
```

Generates steps steps on the PWM channel.

Parameters

pwm	The PWM channel.
steps	The number of steps to cycle, 0 to turn off and -1 to run continously.

Warning

> Fails with program exit if pwm is outside valid range or if pwm has not been initialized.

Definition at line 103 of file pwm.c.

Here is the call graph for this function:

4.14   I/O Switchbox library

Macros

- #define NUM_SWITCHBOX_NAMES 40

Typedefs

- typedef enum io_configuration io_configuration_t

Enumerations

- enum io_configuration {
  SWB_GPIO = 0x00 , SWB_Interrupt_In = 0x01 , SWB_UART0_TX = 0x02 , SWB_UART0_RX = 0x03 ,
  SWB_SPI0_CLK = 0x04 , SWB_SPI0_MISO = 0x05 , SWB_SPI0_MOSI = 0x06 , SWB_SPI0_SS = 0x07 ,
  SWB_SPI1_CLK = 0x08 , SWB_SPI1_MISO = 0x09 , SWB_SPI1_MOSI = 0x0A , SWB_SPI1_SS = 0x0B ,
  SWB_IIC0_SDA = 0x0C , SWB_IIC0_SCL = 0x0D , SWB_IIC1_SDA = 0x0E , SWB_IIC1_SCL = 0x0F ,
  SWB_PWM0 = 0x10 , SWB_PWM1 = 0x11 , SWB_PWM2 = 0x12 , SWB_PWM3 = 0x13 ,
  SWB_PWM4 = 0x14 , SWB_PWM5 = 0x15 , SWB_TIMER_G0 = 0x18 , SWB_TIMER_G1 = 0x19 ,
  SWB_TIMER_G2 = 0x1A , SWB_TIMER_G3 = 0x1B , SWB_TIMER_G4 = 0x1C , SWB_TIMER_G5 = 0x1D
  ,
  SWB_TIMER_G6 = 0x1E , SWB_TIMER_G7 = 0x1F , SWB_UART1_TX = 0x22 , SWB_UART1_RX = 0x23 ,
  SWB_TIMER_IC0 = 0x38 , SWB_TIMER_IC1 = 0x39 , SWB_TIMER_IC2 = 0x3A , SWB_TIMER_IC3 = 0x3B
  ,
  SWB_TIMER_IC4 = 0x3C , SWB_TIMER_IC5 = 0x3D , SWB_TIMER_IC6 = 0x3E , SWB_TIMER_IC7 = 0x3F
  ,
  NUM_IO_CONFIGURATIONS }

Functions

- void switchbox_init (void)
- void switchbox_set_pin (const io_t pin_number, const io_configuration_t pin_type)
- void switchbox_reset (void)
- void switchbox_destroy (void)
- io_configuration_t switchbox_get_pin (const io_t pin_number)

Variables

- char ∗const switchbox_names [NUM_SWITCHBOX_NAMES]

4.14.1 Detailed Description

The switchbox enables run-time (re)mapping of I/O pins.

For example, the transmit output of UART 0 (SWB_UART0_TX) can be mapped to analog pins IO_AR0 & IO_AR1. Or the output of PWM 0 (SWB_PWM0) can be mapped to green LED 0 (pin IO_LD0). Or the output of PWM 0 (pin SWB_PWM0) can be mapped to the green component of color LED 0 (pin IOB_LD0).

Warning

> Switchbox functions (dis)connect IO pins (outside world) to FPGA hardware (on the Zynq 7020). IO pins are named IO_∗ (e.g. IO_LD0) and are of type io_t defined in pinmap.h. The FPGA hardware is named SWB_∗ (e.g. SWB_UART0) of type (io_configuration_t) defined in switchbox.h.

```
#include<pinmap.h>
#include<switchbox.h>

int main (void)
{
  pynq_init();
  switchbox_init();
  // connect pin A0 to UART0's TX pin
  switchbox_set_pin(IO_AR0, SWB_UART0_TX);
  // also see examples in gpio.h
  switchbox_destroy();
  pynq_destroy();
}
```

4.14.2 Macro Definition Documentation

4.14.2.1 NUM_SWITCHBOX_NAMES

```
#define NUM_SWITCHBOX_NAMES 40
```

Definition at line 135 of file switchbox.h.

4.14.3 Typedef Documentation

4.14.3.1 io_configuration_t

```
typedef enum io_configuration io_configuration_t
```

4.14.4 Enumeration Type Documentation

4.14.4.1 io_configuration

```
enum io_configuration
```

Enumerator

SWB_GPIO	Map pin to GPIO
SWB_Interrupt_In	Map pin to internal interrupt (UNUSED)
SWB_UART0_TX	Map pin to TX channel of UART 0
SWB_UART0_RX	Map pin to RX channel of UART 0
SWB_SPI0_CLK	Map pin to clock channel of SPI 0
SWB_SPI0_MISO	Map pin to miso channel of SPI 0
SWB_SPI0_MOSI	Map pin to mosi channel of SPI 0
SWB_SPI0_SS	Map pin to ss channel of SPI 0
SWB_SPI1_CLK	Map pin to clock channel of SPI 1
SWB_SPI1_MISO	Map pin to miso channel of SPI 1
SWB_SPI1_MOSI	Map pin to mosi channel of SPI 1
SWB_SPI1_SS	Map pin to ss channel of SPI 1
SWB_IIC0_SDA	Map pin to sda channel of IIC 0
SWB_IIC0_SCL	Map pin to scl channel of IIC 0
SWB_IIC1_SDA	Map pin to sda channel of IIC 1
SWB_IIC1_SCL	Map pin to scl channel of IIC 1
SWB_PWM0	Map pin to output channel of PWM 0
SWB_PWM1	Map pin to output channel of PWM 1
SWB_PWM2	not connected
SWB_PWM3	not connected
SWB_PWM4	not connected
SWB_PWM5	not connected
SWB_TIMER_G0	
SWB_TIMER_G1	
SWB_TIMER_G2	not connected
SWB_TIMER_G3	not connected
SWB_TIMER_G4	not connected
SWB_TIMER_G5	not connected
SWB_TIMER_G6	not connected
SWB_TIMER_G7	not connected
SWB_UART1_TX	
SWB_UART1_RX	
SWB_TIMER_IC0	
SWB_TIMER_IC1	
SWB_TIMER_IC2	
SWB_TIMER_IC3	
SWB_TIMER_IC4	
SWB_TIMER_IC5	
SWB_TIMER_IC6	
SWB_TIMER_IC7	
NUM_IO_CONFIGURATIONS	number elements in this enum

Definition at line 62 of file switchbox.h.

4.14.5 Function Documentation

4.14.5.1 switchbox_destroy()

```
void switchbox_destroy (
            void ) [extern]
```

Resets all pins of the switch box to be input.

Definition at line 112 of file switchbox.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.14.5.2 switchbox_get_pin()

```
io_configuration_t switchbox_get_pin (
            const io_t pin_number ) [extern]
```

Sets the mode of a specified pin.

Parameters

| *pin_number* | The IO pin number. |

Returns

      The FPGA hardware the IO pin is connected to.

Definition at line 162 of file switchbox.c.

Here is the caller graph for this function:

4.14.5.3 switchbox_init()

```
void switchbox_init (
            void ) [extern]
```

Initializes the switch box.

Initializes the shared memory and sets the io switch base address

Definition at line 105 of file switchbox.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.14.5.4 switchbox_reset()

```
void switchbox_reset (
            void ) [extern]
```

Resets all pins of the switch box to be input.

Definition at line 118 of file switchbox.c.

Here is the caller graph for this function:

4.14.5.5 switchbox_set_pin()

```
void switchbox_set_pin (
            const io_t pin_number,
            const io_configuration_t pin_type )  [extern]
```

Set the type of a switch pin.

Parameters

pin_number	The number of the IO pin to connect (IO_∗, IO_LD0).
pin_type	The FPGA hardware to connect to (SWB_∗, e.g. SWB_PWM0).

Definition at line 126 of file switchbox.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.14.6 Variable Documentation

4.14.6.1 switchbox_names

```
char* const switchbox_names[NUM_SWITCHBOX_NAMES]  [extern]
```

Taken from scpi_names.h, lookup table for channels in the mapping_info function.

Definition at line 25 of file switchbox.c.

4.15 UART library

Enumerations

- enum uart_index_t { UART0 = 0 , UART1 = 1 , NUM_UARTS }

Functions

- void uart_init (const int uart)
- void uart_destroy (const int uart)
- void uart_send (const int uart, const uint8_t data)
- uint8_t uart_recv (const int uart)
- bool uart_has_data (const int uart)
- bool uart_has_space (const int uart)
- void uart_reset_fifos (const int uart)

4.15.1 Detailed Description

Functions to use the Universal Asynchronous Receiver-Transmitter (UART).

Two UART channels can be instantiated, UART0 and UART1. Before sending and receiving bytes the UART must be connect to some I/O pins through the switchbox, e.g.

```
switchbox_set_pin(IO_AR0, SWB_UART0_RX);
switchbox_set_pin(IO_AR1, SWB_UART0_TX);
```

After that, an example of how to use this library for the MASTER.

```
#include <libpynq.h>
int main (void)
{
 // initialise all I/O
 pynq_init();

 // initialize UART 0
 uart_init(UART0);
 // flush FIFOs of UART 0
 uart_reset_fifos(UART0);

 uint8_t byte[] = "Hello\n";
 int i = 0;
 while (byte[i] != '\0') {
  uart_send (UART0, byte[i]);
  printf("sent byte %d\n", byte[i]);
  i++;
 }

  // clean up after use
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

An example of how to use this library for the SLAVE.

```
#include <libpynq.h>
int main (void)
{
 // initialise all I/O
 pynq_init();

 // initialize UART channel 0
 uart_init(UART0);
 // flush FIFOs of UART 0
 uart_reset_fifos (UART0);

 printf("listening\n");
 do {
   // get a byte from UART 0
   uint8_t msg = uart_recv(UART0);
   printf("received byte %d\n", msg);
 } while (1);

  // clean up after use
  pynq_destroy();
  return EXIT_SUCCESS;
}
```

UARTs can be routed through the switch box (see switchbox.h). Note that switchbox numbering (SWB_↩ UART0..SWB_UART1) is then used instead of 0..NUM_UARTS-1 (UART0..UART1).

4.15.2 Enumeration Type Documentation

4.15.2.1 uart_index_t

```
enum uart_index_t
```

Enum of UARTs. Functions use a switch numbered from 0..NUM_UARTS-1. Alternatively, you can use UARTi instead of just i if you find that clearer.

Enumerator

UART0	
UART1	
NUM_UARTS	

Definition at line 107 of file uart.h.

4.15.3 Function Documentation

4.15.3.1 uart_destroy()

```
void uart_destroy (
             const int uart ) [extern]
```

Close the shared memory handle for the specified UART index.

Parameters

uart	The UART index to remove from the shared memory space.

Warning

Fails with program exit if the UART channel is outside valid range.

Definition at line 61 of file uart.c.

Here is the call graph for this function:

4.15.3.2 uart_has_data()

```
bool uart_has_data (
             const int uart ) [extern]
```

Check if the receive FIFO for the specified UART index has data available.

Parameters

uart	The UART index used to check for data.

Returns

True if the receive FIFO has data, false otherwise.

Warning

Fails with program exit if the UART channel is outside valid range.

Definition at line 98 of file uart.c.

4.15.3.3 uart_has_space()

```
bool uart_has_space (
            const int uart ) [extern]
```

Check if the transmit FIFO for the specified UART index has space available.

Parameters

uart	The UART index to check for space.

Returns

True if the FIFO has space, false otherwise.

Warning

Fails with program exit if the UART channel is outside valid range.

Definition at line 110 of file uart.c.

4.15.3.4 uart_init()

```
void uart_init (
            const int uart ) [extern]
```

Initialize the UART specified by the index with a shared memory handle and a buffer size of 4096 bytes.

Parameters

uart	The UART index to initialize.

Warning

Fails with program exit if the UART channel is outside valid range or when the shared memory system has not been instantiated.

Definition at line 48 of file uart.c.

Here is the call graph for this function:

4.15.3.5 uart_recv()

```
uint8_t uart_recv (
            const int uart ) [extern]
```

Receive a byte of data from the specified UART index by waiting for the receive FIFO to have data and then reading the data from the receive buffer.

Parameters

uart	The UART index to receive data from.

Returns

> The received data byte.

Warning

> Fails with program exit if the UART channel is outside valid range.

Definition at line 85 of file uart.c.

4.15.3.6   uart_reset_fifos()

```
void uart_reset_fifos (
            const int uart )  [extern]
```

This function resets both the transmit and receive FIFOs of the UART specified by the `uart` parameter. This can be useful when there is data stuck in the FIFOs or when the FIFOs are not behaving as expected.

Parameters

uart	The UART index of the UART whose FIFOs should be reset.

Warning

> This function is specific to UARTs that have FIFOs, and will have no effect on UARTs that do not have FIFOs.
>
> Resetting the FIFOs will result in the loss of any data that is currently in the FIFOs. Therefore, this function should be used with caution, and only when it is absolutely necessary to do so.
>
> Fails with program exit if the UART channel is outside valid range.

Definition at line 121 of file uart.c.

4.15.3.7   uart_send()

```
void uart_send (
            const int uart,
            const uint8_t data )  [extern]
```

Send a byte of data on the specified UART index by waiting for the transmit FIFO to have space and then writing the data to the transmit buffer.

Parameters

uart	The UART index to send data to.
data	The data to send to the UART index.

Warning

Fails with program exit if the UART channel is outside valid range.

Definition at line 72 of file uart.c.

4.16 Utility library

Functions

- void sleep_msec (int msec)
- void mapping_info (void)

4.16.1 Detailed Description

Some simple helper functions.

4.16.2 Function Documentation

4.16.2.1 mapping_info()

```
void mapping_info (
            void ) [extern]
```

Displays a table to see where all pins have been mapped, what channels have been linked where and the i/o of each mappable pin.

Definition at line 37 of file util.c.

Here is the call graph for this function:

4.16.2.2 sleep_msec()

```
void sleep_msec (
            int msec ) [extern]
```

Wait for msec milliseconds.

Parameters

| *ms* | The amount of milliseconds the PYNQ should stay idle |

Definition at line 32 of file util.c.

Here is the caller graph for this function:

4.17 Versioning library

Data Structures

- struct version_t

Functions

- void print_version (void)
- void check_version (void)

Variables

- const version_t libpynq_version

4.17.1 Detailed Description

Typedef and functions to check the version and compatibility of the libpynq library and the FPGA bitstream.

Semantic versioning (https://semver.org) is used. Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes between libpynq and FPGA bitstream (SD-card image)

- MINOR version when you add functionality in a backward compatible manner.

- PATCH version when you make backward compatible bug fixes.

When the libpynq library version and the FPGA bitstream version are not the same:

- libpynq.MAJOR $<$ bitstream.MAJOR: you MUST update libpynq to the latest version compatible with the bitstream version. The check_version function will fail and exit your program.

- libpynq.MAJOR $>$ bitstream.MAJOR: you MUST update the bitstream to the latest version compatible with the libpynq version (or downgrade the libpynq version to bitstream.MAJOR). The print/check_version function will fail and exit your program.

- libpynq.MINOR $>$ bitstream.MINOR: it is recommended to update the bitstream to the latest version compatible with the libpynq version. The print_version function will print an INFO message.

- libpynq.MINOR $<$ bitstream.MINOR: it is recommended to update the libpynq to the latest version compatible with the bitstream version. The print_version function will print an INFO message.

- libpynq.PATCH != bitstream.PATCH: no action required

4.17.2   Function Documentation

4.17.2.1   check_version()

```
void check_version (
            void  )  [extern]
```

Check the version of the hardware (bitstream) and the libpynq library. Called by e.g. the switchbox but can also be called in user code.

Warning

Fails with program exit when versions are incompatible.

Definition at line 68 of file version.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.17.2.2   print_version()

```
void print_version (
            void  )  [extern]
```

Print the version of the hardware (bitstream) and the libpynq library.

Prints INFO message when minor/patch versions are different.

Definition at line 44 of file version.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.17.3   Variable Documentation

4.17.3.1   libpynq_version

```
const version_t libpynq_version  [extern]
```

Constant containing the version of this the libpynq library.

Definition at line 34 of file version.c.

Chapter 5

Data Structure Documentation

5.1 arm_shared_t Struct Reference

```
#include <arm_shared_memory_system.h>
```

Data Fields

- int file_descriptor
- uint32_t address
- uint32_t length
- void ∗ mmaped_region

5.1.1 Detailed Description

Definition at line 39 of file arm_shared_memory_system.h.

5.1.2 Field Documentation

5.1.2.1 address

```
uint32_t arm_shared_t::address
```

Definition at line 41 of file arm_shared_memory_system.h.

5.1.2.2 file_descriptor

```
int arm_shared_t::file_descriptor
```

Definition at line 40 of file arm_shared_memory_system.h.

5.1.2.3 length

```
uint32_t arm_shared_t::length
```

Definition at line 42 of file arm_shared_memory_system.h.

5.1.2.4 mmaped_region

```
void* arm_shared_t::mmaped_region
```

Definition at line 43 of file arm_shared_memory_system.h.

The documentation for this struct was generated from the following file:

- library/arm_shared_memory_system.h

5.2 display_t Struct Reference

```
#include <display.h>
```

Data Fields

- uint16_t _width
- uint16_t _height
- uint16_t _offsetx
- uint16_t _offsety
- uint16_t _font_direction
- uint16_t _font_fill
- uint16_t _font_fill_color
- uint16_t _font_underline
- uint16_t _font_underline_color
- int16_t _dc
- int16_t _bl

5.2.1 Detailed Description

Internal type, do not use. Type of display that stores parameters for usage in different functions.

Definition at line 116 of file display.h.

5.2.2 Field Documentation

5.2.2.1 _bl

```
int16_t display_t::_bl
```

Definition at line 127 of file display.h.

5.2.2.2 _dc

```
int16_t display_t::_dc
```

Definition at line 126 of file display.h.

5.2.2.3 _font_direction

```
uint16_t display_t::_font_direction
```

Definition at line 121 of file display.h.

5.2.2.4 _font_fill

```
uint16_t display_t::_font_fill
```

Definition at line 122 of file display.h.

5.2.2.5 _font_fill_color

```
uint16_t display_t::_font_fill_color
```

Definition at line 123 of file display.h.

5.2.2.6 _font_underline

```
uint16_t display_t::_font_underline
```

Definition at line 124 of file display.h.

5.2.2.7 _font_underline_color

```
uint16_t display_t::_font_underline_color
```

Definition at line 125 of file display.h.

5.2.2.8 _height

```
uint16_t display_t::_height
```

Definition at line 118 of file display.h.

5.2.2.9 _offsetx

```
uint16_t display_t::_offsetx
```

Definition at line 119 of file display.h.

5.2.2.10 _offsety

`uint16_t display_t::_offsety`

Definition at line 120 of file display.h.

5.2.2.11 _width

`uint16_t display_t::_width`

Definition at line 117 of file display.h.

The documentation for this struct was generated from the following file:

- library/display.h

5.3 FontxFile Struct Reference

`#include <fontx.h>`

Data Fields

- const char ∗ path
- char fxname [10]
- bool opened
- bool valid
- bool is_ank
- uint8_t w
- uint8_t h
- uint16_t fsz
- uint8_t bc
- FILE ∗ file

5.3.1 Detailed Description

Struct representing a font file.

Definition at line 28 of file fontx.h.

5.3.2 Field Documentation

5.3.2.1 bc

`uint8_t FontxFile::bc`

Background color of the font file.

Definition at line 38 of file fontx.h.

5.3.2.2 file

`FILE* FontxFile::file`

Pointer to the font file stream.

Definition at line 39 of file fontx.h.

5.3.2.3 fsz

`uint16_t FontxFile::fsz`

Size of the font file in bytes.

Definition at line 37 of file fontx.h.

5.3.2.4 fxname

`char FontxFile::fxname[10]`

Name of the font file.

Definition at line 30 of file fontx.h.

5.3.2.5 h

`uint8_t FontxFile::h`

Height of each character in the font file.

Definition at line 36 of file fontx.h.

5.3.2.6 is_ank

`bool FontxFile::is_ank`

Flag indicating whether the font file contains only ASCII characters.

Definition at line 33 of file fontx.h.

5.3.2.7 opened

`bool FontxFile::opened`

Flag indicating whether the font file is open.

Definition at line 31 of file fontx.h.

5.3.2.8 path

`const char* FontxFile::path`

Path to the font file.

Definition at line 29 of file fontx.h.

5.3.2.9 valid

`bool FontxFile::valid`

Flag indicating whether the font file is valid.

Definition at line 32 of file fontx.h.

5.3.2.10 w

`uint8_t FontxFile::w`

Width of each character in the font file.

Definition at line 35 of file fontx.h.

The documentation for this struct was generated from the following file:

- library/fontx.h

5.4 IICHandle Struct Reference

Collaboration diagram for IICHandle:

Data Fields

- arm_shared mem_handle
- volatile uint32_t ∗ ptr
- uint32_t ∗ register_map
- uint32_t register_map_length
- uint8_t saddr
- uint32_t selected_register
- uint32_t new_val
- uint32_t recv_cnt
- IICState state
- int addressed

5.4.1 Detailed Description

Definition at line 42 of file iic.c.

5.4.2 Field Documentation

5.4.2.1 addressed

`int IICHandle::addressed`

Definition at line 55 of file iic.c.

5.4.2.2 mem_handle

`arm_shared IICHandle::mem_handle`

Definition at line 43 of file iic.c.

5.4.2.3 new_val

`uint32_t IICHandle::new_val`

Definition at line 52 of file iic.c.

5.4.2.4 ptr

`volatile uint32_t* IICHandle::ptr`

Definition at line 44 of file iic.c.

5.4.2.5 recv_cnt

`uint32_t IICHandle::recv_cnt`

Definition at line 53 of file iic.c.

5.4.2.6 register_map

`uint32_t* IICHandle::register_map`

Definition at line 47 of file iic.c.

5.4.2.7 register_map_length

`uint32_t IICHandle::register_map_length`

Definition at line 48 of file iic.c.

5.4.2.8 saddr

`uint8_t IICHandle::saddr`

Definition at line 50 of file iic.c.

5.4.2.9 selected_register

`uint32_t IICHandle::selected_register`

Definition at line 51 of file iic.c.

5.4.2.10 state

`IICState IICHandle::state`

Definition at line 54 of file iic.c.

The documentation for this struct was generated from the following file:

- library/iic.c

5.5 pin Struct Reference

Data Fields

- char ∗ name
- char ∗ state
- io_configuration_t channel

5.5.1 Detailed Description

Definition at line 99 of file switchbox.c.

5.5.2 Field Documentation

5.5.2.1 channel

`io_configuration_t pin::channel`

Definition at line 102 of file switchbox.c.

5.5.2.2 name

```
char* pin::name
```

Definition at line 100 of file switchbox.c.

5.5.2.3 state

```
char* pin::state
```

Definition at line 101 of file switchbox.c.

The documentation for this struct was generated from the following file:

- library/switchbox.c

5.6 pin_state_t Struct Reference

Data Fields

- char ∗ name
- gpio_direction_t state
- uint8_t channel
- char ∗ level

5.6.1 Detailed Description

Definition at line 25 of file util.c.

5.6.2 Field Documentation

5.6.2.1 channel

```
uint8_t pin_state_t::channel
```

Definition at line 28 of file util.c.

5.6.2.2 level

```
char* pin_state_t::level
```

Definition at line 29 of file util.c.

5.6.2.3 name

`char` * pin_state_t::name

Definition at line 26 of file util.c.

5.6.2.4 state

`gpio_direction_t` pin_state_t::state

Definition at line 27 of file util.c.

The documentation for this struct was generated from the following file:

- library/util.c

5.7 version_t Struct Reference

`#include <version.h>`

Data Fields

- uint8_t release [64]
- uint32_t major
- uint32_t minor
- uint32_t patch

5.7.1 Detailed Description

Typedef of version.

Definition at line 63 of file version.h.

5.7.2 Field Documentation

5.7.2.1 major

`uint32_t version_t::major`

Definition at line 65 of file version.h.

5.7.2.2 minor

`uint32_t version_t::minor`

Definition at line 66 of file version.h.

5.7.2.3 patch

`uint32_t version_t::patch`

Definition at line 67 of file version.h.

5.7.2.4 release

`uint8_t version_t::release[64]`

Definition at line 64 of file version.h.

The documentation for this struct was generated from the following file:

- library/version.h

Chapter 6

File Documentation

6.1 library/adc.c File Reference

```
#include <adc.h>
#include <arm_shared_memory_system.h>
#include <errno.h>
#include <log.h>
#include <platform.h>
#include <stdio.h>
#include <stdlib.h>
```
Include dependency graph for adc.c:

6.2 adc.c

```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <adc.h>
00023 #include <arm_shared_memory_system.h>
00024 #include <errno.h>
00025 #include <log.h>
00026 #include <platform.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029
00030 static struct arm_shared_t adc_handle;
00031 static volatile uint32_t *adc = NULL;
00032
00033 static const uint32_t twopow16 = 0b10000000000000000;
00034
```

```
00035 bool invalid_channel_adc(const adc_channel_t channel) {
00036   if (channel == ADC0) {
00037     return false;
00038   }
00039   if (channel == ADC1) {
00040     return false;
00041   }
00042   if (channel == ADC2) {
00043     return false;
00044   }
00045   if (channel == ADC3) {
00046     return false;
00047   }
00048   if (channel == ADC4) {
00049     return false;
00050   }
00051   if (channel == ADC5) {
00052     return false;
00053   }
00054   return true;
00055 }
00056
00057 bool initialized_adc(void) {
00058   if (adc == NULL) {
00059     return false;
00060   }
00061   return true;
00062 }
00063
00064 bool check_initialized_adc(void) {
00065   if (!initialized_adc()) {
00066     pynq_error("The ADC has not been initialized\n");
00067   }
00068   return true;
00069 }
00070
00071 bool check_channel_adc(const adc_channel_t channel) {
00072   if (invalid_channel_adc(channel)) {
00073     pynq_error("Invalid ADC channel %d\n", channel);
00074   }
00075   return true;
00076 }
00077
00078 void adc_init(void) { adc = arm_shared_init(&adc_handle, xadc_wiz_0, 4096); }
00079
00080 void adc_destroy(void) {
00081   if (adc != NULL) {
00082     (void)arm_shared_close(&adc_handle);
00083     adc = NULL;
00084   }
00085 }
00086
00087 double adc_read_channel(const adc_channel_t channel) {
00088   (void)check_channel_adc(channel);
00089   (void)check_initialized_adc();
00090
00091   // TODO we need to calibrate this
00092   double value = adc[channel] * (3.23 / twopow16);
00093
00094   return value;
00095 }
00096
00097 uint32_t adc_read_channel_raw(adc_channel_t channel) {
00098   (void)check_channel_adc(channel);
00099   (void)check_initialized_adc();
00100
00101   if (adc == NULL) {
00102     return UINT32_MAX;
00103   }
00104   uint32_t value = adc[channel];
00105
00106   return value;
00107 }
```

6.3 library/adc.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
```
Include dependency graph for adc.h: This graph shows which files directly or indirectly include this file:

Enumerations

- enum adc_channel_t {
  ADC0 = ((0x240 / 4) + 1) , ADC1 = ((0x240 / 4) + 9) , ADC2 = ((0x240 / 4) + 6) , ADC3 = ((0x240 / 4) + 15) ,
  ADC4 = ((0x240 / 4) + 5) , ADC5 = ((0x240 / 4) + 13) }

Functions

- bool initialized_adc (void)
- void adc_init (void)
- void adc_destroy (void)
- double adc_read_channel (adc_channel_t channel)
- uint32_t adc_read_channel_raw (adc_channel_t channel)

6.4 adc.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef ADC_H
00023 #define ADC_H
00024
00025 #include <stdbool.h>
00026 #include <stdint.h>
00027
00043 typedef enum {
00045   ADC0 = ((0x240 / 4) + 1),
00047   ADC1 = ((0x240 / 4) + 9),
00049   ADC2 = ((0x240 / 4) + 6),
00051   ADC3 = ((0x240 / 4) + 15),
00053   ADC4 = ((0x240 / 4) + 5),
00055   ADC5 = ((0x240 / 4) + 13),
00056 } adc_channel_t;
00057
00062 extern bool initialized_adc(void);
00063
00067 extern void adc_init(void);
00068
00073 extern void adc_destroy(void);
00074
00082 extern double adc_read_channel(adc_channel_t channel);
00083
00090 extern uint32_t adc_read_channel_raw(adc_channel_t channel);
00091
00096 #endif // ADC_H
```

6.5 library/arm_shared_memory_system.c File Reference

```
#include <arm_shared_memory_system.h>
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <unistd.h>
```
Include dependency graph for arm_shared_memory_system.c:

Functions

- void ∗ arm_shared_init (arm_shared ∗handle, const uint32_t address, const uint32_t length)
- void arm_shared_close (arm_shared ∗handle)

6.6 arm_shared_memory_system.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <arm_shared_memory_system.h>
00023 #include <errno.h>
00024 #include <fcntl.h>
00025 #include <stdio.h>
00026 #include <stdlib.h>
00027 #include <string.h>
00028 #include <sys/mman.h>
00029 #include <sys/types.h>
00030 #include <unistd.h>
00031
00032 void *arm_shared_init(arm_shared *handle, const uint32_t address,
00033                       const uint32_t length) {
00034   if (handle == NULL) {
00035     fprintf(stderr, "You need to pass a valid handle to %s\n", __FUNCTION__);
00036     exit(EXIT_FAILURE);
00037   }
00038
00039   handle->address = address;
00040   handle->length = length;
00041   handle->file_descriptor = open("/dev/mem", O_RDWR | O_SYNC);
00042   if (handle->file_descriptor < 0) {
00043     fprintf(stderr,
00044             "FAILED open memory: %s, please run with sufficient permissions "
00045             "(sudo).\n",
00046             strerror(errno));
00047     exit(EXIT_FAILURE);
00048   }
00049
```

```
00050   long page_size = sysconf(_SC_PAGE_SIZE);
00051
00052   uint32_t start_address = handle->address;
00053   uint32_t page_offset = start_address % page_size;
00054   start_address -= page_offset;
00055   handle->length += page_offset;
00056
00057   handle->mmaped_region =
00058       mmap(NULL, handle->length, PROT_READ | PROT_WRITE, MAP_SHARED,
00059           handle->file_descriptor, start_address);
00060
00061   if (handle->mmaped_region == MAP_FAILED) {
00062     fprintf(stderr, "FAILED to memory map requested region: %s\n",
00063           strerror(errno));
00064     close(handle->file_descriptor);
00065     exit(EXIT_FAILURE);
00066   }
00067   return (void *)(((uint32_t)(handle->mmaped_region)) + page_offset);
00068 }
00069
00070 void arm_shared_close(arm_shared *handle) {
00071   if (handle == NULL) {
00072     fprintf(stderr, "You need to pass a valid handle to %s\n", __FUNCTION__);
00073     exit(EXIT_FAILURE);
00074   }
00075   if (handle->mmaped_region != MAP_FAILED) {
00076     munmap(handle->mmaped_region, handle->length);
00077   }
00078   if (handle->file_descriptor >= 0) {
00079     close(handle->file_descriptor);
00080   }
00081 }
```

6.7 library/arm_shared_memory_system.h File Reference

```
#include <stdint.h>
```
Include dependency graph for arm_shared_memory_system.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct arm_shared_t

Typedefs

- typedef struct arm_shared_t arm_shared

Functions

- void ∗ arm_shared_init (arm_shared ∗handle, const uint32_t address, const uint32_t length)
- void arm_shared_close (arm_shared ∗handle)

6.8 arm_shared_memory_system.h

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef __ARM_SHARED_MEMORY_SYSTEMH_
00023 #define __ARM_SHARED_MEMORY_SYSTEMH_
00024
00037 #include <stdint.h>
00038
00039 struct arm_shared_t {
00040   int file_descriptor;
00041   uint32_t address;
00042   uint32_t length;
00043   void *mmaped_region;
00044 };
00048 typedef struct arm_shared_t arm_shared;
00049
00060 extern void *arm_shared_init(arm_shared *handle, const uint32_t address,
00061                             const uint32_t length);
00062
00069 extern void arm_shared_close(arm_shared *handle);
00070
00074 #endif // ARM_READ_SHARED_H
```

6.9 library/audio.c File Reference

```
#include "audio.h"
#include <libpynq.h>
#include <stdint.h>
#include "i2cps.h"
#include "uio.h"
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <time.h>
#include <unistd.h>
```
Include dependency graph for audio.c:

Macros

- #define SAMPLE_RATE 44100
- #define LOG_DOMAIN "audio"

Functions

- void [audio_init](void)
- void [audio_select_input](int input)
- void [write_audio_reg](unsigned char u8RegAddr, unsigned char u8Data, int iic_fd)
- void [config_audio_pll](void)
- void [config_audio_codec](void)
- void [select_line_in](void)
- void [select_mic](void)
- void [deselect](void)
- void [audio_bypass](unsigned int audio_mmap_size, unsigned int nsamples, unsigned int volume, int uio_↩ index)
- void [audio_record](unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, int uio_↩ index)
- void [audio_play](unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, unsigned int volume, int uio_index)
- void [audio_repeat_play](unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, unsigned int volume, unsigned int repetitions)
- void [audio_generate_tone](unsigned int frequency, uint32_t time_ms, unsigned int volume)

6.9.1 Macro Definition Documentation

6.9.1.1 LOG_DOMAIN

```
#define LOG_DOMAIN "audio"
```

Definition at line 70 of file audio.c.

6.9.1.2 SAMPLE_RATE

```
#define SAMPLE_RATE 44100
```

Definition at line 67 of file audio.c.

6.10 audio.c

[Go to the documentation of this file.]
```
00001 /******************************************************************************
00002 * Copyright (c) 2016, Xilinx, Inc.
00003 * All rights reserved.
00004 *
00005 * Redistribution and use in source and binary forms, with or without
00006 * modification, are permitted provided that the following conditions are met:
00007 *
00008 * 1. Redistributions of source code must retain the above copyright notice,
00009 *    this list of conditions and the following disclaimer.
00010 *
00011 * 2. Redistributions in binary form must reproduce the above copyright
00012 *    notice, this list of conditions and the following disclaimer in the
00013 *    documentation and/or other materials provided with the distribution.
00014 *
00015 * 3. Neither the name of the copyright holder nor the names of its
00016 *    contributors may be used to endorse or promote products derived from
00017 *    this software without specific prior written permission.
00018 *
00019 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00020 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00021 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
```

```
00022  *   PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
00023  *   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  *   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  *   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00026  *   OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00027  *   WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00028  *   OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00029  *   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 /******************************************************************************
00034  * @file audio_adau1761.c
00035  *
00036  * Functions to control audio controller.
00037  *
00038  * <pre>
00039  * MODIFICATION HISTORY:
00040  *
00041  * Ver   Who          Date     Changes
00042  * ----- ------------ -------- ----------------------------------------------
00043  * 1.00  Yun Rock Qu  12/04/17 Support for audio codec ADAU1761
00044  * 1.01  Yun Rock Qu  01/02/18 Enable microphone for CTIA and OMTP standards
00045  *
00046  * </pre>
00047  *
00048  ******************************************************************************/
00049 #include "audio.h"
00050 #include <libpynq.h>
00051 #include <stdint.h>
00052
00053 #include "i2cps.h"
00054 #include "uio.h"
00055 #include <fcntl.h>
00056 #include <linux/i2c-dev.h>
00057 #include <math.h>
00058 #include <stdio.h>
00059 #include <stdlib.h>
00060 #include <string.h>
00061 #include <sys/ioctl.h>
00062 #include <sys/mman.h>
00063 #include <sys/stat.h>
00064 #include <time.h>
00065 #include <unistd.h>
00066
00067 #define SAMPLE_RATE 44100
00068
00069 #undef LOG_DOMAIN
00070 #define LOG_DOMAIN "audio"
00071
00072 void audio_init(void) {
00073   config_audio_pll();
00074   config_audio_codec();
00075 }
00076
00077 void audio_select_input(int input) {
00078   if (input == MIC) {
00079     select_mic();
00080   } else if (input == LINE_IN) {
00081     select_line_in();
00082   } else {
00083     pynq_error("audio_select_input: invalid input %d, must be LINE_IN or MIC\n",
00084                input);
00085   }
00086 }
00087
00088 // Original ADAU1761 code
00089
00090 void write_audio_reg(unsigned char u8RegAddr, unsigned char u8Data,
00091                      int iic_fd) {
00092   unsigned char u8TxData[3];
00093   u8TxData[0] = 0x40;
00094   u8TxData[1] = u8RegAddr;
00095   u8TxData[2] = u8Data;
00096   if (writeI2C_asFile(iic_fd, u8TxData, 3) < 0) {
00097     pynq_error("write_audio_reg: unable to write audio register, ensure sudo "
00098                "chmod 666 /dev/i2c-1 has been executed. \n");
00099   }
00100 }
00101
00102 void config_audio_pll(void) {
00103   int iic_index = 1;
00104   unsigned char u8TxData[8], u8RxData[6];
00105   int iic_fd;
00106   iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00107   if (iic_fd < 0) {
00108     pynq_error("config_audio_pll: unable to set I2C %d\n", iic_index);
```

```
00109    }
00110
00111    // Disable Core Clock
00112    write_audio_reg(R0_CLOCK_CONTROL, 0x0E, iic_fd);
00113    /*  MCLK = 10 MHz
00114     *  R = 0100 = 4, N = 0x064C = 1612, M = 0x0C35 = 3125
00115     *  PLL required output = 1024x44.1 KHz = 45.1584 MHz
00116     *  PLLout/MCLK       = 45.1584 MHz/10 MHz = 4.51584 MHz
00117     *                    = R + (N/M)
00118     *                    = 4 + (1612/3125)
00119     *  Fs = PLL/1024 = 44.1 KHz
00120     */
00121
00122    // Register write address [15:8]
00123    u8TxData[0] = 0x40;
00124    // Register write address [7:0]
00125    u8TxData[1] = 0x02;
00126    // byte 6 – M[15:8]
00127    u8TxData[2] = 0x0C;
00128    // byte 5 – M[7:0]
00129    u8TxData[3] = 0x35;
00130    // byte 4 – N[15:8]
00131    u8TxData[4] = 0x06;
00132    // byte 3 – N[7:0]
00133    u8TxData[5] = 0x4C;
00134    // byte 2 – bits 6:3 = R[3:0], 2:1 = X[1:0], 0 = PLL operation mode
00135    u8TxData[6] = 0x21;
00136    // byte 1 – 1 = PLL Lock, 0 = Core clock enable
00137    u8TxData[7] = 0x03;
00138    // Write bytes to PLL control register R1 at 0x4002
00139    if (writeI2C_asFile(iic_fd, u8TxData, 8) < 0) {
00140      pynq_error("config_audio_pll: unable to write audio register, ensure sudo "
00141                 "chmod 666 /dev/i2c-1 has been executed. \n");
00142    }
00143
00144    // Poll PLL Lock bit
00145    u8TxData[0] = 0x40;
00146    u8TxData[1] = 0x02;
00147    do {
00148      if (writeI2C_asFile(iic_fd, u8TxData, 2) < 0) {
00149        pynq_error("writeI2C_asFile: unable to write audio register, ensure sudo "
00150                   "chmod 666 /dev/i2c-1 has been executed. \n");
00151      }
00152      if (readI2C_asFile(iic_fd, u8RxData, 6) < 0) {
00153        pynq_error("readI2C_asFile: unable to write audio register, ensure sudo "
00154                   "chmod 666 /dev/i2c-1 has been executed. \n");
00155      }
00156    } while ((u8RxData[5] & 0x02) == 0);
00157
00158    /* Clock control register:  bit 3       CLKSRC = PLL Clock input
00159     *                          bit 2:1     INFREQ = 1024 x fs
00160     *                          bit 0       COREN = Core Clock enabled
00161     */
00162    write_audio_reg(R0_CLOCK_CONTROL, 0x0F, iic_fd);
00163
00164    if (unsetI2C(iic_fd) < 0) {
00165      pynq_error("config_audio_pll: unable to set I2C %d\n", iic_fd);
00166    }
00167 }
00168
00169 /*******************************************************************************
00170  * Function to configure the audio codec.
00171  * @param   iic_index is the i2c index in /dev list.
00172  * @return  none.
00173  *******************************************************************************/
00174 void config_audio_codec(void) {
00175    int iic_index = 1;
00176    int iic_fd;
00177    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00178    if (iic_fd < 0) {
00179      pynq_error("config_audio_codec: unable to set I2C %d\n", iic_index);
00180    }
00181
00182    /*
00183     * Input path control registers are configured
00184     * in select_mic and select_line_in
00185     */
00186
00187    // Mute Mixer1 and Mixer2 here, enable when MIC and Line In used
00188    write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
00189    write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
00190    // Set LDVOL and RDVOL to 21 dB and Enable left and right differential
00191    write_audio_reg(R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);
00192    write_audio_reg(R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);
00193    // Enable MIC bias
00194    write_audio_reg(R10_RECORD_MICROPHONE_BIAS_CONTROL, 0x01, iic_fd);
00195    // Enable ALC control and noise gate
```

```
00196    write_audio_reg(R14_ALC_CONTROL_3, 0x20, iic_fd);
00197    // Put CODEC in Master mode
00198    write_audio_reg(R15_SERIAL_PORT_CONTROL_0, 0x01, iic_fd);
00199    // Enable ADC on both channels, normal polarity and ADC high-pass filter
00200    write_audio_reg(R19_ADC_CONTROL, 0x33, iic_fd);
00201    // Mute play back Mixer3 and Mixer4 and enable when output is required
00202    write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
00203    write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
00204    // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
00205    write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
00206    write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
00207    // Mute left and right channels output; enable them when output is needed
00208    write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, 0xE5, iic_fd);
00209    write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, 0xE5, iic_fd);
00210    // Enable play back right and left channels
00211    write_audio_reg(R35_PLAYBACK_POWER_MANAGEMENT, 0x03, iic_fd);
00212    // Enable DAC for both channels
00213    write_audio_reg(R36_DAC_CONTROL_0, 0x03, iic_fd);
00214    // Set SDATA_In to DAC
00215    write_audio_reg(R58_SERIAL_INPUT_ROUTE_CONTROL, 0x01, iic_fd);
00216    // Set SDATA_Out to ADC
00217    write_audio_reg(R59_SERIAL_OUTPUT_ROUTE_CONTROL, 0x01, iic_fd);
00218    // Enable DSP and DSP Run
00219    write_audio_reg(R61_DSP_ENABLE, 0x01, iic_fd);
00220    write_audio_reg(R62_DSP_RUN, 0x01, iic_fd);
00221    /*
00222     * Enable Digital Clock Generator 0 and 1.
00223     * Generator 0 generates sample rates for the ADCs, DACs, and DSP.
00224     * Generator 1 generates BCLK and LRCLK for the serial port.
00225     */
00226    write_audio_reg(R65_CLOCK_ENABLE_0, 0x7F, iic_fd);
00227    write_audio_reg(R66_CLOCK_ENABLE_1, 0x03, iic_fd);
00228
00229    if (unsetI2C(iic_fd) < 0) {
00230      pynq_error("config_audio_codec: unable to unset I2C %d\n", iic_index);
00231    }
00232 }
00233
00234 void select_line_in(void) {
00235    int iic_index = 1;
00236    int iic_fd;
00237    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00238    if (iic_fd < 0) {
00239      pynq_error("select_line_in: unable to set I2C %d\n", iic_index);
00240    }
00241
00242    // Mixer 1  (left channel)
00243    write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
00244    // Enable LAUX (MX1AUXG)
00245    write_audio_reg(R5_RECORD_MIXER_LEFT_CONTROL_1, 0x07, iic_fd);
00246
00247    // Mixer 2
00248    write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
00249    // Enable RAUX (MX2AUXG)
00250    write_audio_reg(R7_RECORD_MIXER_RIGHT_CONTROL_1, 0x07, iic_fd);
00251
00252    if (unsetI2C(iic_fd) < 0) {
00253      pynq_error("select_line_in: unable to unset I2C %d\n", iic_index);
00254    }
00255 }
00256
00257 void select_mic(void) {
00258    int iic_index = 1;
00259    int iic_fd;
00260    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00261    if (iic_fd < 0) {
00262      pynq_error("select_mic: unable to set I2C %d, ensure sudo chmod 666 "
00263                 "/dev/i2c-1 has been executed\n",
00264                 iic_index);
00265    }
00266
00267    // Mixer 1 (left channel)
00268    write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
00269    // LDBOOST, set to 20 dB
00270    write_audio_reg(R5_RECORD_MIXER_LEFT_CONTROL_1, 0x10, iic_fd);
00271    // LDVOL, set to 21 dB
00272    write_audio_reg(R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);
00273
00274    // Mixer 2 (right channel)
00275    write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
00276    // RDBOOST, set to 20 dB
00277    write_audio_reg(R7_RECORD_MIXER_RIGHT_CONTROL_1, 0x10, iic_fd);
00278    // RDVOL, set to 21 dB
00279    write_audio_reg(R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);
00280
00281    if (unsetI2C(iic_fd) < 0) {
00282      pynq_error("select_mic: unable to unset I2C %d\n", iic_index);
```

```
00283    }
00284 }
00285
00286 void deselect(void) {
00287    int iic_index = 1;
00288    int iic_fd;
00289    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00290    if (iic_fd < 0) {
00291      pynq_error("deselect: unable to set I2C %d\n", iic_index);
00292    }
00293
00294    // mute mixer 1 in left channel
00295    write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
00296    // mute mixer 2 in right channel
00297    write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
00298
00299    if (unsetI2C(iic_fd) < 0) {
00300      pynq_error("deselect: unable to unset I2C %d\n", iic_index);
00301    }
00302 }
00303
00304 void audio_bypass(unsigned int audio_mmap_size, unsigned int nsamples,
00305                   unsigned int volume, int uio_index) {
00306    if (uio_index > 2) {
00307      pynq_error("audio_bypass: uio_index outside of range. is %d, should be "
00308                 "below 3. \n",
00309                 uio_index);
00310    }
00311    if (volume > 100) {
00312      pynq_error("audio_bypass: volume outside allowed range. Is %d, should be "
00313                 "below 100 \n",
00314                 volume);
00315    }
00316
00317    int iic_index = 1;
00318    int status;
00319    void *uio_ptr;
00320    int DataL, DataR;
00321    int iic_fd;
00322
00323    uio_ptr = setUIO(uio_index, audio_mmap_size);
00324    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00325    if (iic_fd < 0) {
00326      pynq_error("audio_bypass: unable to set I2C %d, ensure sudo chmod 666 "
00327                 "/dev/i2c-1 has been executed\n",
00328                 iic_index);
00329    }
00330
00331    // Mute mixer1 and mixer2 input
00332    write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
00333    write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
00334    // Enable Mixer3 and Mixer4
00335    write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
00336    write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);
00337
00338    unsigned char vol_register = (unsigned char)volume << 2 | 0x3;
00339    // Enable Left/Right Headphone out
00340    write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
00341                    iic_fd);
00342    write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
00343                    iic_fd);
00344
00345    for (unsigned int i = 0; i < nsamples; i++) {
00346      // wait for RX data to become available
00347      do {
00348        status = *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
00349      } while (status == 0);
00350      *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
00351          0x00000001;
00352
00353      // Read the sample from the input
00354      DataL = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_L_REG));
00355      DataR = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_R_REG));
00356
00357      // Write the sample to output
00358      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
00359      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
00360    }
00361
00362    write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
00363    write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
00364    write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
00365    write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
00366    write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, 0xE5, iic_fd);
00367    write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, 0xE5, iic_fd);
00368
00369    if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
```

```
00370      pynq_error("audio_bypass: unable to free UIO %d, ensure sudo chmod 666 "
00371                 "/dev/i2c-1 has been executed\n",
00372                 uio_index);
00373    }
00374    if (unsetI2C(iic_fd) < 0) {
00375      pynq_error("audio_bypass: unable to unset I2C %d, ensure sudo chmod 666 "
00376                 "/dev/i2c-1 has been executed\n",
00377                 iic_index);
00378    }
00379 }
00380
00381 void audio_record(unsigned int audio_mmap_size, unsigned int *BufAddr,
00382                   unsigned int nsamples, int uio_index) {
00383    if (uio_index > 2) {
00384      pynq_error("audio_record: uio_index outside of range. is %d, should be "
00385                 "below 3. \n",
00386                 uio_index);
00387    }
00388    int iic_index = 1;
00389    unsigned int i, status;
00390    void *uio_ptr;
00391    int DataL, DataR;
00392    int iic_fd;
00393
00394    uio_ptr = setUIO(uio_index, audio_mmap_size);
00395    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00396    if (iic_fd < 0) {
00397      pynq_error("audio_record: unable to set I2C %d, ensure sudo chmod 666 "
00398                 "/dev/i2c-1 has been executed\n",
00399                 iic_index);
00400    }
00401
00402    for (i = 0; i < nsamples; i++) {
00403      do {
00404        status = *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
00405      } while (status == 0);
00406      *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
00407          0x00000001;
00408
00409      // Read the sample from the input
00410      DataL = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_L_REG));
00411      DataR = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_R_REG));
00412
00413      // Write the sample into memory
00414      *(BufAddr + 2 * i) = DataL;
00415      *(BufAddr + 2 * i + 1) = DataR;
00416    }
00417
00418    if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
00419      pynq_error("audio_record: unable to free UIO %d, ensure sudo chmod 666 "
00420                 "/dev/i2c-1 has been executed\n",
00421                 uio_index);
00422    }
00423    if (unsetI2C(iic_fd) < 0) {
00424      pynq_error("audio_record: unable to unset I2C %d, ensure sudo chmod 666 "
00425                 "/dev/i2c-1 has been executed\n",
00426                 iic_index);
00427    }
00428 }
00429
00430 void audio_play(unsigned int audio_mmap_size, unsigned int *BufAddr,
00431                 unsigned int nsamples, unsigned int volume, int uio_index) {
00432    if (uio_index > 2) {
00433      pynq_error(
00434          "audio_play: uio_index outside of range. is %d, should be below 3. \n",
00435          uio_index);
00436    }
00437    if (volume > 100) {
00438      pynq_error("audio_play: volume outside allowed range. Is %d, should be "
00439                 "below 100 \n",
00440                 volume);
00441    }
00442    int iic_index = 1;
00443    unsigned int i, status;
00444    void *uio_ptr;
00445    int DataL, DataR;
00446    int iic_fd;
00447
00448    uio_ptr = setUIO(uio_index, audio_mmap_size);
00449    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00450    if (iic_fd < 0) {
00451      pynq_error("audio_play: unable to set I2C %d, ensure sudo chmod 666 "
00452                 "/dev/i2c-1 has been executed\n",
00453                 iic_index);
00454    }
00455
00456    // Unmute left and right DAC, enable Mixer3 and Mixer4
```

```
00457    write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
00458    write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);
00459
00460    unsigned char vol_register = (unsigned char)volume << 2 | 0x3;
00461    // Enable Left/Right Headphone out
00462    write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
00463                    iic_fd);
00464    write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
00465                    iic_fd);
00466
00467    for (i = 0; i < nsamples; i++) {
00468      do {
00469        status = *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
00470      } while (status == 0);
00471      *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
00472          0x00000001;
00473
00474      // Read the sample from memory
00475      DataL = *(BufAddr + 2 * i);
00476      DataR = *(BufAddr + 2 * i + 1);
00477
00478      // Write the sample to output
00479      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
00480      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
00481    }
00482
00483    // Mute left and right DAC
00484    write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
00485    write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
00486    // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
00487    write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
00488    write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
00489
00490    if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
00491      pynq_error("audio_play: unable to free UIO %d, ensure sudo chmod 666 "
00492                 "/dev/i2c-1 has been executed\n",
00493                 uio_index);
00494    }
00495    if (unsetI2C(iic_fd) < 0) {
00496      pynq_error("audio_play: unable to unset I2C %d, ensure sudo chmod 666 "
00497                 "/dev/i2c-1 has been executed\n",
00498                 iic_index);
00499    }
00500 }
00501
00502 void audio_repeat_play(unsigned int audio_mmap_size, unsigned int *BufAddr,
00503                        unsigned int nsamples, unsigned int volume,
00504                        unsigned int repetitions) {
00505    if (volume > 100) {
00506      pynq_error("audio_repeat_play: volume outside allowed range. Is %d, should "
00507                 "be below 100 \n",
00508                 volume);
00509    }
00510    int iic_index = 1;
00511    unsigned int i, status;
00512    void *uio_ptr;
00513    int DataL, DataR;
00514    int iic_fd;
00515
00516    uio_ptr = setUIO(0, audio_mmap_size);
00517    iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00518    if (iic_fd < 0) {
00519      pynq_error("audio_repeat_play: unable to set I2C %d, ensure sudo chmod 666 "
00520                 "/dev/i2c-1 has been executed\n",
00521                 iic_index);
00522    }
00523
00524    // Unmute left and right DAC, enable Mixer3 and Mixer4
00525    write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
00526    write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);
00527
00528    unsigned char vol_register = (unsigned char)volume << 2 | 0x3;
00529    // Enable Left/Right Headphone out
00530    write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
00531                    iic_fd);
00532    write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
00533                    iic_fd);
00534
00535    for (unsigned int repeat = 0; repeat < repetitions; repeat++) {
00536      for (i = 0; i < nsamples; i++) {
00537        do {
00538          status =
00539              *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
00540        } while (status == 0);
00541        *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
00542            0x00000001;
00543
```

```
00544        // Read the sample from memory
00545        DataL = *(BufAddr + 2 * i);
00546        DataR = *(BufAddr + 2 * i + 1);
00547
00548        // Write the sample to output
00549        *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
00550        *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
00551      }
00552   }
00553   // Mute left and right DAC
00554   write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
00555   write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
00556   // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
00557   write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
00558   write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
00559
00560   if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
00561     pynq_error("audio_repeat_play: unable to free UIO %d\n", 0);
00562   }
00563   if (unsetI2C(iic_fd) < 0) {
00564     pynq_error("audio_repeat_play: unable to unset I2C %d, ensure sudo chmod "
00565                "666 /dev/i2c-1 has been executed\n",
00566                iic_index);
00567   }
00568 }
00569
00570 void audio_generate_tone(unsigned int frequency, uint32_t time_ms,
00571                          unsigned int volume) {
00572
00573   if (frequency < 10) {
00574     pynq_error("audio_generate_tone: frequency should be 10 or higher, "
00575                "frequency is: %d\n",
00576                frequency);
00577   }
00578   if (volume > 100) {
00579     pynq_error("audio_generate_tone: volume outside allowed range. Is %d, "
00580                "should be below 100 \n",
00581                volume);
00582   }
00583   double period = 1 / ((double)(frequency));
00584   unsigned int samplesPerPeriod = (int)(SAMPLE_RATE * period);
00585   double time_s = ((double)(time_ms)) / 1000;
00586   int totalPeriods = (int)(time_s / period); // Number of times one period must
00587                                              // be played to play for time_ms
00588
00589   uint32_t audioBuffer[16 * 1024 + 1] = {0};
00590   unsigned int i, status;
00591
00592   for (i = 0; i < samplesPerPeriod; i++) {
00593     double t = (double)i / SAMPLE_RATE;
00594     double value = sin(6.28318531 * frequency * t); // 6.28... = 2pi
00595     value = value + 1;
00596     value = value * 16000;
00597     audioBuffer[2 * i] = (uint32_t)value;
00598     audioBuffer[2 * i + 1] = (uint32_t)value;
00599   }
00600
00601   unsigned int audio_mmap_size = 64 * 1024;
00602   unsigned int *BufAddr = audioBuffer;
00603   int iic_index = 1;
00604   void *uio_ptr;
00605   int DataL, DataR;
00606   int iic_fd;
00607
00608   uio_ptr = setUIO(0, audio_mmap_size);
00609   iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
00610   if (iic_fd < 0) {
00611     pynq_error("audio_generate_tone: unable to set I2C %d, ensure sudo chmod "
00612                "666 /dev/i2c-1 has been executed\n",
00613                iic_index);
00614   }
00615
00616   // Unmute left and right DAC, enable Mixer3 and Mixer4
00617   write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
00618   write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);
00619
00620   unsigned char vol_register = (unsigned char)volume << 2 | 0x3;
00621   // Enable Left/Right Headphone out
00622   write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
00623                   iic_fd);
00624   write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
00625                   iic_fd);
00626
00627   for (int period = 0; period < totalPeriods; period++) {
00628     for (i = 0; i < samplesPerPeriod; i++) {
00629       do {
00630         status =
```

```
00631             *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
00632         } while (status == 0);
00633         *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
00634             0x00000001;
00635
00636         // Read the sample from memory
00637         DataL = *(BufAddr + 2 * i);
00638         DataR = *(BufAddr + 2 * i + 1);
00639
00640         // Write the sample to output
00641         *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
00642         *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
00643     }
00644 }
00645 // Mute left and right DAC
00646 write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
00647 write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
00648 // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
00649 write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
00650 write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
00651
00652 if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
00653     pynq_error("audio_generate_tone: unable to free UIO %d, ensure sudo chmod "
00654                "666 /dev/i2c-1 has been executed\n",
00655                0);
00656 }
00657 if (unsetI2C(iic_fd) < 0) {
00658     pynq_error("audio_generate_tone: unable to unset I2C %d, ensure has been "
00659                "executed\n",
00660                iic_index);
00661 }
00662 }
```

6.11   library/audio.h File Reference

`#include <stdint.h>`
Include dependency graph for audio.h: This graph shows which files directly or indirectly include this file:

Macros

- #define LINE_IN 0
- #define MIC 1
- #define IIC_SLAVE_ADDR 0x3B
- #define IIC_SCLK_RATE 400000
- #define I2S_DATA_RX_L_REG 0x00
- #define I2S_DATA_RX_R_REG 0x04
- #define I2S_DATA_TX_L_REG 0x08
- #define I2S_DATA_TX_R_REG 0x0C
- #define I2S_STATUS_REG 0x10

Enumerations

- enum audio_adau1761_regs {
  R0_CLOCK_CONTROL = 0x00 , R1_PLL_CONTROL = 0x02 , R2_DIGITAL_MIC_JACK_DETECTION_CONTROL
  = 0x08 , R3_RECORD_POWER_MANAGEMENT = 0x09 ,
  R4_RECORD_MIXER_LEFT_CONTROL_0 = 0x0A , R5_RECORD_MIXER_LEFT_CONTROL_1 = 0x0B ,
  R6_RECORD_MIXER_RIGHT_CONTROL_0 = 0x0C , R7_RECORD_MIXER_RIGHT_CONTROL_1 = 0x0D
  ,
  R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL = 0x0E , R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL
  = 0x0F , R10_RECORD_MICROPHONE_BIAS_CONTROL = 0x10 , R11_ALC_CONTROL_0 = 0x11 ,
  R12_ALC_CONTROL_1 = 0x12 , R13_ALC_CONTROL_2 = 0x13 , R14_ALC_CONTROL_3 = 0x14 ,
  R15_SERIAL_PORT_CONTROL_0 = 0x15 ,
  R16_SERIAL_PORT_CONTROL_1 = 0x16 , R17_CONVERTER_CONTROL_0 = 0x17 , R18_CONVERTER_CONTROL_1

= 0x18 , R19_ADC_CONTROL = 0x19 ,
R20_LEFT_INPUT_DIGITAL_VOLUME = 0x1A , R21_RIGHT_INPUT_DIGITAL_VOLUME = 0x1B ,
R22_PLAYBACK_MIXER_LEFT_CONTROL_0 = 0x1C , R23_PLAYBACK_MIXER_LEFT_CONTROL_1
= 0x1D ,
R24_PLAYBACK_MIXER_RIGHT_CONTROL_0 = 0x1E , R25_PLAYBACK_MIXER_RIGHT_CONTROL_1 =
0x1F , R26_PLAYBACK_LR_MIXER_LEFT_LINE_OUTPUT_CONTROL = 0x20 , R27_PLAYBACK_LR_MIXER_RIGHT_LINE_
= 0x21 ,
R28_PLAYBACK_LR_MIXER_MONO_OUTPUT_CONTROL = 0x22 , R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CON
= 0x23 , R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL = 0x24 , R31_PLAYBACK_LINE_OUTPUT_LEFT_VO
= 0x25 ,
R32_PLAYBACK_LINE_OUTPUT_RIGHT_VOLUME_CONTROL = 0x26 , R33_PLAYBACK_MONO_OUTPUT_CONTROL
= 0x27 , R34_PLAYBACK_POP_CLICK_SUPPRESSION = 0x28 , R35_PLAYBACK_POWER_MANAGEMENT
= 0x29 ,
R36_DAC_CONTROL_0 = 0x2A , R37_DAC_CONTROL_1 = 0x2B , R38_DAC_CONTROL_2 = 0x2C ,
R39_SERIAL_PORT_PAD_CONTROL = 0x2D ,
R40_CONTROL_PORT_PAD_CONTROL_0 = 0x2F , R41_CONTROL_PORT_PAD_CONTROL_1 = 0x30 ,
R42_JACK_DETECT_PIN_CONTROL = 0x31 , R67_DEJITTER_CONTROL = 0x36 ,
R58_SERIAL_INPUT_ROUTE_CONTROL = 0xF2 , R59_SERIAL_OUTPUT_ROUTE_CONTROL = 0xF3 ,
R61_DSP_ENABLE = 0xF5 , R62_DSP_RUN = 0xF6 ,
R63_DSP_SLEW_MODES = 0xF7 , R64_SERIAL_PORT_SAMPLING_RATE = 0xF8 , R65_CLOCK_ENABLE_0
= 0xF9 , R66_CLOCK_ENABLE_1 = 0xFA }

Functions

- void audio_init (void)
- void audio_select_input (int input)
- void write_audio_reg (unsigned char u8RegAddr, unsigned char u8Data, int iic_fd)
- void config_audio_pll (void)
- void config_audio_codec (void)
- void select_line_in (void)
- void select_mic (void)
- void deselect (void)
- void audio_bypass (unsigned int audio_mmap_size, unsigned int nsamples, unsigned int volume, int uio_↩
  index)
- void audio_record (unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, int uio_↩
  index)
- void audio_play (unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, unsigned int
  volume, int uio_index)
- void audio_repeat_play (unsigned int audio_mmap_size, unsigned int ∗BufAddr, unsigned int nsamples, un-
  signed int volume, unsigned int repetitions)
- void audio_generate_tone (unsigned int frequency, uint32_t time_ms, unsigned int volume)

6.12 audio.h

Go to the documentation of this file.
```
00001 #ifndef AUDIO_H
00002 #define AUDIO_H
00003 #include <stdint.h>
00004
00032 #define LINE_IN 0
00033 #define MIC 1
00034
00035 // Slave address for the ADAU audio controller 8
00036 #define IIC_SLAVE_ADDR 0x3B
00037
00038 // I2C Serial Clock frequency in Hertz
00039 #define IIC_SCLK_RATE 400000
00040
```

```
00041 // I2S Register
00042 #define I2S_DATA_RX_L_REG 0x00
00043 #define I2S_DATA_RX_R_REG 0x04
00044 #define I2S_DATA_TX_L_REG 0x08
00045 #define I2S_DATA_TX_R_REG 0x0C
00046 #define I2S_STATUS_REG 0x10
00047
00048 // Audio registers
00049 enum audio_adau1761_regs {
00050   R0_CLOCK_CONTROL = 0x00,
00051   R1_PLL_CONTROL = 0x02,
00052   R2_DIGITAL_MIC_JACK_DETECTION_CONTROL = 0x08,
00053   R3_RECORD_POWER_MANAGEMENT = 0x09,
00054   R4_RECORD_MIXER_LEFT_CONTROL_0 = 0x0A,
00055   R5_RECORD_MIXER_LEFT_CONTROL_1 = 0x0B,
00056   R6_RECORD_MIXER_RIGHT_CONTROL_0 = 0x0C,
00057   R7_RECORD_MIXER_RIGHT_CONTROL_1 = 0x0D,
00058   R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL = 0x0E,
00059   R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL = 0x0F,
00060   R10_RECORD_MICROPHONE_BIAS_CONTROL = 0x10,
00061   R11_ALC_CONTROL_0 = 0x11,
00062   R12_ALC_CONTROL_1 = 0x12,
00063   R13_ALC_CONTROL_2 = 0x13,
00064   R14_ALC_CONTROL_3 = 0x14,
00065   R15_SERIAL_PORT_CONTROL_0 = 0x15,
00066   R16_SERIAL_PORT_CONTROL_1 = 0x16,
00067   R17_CONVERTER_CONTROL_0 = 0x17,
00068   R18_CONVERTER_CONTROL_1 = 0x18,
00069   R19_ADC_CONTROL = 0x19,
00070   R20_LEFT_INPUT_DIGITAL_VOLUME = 0x1A,
00071   R21_RIGHT_INPUT_DIGITAL_VOLUME = 0x1B,
00072   R22_PLAYBACK_MIXER_LEFT_CONTROL_0 = 0x1C,
00073   R23_PLAYBACK_MIXER_LEFT_CONTROL_1 = 0x1D,
00074   R24_PLAYBACK_MIXER_RIGHT_CONTROL_0 = 0x1E,
00075   R25_PLAYBACK_MIXER_RIGHT_CONTROL_1 = 0x1F,
00076   R26_PLAYBACK_LR_MIXER_LEFT_LINE_OUTPUT_CONTROL = 0x20,
00077   R27_PLAYBACK_LR_MIXER_RIGHT_LINE_OUTPUT_CONTROL = 0x21,
00078   R28_PLAYBACK_LR_MIXER_MONO_OUTPUT_CONTROL = 0x22,
00079   R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL = 0x23,
00080   R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL = 0x24,
00081   R31_PLAYBACK_LINE_OUTPUT_LEFT_VOLUME_CONTROL = 0x25,
00082   R32_PLAYBACK_LINE_OUTPUT_RIGHT_VOLUME_CONTROL = 0x26,
00083   R33_PLAYBACK_MONO_OUTPUT_CONTROL = 0x27,
00084   R34_PLAYBACK_POP_CLICK_SUPPRESSION = 0x28,
00085   R35_PLAYBACK_POWER_MANAGEMENT = 0x29,
00086   R36_DAC_CONTROL_0 = 0x2A,
00087   R37_DAC_CONTROL_1 = 0x2B,
00088   R38_DAC_CONTROL_2 = 0x2C,
00089   R39_SERIAL_PORT_PAD_CONTROL = 0x2D,
00090   R40_CONTROL_PORT_PAD_CONTROL_0 = 0x2F,
00091   R41_CONTROL_PORT_PAD_CONTROL_1 = 0x30,
00092   R42_JACK_DETECT_PIN_CONTROL = 0x31,
00093   R67_DEJITTER_CONTROL = 0x36,
00094   R58_SERIAL_INPUT_ROUTE_CONTROL = 0xF2,
00095   R59_SERIAL_OUTPUT_ROUTE_CONTROL = 0xF3,
00096   R61_DSP_ENABLE = 0xF5,
00097   R62_DSP_RUN = 0xF6,
00098   R63_DSP_SLEW_MODES = 0xF7,
00099   R64_SERIAL_PORT_SAMPLING_RATE = 0xF8,
00100   R65_CLOCK_ENABLE_0 = 0xF9,
00101   R66_CLOCK_ENABLE_1 = 0xFA
00102 };
00103
00109 extern void audio_init(void);
00110
00116 extern void audio_select_input(int input);
00117
00118 // Original ADAU1761 code
00119
00120 extern void write_audio_reg(unsigned char u8RegAddr, unsigned char u8Data,
00121                             int iic_fd);
00122
00123 extern void config_audio_pll(void);
00124
00125 extern void config_audio_codec(void);
00126
00130 extern void select_line_in(void);
00131
00135 extern void select_mic(void);
00136
00140 extern void deselect(void);
00141
00149 extern void audio_bypass(unsigned int audio_mmap_size, unsigned int nsamples,
00150                          unsigned int volume, int uio_index);
00151
00164 extern void audio_record(unsigned int audio_mmap_size, unsigned int *BufAddr,
00165                          unsigned int nsamples, int uio_index);
```

```
00166
00167 /*
00168  * @brief Function to support audio playing without the audio codec controller.
00169  *
00170  * Notice that the buffer has to be twice the size of the number of samples,
00171  * because both left and right channels are sampled.
00172  * Consecutive indexes are played synchronisly on left and right output.
00173  *
00174  * @param   audio_mmap_size is the address range of the audio codec.
00175  * @param   BufAddr is the buffer address.
00176  * @param   nsamples is the number of samples.
00177  * @param   uio_index is the uio index in /dev list.
00178  * @param   volume is the volume of the output.
00179  */
00180 extern void audio_play(unsigned int audio_mmap_size, unsigned int *BufAddr,
00181                        unsigned int nsamples, unsigned int volume,
00182                        int uio_index);
00183
00193 extern void audio_repeat_play(unsigned int audio_mmap_size,
00194                               unsigned int *BufAddr, unsigned int nsamples,
00195                               unsigned int volume, unsigned int repetitions);
00196
00197 /*
00198  * @brief Function to generate a specific tone on the audio output.
00199  * @param   frequency is the frequency in Hz to be played.
00200  * @param   time_ms is the time the frequency should be played in ms.
00201  * @param   volume is the volume of the output.
00202  */
00203 extern void audio_generate_tone(unsigned int frequency, uint32_t time_ms,
00204                                 unsigned int volume);
00205
00210 #endif
```

6.13   library/buttons.c File Reference

#include <buttons.h>
#include <gpio.h>
#include <log.h>
#include <pinmap.h>
#include <platform.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>

Include dependency graph for buttons.c:

Macros

- #define LOG_DOMAIN "buttons"

Functions

- void buttons_init (void)
- void buttons_destroy (void)
- void switches_init (void)
- void switches_destroy (void)
- int get_button_state (const int button)
- int wait_until_button_state (const int button, const int state)
- int sleep_msec_button_pushed (const int button, const int ms)
- void sleep_msec_buttons_pushed (int button_states[ ], const int ms)
- int wait_until_button_pushed (const int button)
- int wait_until_button_released (const int button)
- int wait_until_any_button_pushed (void)
- int wait_until_any_button_released (void)
- int get_switch_state (const int switch_num)

6.13.1 Macro Definition Documentation

6.13.1.1 LOG_DOMAIN

```
#define LOG_DOMAIN "buttons"
```

Definition at line 34 of file buttons.c.

6.14 buttons.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <buttons.h>
00023 #include <gpio.h>
00024 #include <log.h>
00025 #include <pinmap.h>
00026 #include <platform.h>
00027 #include <stdbool.h>
00028 #include <stdio.h>
00029 #include <stdlib.h>
00030 #include <sys/time.h>
00031 #include <unistd.h>
00032
00033 #undef LOG_DOMAIN
00034 #define LOG_DOMAIN "buttons"
00035
00036 static bool buttons_initialized = false;
00037 static bool switches_initialized = false;
00038
00039 void buttons_init(void) {
00040   if (buttons_initialized == true) {
00041     pynq_error("buttons_destroy: buttons already initialized\n");
00042   }
00043   gpio_set_direction(IO_BTN0, GPIO_DIR_INPUT);
00044   gpio_set_direction(IO_BTN1, GPIO_DIR_INPUT);
00045   gpio_set_direction(IO_BTN2, GPIO_DIR_INPUT);
00046   gpio_set_direction(IO_BTN3, GPIO_DIR_INPUT);
00047   buttons_initialized = true;
00048 }
00049
00050 void buttons_destroy(void) { /* Anything to do here? */
00051   if (buttons_initialized == false) {
00052     pynq_error("buttons_destroy: buttons weren't initialized\n");
00053   }
00054 }
00055
00056 void switches_init(void) {
00057   if (switches_initialized == true) {
00058     pynq_error("switches_destroy: switches already initialized\n");
00059   }
00060   gpio_set_direction(IO_SW0, GPIO_DIR_INPUT);
00061   gpio_set_direction(IO_SW1, GPIO_DIR_INPUT);
00062   switches_initialized = true;
00063 }
00064
00065 void switches_destroy(void) { /* Anything to do here? */
00066   if (switches_initialized == false) {
```

```
00067      pynq_error("switches_destroy: switches weren't initialized\n");
00068    }
00069 }
00070
00071 int get_button_state(const int button) {
00072   if (buttons_initialized == false) {
00073     pynq_error("get_button_state: buttons weren't initialized\n");
00074   }
00075   if (button < 0 || button >= NUM_BUTTONS) {
00076     pynq_error("get_button_state: invalid button=%d, must be 0..%d-1\n",
00077               NUM_BUTTONS);
00078   }
00079   return (gpio_get_level(IO_BTN0 + button) == GPIO_LEVEL_LOW ? BUTTON_NOT_PUSHED
00080                                                             : BUTTON_PUSHED);
00081 }
00082
00083 int wait_until_button_state(const int button, const int state) {
00084   if (buttons_initialized == false) {
00085     pynq_error("wait_until_button_state: buttons weren't initialized\n");
00086   }
00087   if (button < 0 || button >= NUM_BUTTONS) {
00088     pynq_error("get_button_state: invalid button=%d, must be 0..%d-1\n", button,
00089               NUM_BUTTONS);
00090   }
00091   const io_t btn = IO_BTN0 + button;
00092   if (gpio_get_direction(btn) != GPIO_DIR_INPUT) {
00093     pynq_error("get_button_state: button %d has not been set as input\n",
00094               button);
00095   }
00096   struct timeval call, close;
00097   int dTime;
00098   gettimeofday(&call, NULL);
00099   const unsigned int check =
00100       (state == BUTTON_NOT_PUSHED ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH);
00101   while (gpio_get_level(btn) != check) {
00102   }
00103   gettimeofday(&close, NULL);
00104   dTime = (close.tv_sec - call.tv_sec) * 1000.0;    // # of ms
00105   dTime += (close.tv_usec - call.tv_usec) / 1000.0; // # of usec in ms
00106   return dTime;
00107 }
00108
00109 int sleep_msec_button_pushed(const int button, const int ms) {
00110   if (buttons_initialized == false) {
00111     pynq_error("sleep_msec_button: buttons weren't initialized\n");
00112   }
00113   if (button < 0 || button >= NUM_BUTTONS) {
00114     pynq_error("sleep_msec_button_pushed: invalid button=%d, must be 0..%d-1\n",
00115               button, NUM_BUTTONS);
00116   }
00117   const io_t btn = IO_BTN0 + button;
00118   if (gpio_get_direction(btn) != GPIO_DIR_INPUT) {
00119     pynq_error(
00120         "sleep_msec_button_pushed: button %d has not been set as input\n",
00121         button);
00122   }
00123   int status;
00124   struct timeval call, close;
00125   double dTime;
00126   // mapping call time to call struct
00127   gettimeofday(&call, NULL);
00128   do {
00129     // update level and latch if is pushed
00130     if (status != GPIO_LEVEL_HIGH) {
00131       status = gpio_get_level(btn);
00132     }
00133     (void)gettimeofday(&close, NULL);
00134     dTime = (close.tv_sec - call.tv_sec) * 1000.0;    // # of ms
00135     dTime += (close.tv_usec - call.tv_usec) / 1000.0; // # of usec in ms
00136   } while (dTime < ms);
00137   return (status == GPIO_LEVEL_LOW ? BUTTON_NOT_PUSHED : BUTTON_PUSHED);
00138 }
00139
00140 void sleep_msec_buttons_pushed(int button_states[], const int ms) {
00141   if (buttons_initialized == false) {
00142     pynq_error("sleep_msec_buttons_pushed: buttons weren't initialized\n");
00143   }
00144   if (button_states == NULL) {
00145     pynq_error("sleep_msec_buttons_pushed: button_states is NULL\n");
00146   }
00147   struct timeval call, close;
00148   int dTime;
00149   const io_t buttons[NUM_BUTTONS] = {IO_BTN0, IO_BTN1, IO_BTN2, IO_BTN3};
00150   // mapping call time to call struct
00151   (void)gettimeofday(&call, NULL);
00152   do {
00153     for (int i = 0; i < NUM_BUTTONS; i++) {
```

```
00154          if (button_states[i] != BUTTON_PUSHED) {
00155            button_states[i] =
00156                (gpio_get_level(buttons[i]) == GPIO_LEVEL_HIGH ? BUTTON_PUSHED
00157                                                   : BUTTON_NOT_PUSHED);
00158          }
00159        }
00160        (void)gettimeofday(&close, NULL);
00161        dTime = (close.tv_sec - call.tv_sec) * 1000.0;     // # of ms
00162        dTime += (close.tv_usec - call.tv_usec) / 1000.0; // # of usec in ms
00163    } while (dTime < ms);
00164 }
00165
00166 int wait_until_button_pushed(const int button) {
00167    // all checks are done in wait_until_button state
00168    return wait_until_button_state(button, BUTTON_PUSHED);
00169 }
00170
00171 int wait_until_button_released(const int button) {
00172    // all checks are done in wait_until_button state
00173    return wait_until_button_state(button, BUTTON_NOT_PUSHED);
00174 }
00175
00176 int wait_until_any_button_pushed(void) {
00177    const io_t buttons[NUM_BUTTONS] = {IO_BTN0, IO_BTN1, IO_BTN2, IO_BTN3};
00178    if (buttons_initialized == false) {
00179      pynq_error("wait_until_any_button_pushed: buttons weren't initialized\n");
00180    }
00181    for (int b = 0; b < NUM_BUTTONS; b++) {
00182      if (gpio_get_direction(b) != GPIO_DIR_INPUT) {
00183        pynq_error(
00184            "wait_until_any_button_pushed: button %d has not been set as input\n",
00185            b);
00186      }
00187    }
00188    do {
00189      for (int b = 0; b < NUM_BUTTONS; b++) {
00190        if (gpio_get_level(buttons[b]) == GPIO_LEVEL_HIGH) {
00191          return b; // we return the index, i.e. 0..NUM_BUTTONS-1
00192        }
00193      }
00194    } while (true);
00195 }
00196
00197 int wait_until_any_button_released(void) {
00198    const io_t buttons[NUM_BUTTONS] = {IO_BTN0, IO_BTN1, IO_BTN2, IO_BTN3};
00199    if (buttons_initialized == false) {
00200      pynq_error("wait_until_any_button_released: buttons weren't initialized\n");
00201    }
00202    for (int b = 0; b < NUM_BUTTONS; b++) {
00203      if (gpio_get_direction(b) != GPIO_DIR_INPUT) {
00204        pynq_error("wait_until_any_button_released: button %d has not been set "
00205                   "as input\n",
00206                   b);
00207      }
00208    }
00209    do {
00210      for (int b = 0; b < NUM_BUTTONS; b++) {
00211        if (gpio_get_level(buttons[b]) == GPIO_LEVEL_LOW)
00212          return b; // we return the index, i.e. 0..NUM_BUTTONS-1
00213      }
00214    } while (true);
00215 }
00216
00217 int get_switch_state(const int switch_num) {
00218    if (switches_initialized == false) {
00219      pynq_error("get_switch_state: switches weren't initialized\n");
00220    }
00221    if (switch_num != SWITCH0 && switch_num != SWITCH1) {
00222      pynq_error("get_switch_state: invalid switch_num=%d, must be 0..%i-1\n",
00223                 switch_num, NUM_SWITCHES);
00224    }
00225    return (gpio_get_level(IO_SW0 + switch_num) == GPIO_LEVEL_LOW ? SWITCH_ON
00226                                                   : SWITCH_OFF);
00227 }
```

6.15 library/buttons.h File Reference

```
#include <gpio.h>
```
Include dependency graph for buttons.h: This graph shows which files directly or indirectly include this file:

Macros

- #define BUTTON_NOT_PUSHED 0
- #define BUTTON_PUSHED 1
- #define SWITCH_OFF 0
- #define SWITCH_ON 1

Enumerations

- enum button_index_t {
  BUTTON0 , BUTTON1 , BUTTON2 , BUTTON3 ,
  NUM_BUTTONS }
- enum switches_index_t { SWITCH0 , SWITCH1 , NUM_SWITCHES }

Functions

- void switches_init (void)
- void switches_destroy (void)
- void buttons_init (void)
- void buttons_destroy (void)
- int get_button_state (const int button)
- int wait_until_button_state (const int button, const int state)
- int sleep_msec_button_pushed (const int button, const int msec)
- void sleep_msec_buttons_pushed (int button_states[ ], const int ms)
- int wait_until_button_pushed (const int button)
- int wait_until_button_released (const int button)
- int wait_until_any_button_pushed (void)
- int wait_until_any_button_released (void)
- int get_switch_state (const int switch_num)

6.16 buttons.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef BUTTONS_H
00023 #define BUTTONS_H
00024
00025 #include <gpio.h>
00026
00074 #define BUTTON_NOT_PUSHED 0
00075 #define BUTTON_PUSHED 1
00076 #define SWITCH_OFF 0
```

```
00077 #define SWITCH_ON 1
00078
00086 typedef enum { BUTTON0, BUTTON1, BUTTON2, BUTTON3, NUM_BUTTONS } button_index_t;
00087
00094 typedef enum { SWITCH0, SWITCH1, NUM_SWITCHES } switches_index_t;
00095
00099 extern void switches_init(void);
00100
00104 extern void switches_destroy(void);
00105
00109 extern void buttons_init(void);
00110
00114 extern void buttons_destroy(void);
00115
00123 extern int get_button_state(const int button);
00124
00135 extern int wait_until_button_state(const int button, const int state);
00136
00147 extern int sleep_msec_button_pushed(const int button, const int msec);
00148
00157 extern void sleep_msec_buttons_pushed(int button_states[], const int ms);
00158
00167 extern int wait_until_button_pushed(const int button);
00168
00177 extern int wait_until_button_released(const int button);
00178
00186 extern int wait_until_any_button_pushed(void);
00187
00195 extern int wait_until_any_button_released(void);
00196
00203 extern int get_switch_state(const int switch_num);
00204
00209 #endif
```

6.17 library/display.c File Reference

```
#include <arm_shared_memory_system.h>
#include <display.h>
#include <gpio.h>
#include <lcdconfig.h>
#include <log.h>
#include <math.h>
#include <platform.h>
#include <string.h>
#include <switchbox.h>
#include <unistd.h>
#include <util.h>
```
Include dependency graph for display.c:

Macros

- #define LOG_DOMAIN "display"
- #define TAG "ST7789"
- #define _DEBUG_ 0
- #define M_PI 3.14159265358979323846
- #define GPIO_MODE_OUTPUT 1

Enumerations

- enum spi_mode_t { SPI_Data_Mode = 1 , SPI_Command_Mode = 0 }

Functions

- gpio_level_t spi_to_gpio (spi_mode_t mode)
- bool spi_master_write_command (display_t ∗display, uint8_t cmd)
- bool spi_master_write_data_byte (display_t ∗display, uint8_t data)
- bool spi_master_write_data_word (display_t ∗display, uint16_t data)
- bool spi_master_write_addr (display_t ∗display, uint16_t addr1, uint16_t addr2)
- bool spi_master_write_color (display_t ∗display, uint16_t color, uint16_t size)
- bool spi_master_write_colors (display_t ∗display, uint16_t ∗colors, uint16_t size)
- void spi_master_init (display_t ∗display)
- void displayInit (display_t ∗display, int width, int height, int offsetx, int offsety)
- void display_set_flip (display_t ∗display, bool xflip, bool yflip)
- void display_init (display_t ∗display)
- void display_destroy (display_t ∗display __attribute__((unused)))
- void displayDrawPixel (display_t ∗display, uint16_t x, uint16_t y, uint16_t color)
- void displayDrawMultiPixels (display_t ∗display, uint16_t x, uint16_t y, uint16_t size, uint16_t ∗colors)
- void displayDrawFillRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDisplayOff (display_t ∗display)
- void displayDisplayOn (display_t ∗display)
- void displayFillScreen (display_t ∗display, uint16_t color)
- void displayDrawLine (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDrawRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDrawRectAngle (display_t ∗display, uint16_t xc, uint16_t yc, uint16_t w, uint16_t h, uint16_t angle, uint16_t color)
- void displayDrawTriangle (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t x3, uint16_t y3, uint16_t color)
- void displayDrawTriangleCenter (display_t ∗display, uint16_t xc, uint16_t yc, uint16_t w, uint16_t h, uint16_t angle, uint16_t color)
- void displayDrawCircle (display_t ∗display, uint16_t x_center, uint16_t y_center, uint16_t r, uint16_t color)
- void displayDrawFillCircle (display_t ∗display, uint16_t x_center, uint16_t y_center, uint16_t r, uint16_t color)
- void displayDrawRoundRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t r, uint16_t color)
- uint16_t rgb_conv (uint16_t r, uint16_t g, uint16_t b)
- int displayDrawChar (display_t ∗display, FontxFile ∗fxs, uint16_t x, uint16_t y, uint8_t ascii, uint16_t color)
- int displayDrawString (display_t ∗display, FontxFile ∗fx, uint16_t x, uint16_t y, uint8_t ∗ascii, uint16_t color)
- void displaySetFontDirection (display_t ∗display, uint16_t dir)
- void displaySetFontFill (display_t ∗display, uint16_t color)
- void displayUnsetFontFill (display_t ∗display)
- void displaySetFontUnderLine (display_t ∗display, uint16_t color)
- void displayUnsetFontUnderLine (display_t ∗display)
- void displayBacklightOff (display_t ∗display)
- void displayBacklightOn (display_t ∗display)
- void displayInversionOff (display_t ∗display)
- void displayInversionOn (display_t ∗display)

6.17.1 Macro Definition Documentation

6.17.1.1 _DEBUG_

```
#define _DEBUG_ 0
```

Definition at line 42 of file display.c.

6.17.1.2 GPIO_MODE_OUTPUT

```
#define GPIO_MODE_OUTPUT 1
```

Definition at line 52 of file display.c.

6.17.1.3 LOG_DOMAIN

```
#define LOG_DOMAIN "display"
```

Definition at line 39 of file display.c.

6.17.1.4 M_PI

```
#define M_PI 3.14159265358979323846
```

Definition at line 44 of file display.c.

6.17.1.5 TAG

```
#define TAG "ST7789"
```

Definition at line 41 of file display.c.

6.17.2 Enumeration Type Documentation

6.17.2.1 spi_mode_t

```
enum spi_mode_t
```

Enumerator

SPI_Data_Mode	
SPI_Command_Mode	

Definition at line 50 of file display.c.

6.17.3 Function Documentation

6.17.3.1 display_destroy()

```
void display_destroy (
            display_t *display __attribute__(unused) )
```

Definition at line 306 of file display.c.

Here is the call graph for this function:

6.17.3.2 displayDrawMultiPixels()

```
void displayDrawMultiPixels (
            display_t * display,
            uint16_t x,
            uint16_t y,
            uint16_t size,
            uint16_t * colors )
```

Definition at line 336 of file display.c.

Here is the call graph for this function:

6.17.3.3 displayInit()

```
void displayInit (
            display_t * display,
            int width,
            int height,
            int offsetx,
            int offsety )
```

Definition at line 229 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.4 spi_master_init()

```
void spi_master_init (
            display_t * display )
```

Definition at line 148 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.5 spi_master_write_addr()

```
bool spi_master_write_addr (
            display_t * display,
            uint16_t addr1,
            uint16_t addr2 )
```

Definition at line 96 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.6 spi_master_write_color()

```
bool spi_master_write_color (
            display_t * display,
            uint16_t color,
            uint16_t size )
```

Definition at line 115 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.7 spi_master_write_colors()

```
bool spi_master_write_colors (
            display_t * display,
            uint16_t * colors,
            uint16_t size )
```

Definition at line 130 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.8 spi_master_write_command()

```
bool spi_master_write_command (
            display_t * display,
            uint8_t cmd )
```

Definition at line 65 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.9 spi_master_write_data_byte()

```
bool spi_master_write_data_byte (
            display_t * display,
            uint8_t data )
```

Definition at line 74 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.10 spi_master_write_data_word()

```
bool spi_master_write_data_word (
            display_t * display,
            uint16_t data )
```

Definition at line 83 of file display.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.17.3.11 spi_to_gpio()

```
gpio_level_t spi_to_gpio (
            spi_mode_t mode )
```

Definition at line 54 of file display.c.

Here is the caller graph for this function:

6.18 display.c

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2020
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023
00024 Modified by Eindhoven University of Technology 2023.
00025 */
00026 #include <arm_shared_memory_system.h>
00027 #include <display.h>
00028 #include <gpio.h>
00029 #include <lcdconfig.h>
00030 #include <log.h>
00031 #include <math.h>
00032 #include <platform.h>
00033 #include <string.h>
00034 #include <switchbox.h>
00035 #include <unistd.h>
00036 #include <util.h>
00037
00038 #undef LOG_DOMAIN
00039 #define LOG_DOMAIN "display"
00040
00041 #define TAG "ST7789"
00042 #define _DEBUG_ 0
00043
00044 #define M_PI 3.14159265358979323846
00045
00046 static arm_shared spi0_handle;
00047 static volatile uint32_t *spi0 = NULL;
00048
00049 // states that are set for usage of the DC pin in SPI
00050 typedef enum { SPI_Data_Mode = 1, SPI_Command_Mode = 0 } spi_mode_t;
00051
00052 #define GPIO_MODE_OUTPUT 1
00053
00054 gpio_level_t spi_to_gpio(spi_mode_t mode) {
00055   switch (mode) {
00056   case SPI_Data_Mode:
00057     return GPIO_LEVEL_HIGH;
00058   case SPI_Command_Mode:
00059     return GPIO_LEVEL_LOW;
00060   default:
00061     return GPIO_LEVEL_LOW;
00062   }
00063 }
00064
00065 bool spi_master_write_command(display_t *display, uint8_t cmd) {
00066   gpio_set_level(display->_dc, spi_to_gpio(SPI_Command_Mode));
00067   spi0[0x68 / 4] = cmd;
00068   while (((spi0[0x64 / 4]) & 4) == 0) {
00069   }
00070   usleep(1);
00071   return true;
00072 }
00073
00074 bool spi_master_write_data_byte(display_t *display, uint8_t data) {
00075   gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
00076
00077   spi0[0x68 / 4] = data;
00078   while (((spi0[0x64 / 4]) & 4) == 0) {
00079   }
00080   return true;
00081 }
00082
```

```
00083 bool spi_master_write_data_word(display_t *display, uint16_t data) {
00084   static uint8_t Byte[2];
00085   Byte[0] = (data >> 8) & 0xFF;
00086   Byte[1] = data & 0xFF;
00087   gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
00088   spi0[0x68 / 4] = Byte[0];
00089   spi0[0x68 / 4] = Byte[1];
00090
00091   while (((spi0[0x64 / 4]) & 4) == 0) {
00092   }
00093   return true;
00094 }
00095
00096 bool spi_master_write_addr(display_t *display, uint16_t addr1, uint16_t addr2) {
00097   static uint8_t Byte[4];
00098   Byte[0] = (addr1 >> 8) & 0xFF;
00099   Byte[1] = addr1 & 0xFF;
00100   Byte[2] = (addr2 >> 8) & 0xFF;
00101   Byte[3] = addr2 & 0xFF;
00102   gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
00103
00104   // check ordering
00105   spi0[0x68 / 4] = Byte[0];
00106   spi0[0x68 / 4] = Byte[1];
00107   spi0[0x68 / 4] = Byte[2];
00108   spi0[0x68 / 4] = Byte[3];
00109
00110   while (((spi0[0x64 / 4]) & 4) == 0) {
00111   }
00112   return true;
00113 }
00114
00115 bool spi_master_write_color(display_t *display, uint16_t color, uint16_t size) {
00116   gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
00117   for (int i = 0; i < size; i++) {
00118     while (((spi0[0x64 / 4]) & 8) == 8) {
00119     }
00120     spi0[0x68 / 4] = (color >> 8) & 0xFF;
00121     while (((spi0[0x64 / 4]) & 8) == 8) {
00122     }
00123     spi0[0x68 / 4] = (color)&0xFF;
00124   }
00125   while (((spi0[0x64 / 4]) & 4) == 0) {
00126   }
00127   return -1;
00128 }
00129
00130 bool spi_master_write_colors(display_t *display, uint16_t *colors,
00131                              uint16_t size) {
00132   gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
00133   for (int i = 0; i < size; i++) {
00134     while (((spi0[0x64 / 4]) & 8) == 8) {
00135     }
00136     spi0[0x68 / 4] = (colors[i] >> 8) & 0xFF;
00137     while (((spi0[0x64 / 4]) & 8) == 8) {
00138     }
00139     spi0[0x68 / 4] = (colors[i]) & 0xFF;
00140   }
00141   // wait till empty, then add a small extra buffer
00142   // because last byte we don't exactly know when send.
00143   while (((spi0[0x64 / 4]) & 4) == 0) {
00144   }
00145   return true;
00146 }
00147
00148 void spi_master_init(display_t *display) {
00149   // linking given pins in the switchbox
00150   switchbox_set_pin(LCD_MOSI, SWB_SPI1_MOSI);
00151   switchbox_set_pin(LCD_SCLK, SWB_SPI1_CLK);
00152   switchbox_set_pin(LCD_CS, SWB_SPI1_SS);
00153   switchbox_set_pin(LCD_DC, SWB_GPIO);
00154   switchbox_set_pin(LCD_RESET, SWB_GPIO);
00155   switchbox_set_pin(LCD_BL, SWB_GPIO);
00156
00157   // setting the appropriate direction of each protocol pin
00158   gpio_set_direction(LCD_DC, GPIO_DIR_OUTPUT);
00159   gpio_set_direction(LCD_RESET, GPIO_DIR_OUTPUT);
00160   gpio_set_direction(LCD_BL, GPIO_DIR_OUTPUT);
00161   gpio_set_level(LCD_DC, GPIO_LEVEL_LOW);
00162   gpio_set_level(LCD_RESET, GPIO_LEVEL_LOW);
00163   gpio_set_level(LCD_BL, GPIO_LEVEL_LOW);
00164
00165   // creating a shared memory instance for communicating the hardware addresses
00166   // of the linked pins
00167   spi0 = arm_shared_init(&spi0_handle, axi_quad_spi_1, 4096);
00168   if (_DEBUG_)
00169     printf("spi reset: %08X\n", spi0[0x40 / 4]);
```

```
00170    spi0[0x40 / 4] = 0x0000000a;
00171    if (_DEBUG_)
00172      printf("spi control: %08X\n", spi0[0x60 / 4]);
00173    spi0[0x60 / 4] = (1 << 4) | (1 << 3) | (1 << 2) | (1 << 1);
00174    if (_DEBUG_)
00175      printf("spi control: %08X\n", spi0[0x60 / 4]);
00176    if (_DEBUG_)
00177      printf("spi status: 08X\n", spi0[0x64 / 4]);
00178
00179    // select slave 1
00180    spi0[0x70 / 4] = 0;
00181    if (_DEBUG_)
00182      printf("spi control: %08X\n", spi0[0x60 / 4]);
00183    if (_DEBUG_)
00184      printf("testing DISPLAY\n");
00185    if (_DEBUG_)
00186      printf("LCD_CS=%d\n", LCD_CS);
00187    if (LCD_CS >= 0) {
00188      gpio_reset_pin(LCD_CS);
00189      gpio_set_direction(LCD_CS, GPIO_MODE_OUTPUT);
00190      gpio_set_level(LCD_CS, 0);
00191    }
00192
00193    if (_DEBUG_)
00194      printf("LCD_DC=%d", LCD_DC);
00195    gpio_reset_pin(LCD_DC);
00196    gpio_set_direction(LCD_DC, GPIO_MODE_OUTPUT);
00197    gpio_set_level(LCD_DC, 0);
00198    if (_DEBUG_)
00199      printf("LCD_RESET=%d", LCD_RESET);
00200
00201    if (LCD_RESET >= 0) {
00202      gpio_reset_pin(LCD_RESET);
00203      gpio_set_direction(LCD_RESET, GPIO_MODE_OUTPUT);
00204      gpio_set_level(LCD_RESET, 1);
00205      sleep_msec(100);
00206      gpio_set_level(LCD_RESET, 0);
00207      sleep_msec(500);
00208      gpio_set_level(LCD_RESET, 1);
00209      sleep_msec(300);
00210    }
00211
00212    if (_DEBUG_)
00213      printf("LCD_BL=%d", LCD_BL);
00214    if (LCD_BL >= 0) {
00215      gpio_reset_pin(LCD_BL);
00216      gpio_set_direction(LCD_BL, GPIO_MODE_OUTPUT);
00217      gpio_set_level(LCD_BL, 0);
00218    }
00219
00220    if (_DEBUG_)
00221      printf("LCD_MOSI=%d", LCD_MOSI);
00222    if (_DEBUG_)
00223      printf("LCD_SCLK=%d\n", LCD_SCLK);
00224
00225    display->_dc = LCD_DC;
00226    display->_bl = LCD_BL;
00227  }
00228
00229  void displayInit(display_t *display, int width, int height, int offsetx,
00230                   int offsety) {
00231    spi_master_init(display);
00232    display->_width = width;
00233    display->_height = height;
00234    display->_offsetx = offsetx;
00235    display->_offsety = offsety;
00236    display->_font_direction = TEXT_DIRECTION0;
00237    display->_font_fill = false;
00238    display->_font_underline = false;
00239
00240    spi_master_write_command(display, 0x01); // software Reset
00241    sleep_msec(150);
00242
00243    spi_master_write_command(display, 0x11); // sleep Out
00244    sleep_msec(255);
00245
00246    spi_master_write_command(display, 0x3A); // Interface Pixel Format
00247    spi_master_write_data_byte(display, 0x55);
00248    sleep_msec(10);
00249
00250    spi_master_write_command(display, 0x36); // Memory Data Access Control
00251    spi_master_write_data_byte(display, 0x00);
00252
00253    spi_master_write_command(display, 0x2A); // Column Address Set
00254    spi_master_write_data_byte(display, 0x00);
00255    spi_master_write_data_byte(display, 0x00);
00256    spi_master_write_data_byte(display, 0x00);
```

```
00257    spi_master_write_data_byte(display, 0xF0);
00258
00259    spi_master_write_command(display, 0x2B); // Row Address Set
00260    spi_master_write_data_byte(display, 0x00);
00261    spi_master_write_data_byte(display, 0x00);
00262    spi_master_write_data_byte(display, 0x00);
00263    spi_master_write_data_byte(display, 0xF0);
00264
00265    spi_master_write_command(display, 0x21); // Display Inversion On
00266    sleep_msec(10);
00267
00268    spi_master_write_command(display, 0x13); // Normal Display Mode On
00269    sleep_msec(10);
00270
00271    spi_master_write_command(display, 0x29); // Display ON
00272    sleep_msec(255);
00273
00274    if (display->_bl >= 0) {
00275      gpio_set_level(display->_bl, 1);
00276    }
00277 }
00278
00279 void display_set_flip(display_t *display, bool xflip, bool yflip) {
00280    if (display == NULL) {
00281      pynq_error("display_destroy: display has not been initialized\n");
00282    }
00283    if (display->_width != DISPLAY_WIDTH || display->_height != DISPLAY_HEIGHT) {
00284      pynq_error("display_destroy: internal error (wrong display hardware)\n");
00285    }
00286    spi_master_write_command(display, 0x36); // Memory Data Access Control
00287    uint8_t set = (yflip << 7) | (xflip << 6);
00288    spi_master_write_data_byte(display, set);
00289    if (yflip) {
00290      display->_offsety = 320 - display->_height;
00291    } else {
00292      display->_offsety = 0;
00293    }
00294    if (xflip) {
00295      display->_offsetx = 240 - display->_width;
00296    } else {
00297      display->_offsetx = 0;
00298    }
00299 }
00300
00301 void display_init(display_t *display) {
00302    displayInit(display, DISPLAY_WIDTH, DISPLAY_HEIGHT, 0, 0);
00303    display_set_flip(display, true, true);
00304 }
00305
00306 void display_destroy(display_t *display __attribute__((unused))) {
00307    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00308      pynq_error("display_destroy: display has not been initialized\n");
00309    }
00310    // if channel is open
00311    if (spi0 != NULL) {
00312      (void)arm_shared_close(&spi0_handle);
00313      spi0 = NULL;
00314    }
00315 }
00316
00317 void displayDrawPixel(display_t *display, uint16_t x, uint16_t y,
00318                      uint16_t color) {
00319    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00320      pynq_error("displayDrawPixel: display has not been initialized\n");
00321    }
00322    if (x >= display->_width || y >= display->_height) {
00323      pynq_error("displayDrawPixel: x=%d y=%d outside screen boundaries\n", x, y);
00324    }
00325    uint16_t _x = x + display->_offsetx;
00326    uint16_t _y = y + display->_offsety;
00327
00328    spi_master_write_command(display, 0x2A); // set column(x) address
00329    spi_master_write_addr(display, _x, _x);
00330    spi_master_write_command(display, 0x2B); // set Page(y) address
00331    spi_master_write_addr(display, _y, _y);
00332    spi_master_write_command(display, 0x2C); // memory write
00333    spi_master_write_data_word(display, color);
00334 }
00335
00336 void displayDrawMultiPixels(display_t *display, uint16_t x, uint16_t y,
00337                            uint16_t size, uint16_t *colors) {
00338    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00339      pynq_error("displayDrawMultiPixels: display has not been initialized\n");
00340    }
00341    if (x > display->_width || x + size > display->_width ||
00342        y >= display->_height) {
00343      pynq_error(
```

```
00344          "displayDrawMultiPixels: x=%d y=%d size=%d outside screen boundaries\n",
00345          x, y, size);
00346   }
00347
00348   uint16_t _x1 = x + display->_offsetx;
00349   uint16_t _x2 = _x1 + size;
00350   uint16_t _y1 = y + display->_offsety;
00351   uint16_t _y2 = _y1;
00352
00353   spi_master_write_command(display, 0x2A); // set column(x) address
00354   spi_master_write_addr(display, _x1, _x2);
00355   spi_master_write_command(display, 0x2B); // set Page(y) address
00356   spi_master_write_addr(display, _y1, _y2);
00357   spi_master_write_command(display, 0x2C); // memory write
00358   spi_master_write_colors(display, colors, size);
00359 }
00360
00361 void displayDrawFillRect(display_t *display, uint16_t x1, uint16_t y1,
00362                          uint16_t x2, uint16_t y2, uint16_t color) {
00363   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00364     pynq_error("displayDrawPixel: display has not been initialized\n");
00365   }
00366   if (x1 >= display->_width || x2 >= display->_width ||
00367       y1 >= display->_height || y2 >= display->_height) {
00368     pynq_error("displayDrawFillRect: x1=%d y1=%d x2=%d y2=%d outside screen "
00369                "boundaries\n",
00370                x1, y1, x2, y2);
00371   }
00372   // swapping points so that it is always plotted from x1 y1 bottom left, x2 y2
00373   // top right
00374   uint16_t x1_temp = x1, x2_temp = x2;
00375   uint16_t y1_temp = y1, y2_temp = y2;
00376   if (x1 > x2) {
00377     x1 = x2_temp;
00378     x2 = x1_temp;
00379   }
00380
00381   if (y1 > y2) {
00382     y1 = y2_temp;
00383     y2 = y1_temp;
00384   }
00385
00386   // printf("offset(x)=%d offset(y)=%d",display->_offsetx,display->_offsety);
00387   uint16_t _x1 = x1 + display->_offsetx;
00388   uint16_t _x2 = x2 + display->_offsetx;
00389   uint16_t _y1 = y1 + display->_offsety;
00390   uint16_t _y2 = y2 + display->_offsety;
00391
00392   spi_master_write_command(display, 0x2A); // set column(x) address
00393   spi_master_write_addr(display, _x1, _x2);
00394   spi_master_write_command(display, 0x2B); // set Page(y) address
00395   spi_master_write_addr(display, _y1, _y2);
00396   spi_master_write_command(display, 0x2C); // memory write
00397   for (int i = _x1; i <= _x2; i++) {
00398     uint16_t size = _y2 - _y1 + 1;
00399     spi_master_write_color(display, color, size);
00400   }
00401 }
00402
00403 void displayDisplayOff(display_t *display) {
00404   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00405     pynq_error("displayDisplayOff: display has not been initialized\n");
00406   }
00407   spi_master_write_command(display, 0x28); // display off
00408 }
00409
00410 void displayDisplayOn(display_t *display) {
00411   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00412     pynq_error("displayDisplayOn: display has not been initialized\n");
00413   }
00414   spi_master_write_command(display, 0x29); // display on
00415 }
00416
00417 void displayFillScreen(display_t *display, uint16_t color) {
00418   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00419     pynq_error("displayFillScreen: display has not been initialized\n");
00420   }
00421   displayDrawFillRect(display, 0, 0, display->_width - 1, display->_height - 1,
00422                       color);
00423 }
00424
00425 void displayDrawLine(display_t *display, uint16_t x1, uint16_t y1, uint16_t x2,
00426                      uint16_t y2, uint16_t color) {
00427   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00428     pynq_error("displayDrawLine: display has not been initialized\n");
00429   }
00430   if (x1 >= display->_width || y1 >= display->_height) {
```

```
00431       pynq_error("displayDrawLine: x1=%d y1=%d outside screen boundaries\n", x1,
00432                  y1);
00433   } else if (x2 >= display->_width || y2 >= display->_height) {
00434       pynq_error("displayDrawLine: x2=%d y2=%d outside screen boundaries\n", x2,
00435                  y2);
00436   }
00437   int i;
00438   int dx, dy;
00439   int sx, sy;
00440   int E;
00441
00442   /* distance between two points */
00443   dx = (x2 > x1) ? x2 - x1 : x1 - x2;
00444   dy = (y2 > y1) ? y2 - y1 : y1 - y2;
00445
00446   /* direction of two point */
00447   sx = (x2 > x1) ? 1 : -1;
00448   sy = (y2 > y1) ? 1 : -1;
00449
00450   /* inclination < 1 */
00451   if (dx > dy) {
00452     E = -dx;
00453     for (i = 0; i <= dx; i++) {
00454       displayDrawPixel(display, x1, y1, color);
00455       x1 += sx;
00456       E += 2 * dy;
00457       if (E >= 0) {
00458         y1 += sy;
00459         E -= 2 * dx;
00460       }
00461     }
00462
00463     /* inclination >= 1 */
00464   } else {
00465     E = -dy;
00466     for (i = 0; i <= dy; i++) {
00467       displayDrawPixel(display, x1, y1, color);
00468       y1 += sy;
00469       E += 2 * dx;
00470       if (E >= 0) {
00471         x1 += sx;
00472         E -= 2 * dy;
00473       }
00474     }
00475   }
00476 }
00477
00478 void displayDrawRect(display_t *display, uint16_t x1, uint16_t y1, uint16_t x2,
00479                      uint16_t y2, uint16_t color) {
00480   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00481     pynq_error("displayDrawRect: display has not been initialized\n");
00482   }
00483   if (x1 >= display->_width || y1 >= display->_height) {
00484     pynq_error("displayDrawRect: x1=%d y1=%d outside screen boundaries\n", x1,
00485                y1);
00486   } else if (x2 >= display->_width || y2 >= display->_height) {
00487     pynq_error("displayDrawRect: x2=%d y2=%d outside screen boundaries\n", x2,
00488                y2);
00489   }
00490   displayDrawLine(display, x1, y1, x2, y1, color);
00491   displayDrawLine(display, x2, y1, x2, y2, color);
00492   displayDrawLine(display, x2, y2, x1, y2, color);
00493   displayDrawLine(display, x1, y2, x1, y1, color);
00494 }
00495
00496 void displayDrawRectAngle(display_t *display, uint16_t xc, uint16_t yc,
00497                           uint16_t w, uint16_t h, uint16_t angle,
00498                           uint16_t color) {
00499   double xd, yd, rd;
00500   int x1, y1;
00501   int x2, y2;
00502   int x3, y3;
00503   int x4, y4;
00504   rd = -angle * M_PI / 180.0;
00505   xd = 0.0 - w / 2;
00506   yd = h / 2;
00507   x1 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00508   y1 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
00509
00510   yd = 0.0 - yd;
00511   x2 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00512   y2 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
00513
00514   xd = w / 2;
00515   yd = h / 2;
00516   x3 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00517   y3 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
```

```
00518
00519    yd = 0.0 - yd;
00520    x4 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00521    y4 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
00522
00523    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00524      pynq_error("displayDrawRectAngle: display has not been initialized\n");
00525    }
00526    if (x1 >= display->_width || y1 >= display->_height) {
00527      pynq_error("displayDrawRectAngle: x1=%d y1=%d outside screen boundaries\n",
00528                 x1, y1);
00529    } else if (x2 >= display->_width || y2 >= display->_height) {
00530      pynq_error("displayDrawRectAngle: x2=%d y2=%d outside screen boundaries\n",
00531                 x2, y2);
00532    } else if (x3 >= display->_width || y3 >= display->_height) {
00533      pynq_error("displayDrawRectAngle: x3=%d y3=%d outside screen boundaries\n",
00534                 x3, y3);
00535    } else if (x4 >= display->_width || y4 >= display->_height) {
00536      pynq_error("displayDrawRectAngle: x4=%d y4=%d outside screen boundaries\n",
00537                 x4, y4);
00538    }
00539
00540    displayDrawLine(display, x1, y1, x2, y2, color);
00541    displayDrawLine(display, x1, y1, x3, y3, color);
00542    displayDrawLine(display, x2, y2, x4, y4, color);
00543    displayDrawLine(display, x3, y3, x4, y4, color);
00544 }
00545
00546 // x1: First X coordinate of triangle point
00547 // y1: First Y coordinate of triangle point
00548 // x2: Second X coordinate of triangle point
00549 // y2: Second Y coordinate of triangle point
00550 // x3: Third X coordinate of triangle point
00551 // y3: Third Y coordinate of triangle point
00552 // color:color
00553 void displayDrawTriangle(display_t *display, uint16_t x1, uint16_t y1,
00554                          uint16_t x2, uint16_t y2, uint16_t x3, uint16_t y3,
00555                          uint16_t color) {
00556    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00557      pynq_error("displayDrawTriangle: display has not been initialized\n");
00558    }
00559    if (x1 >= display->_width || y1 >= display->_height) {
00560      pynq_error("displayDrawRectAngle: x1=%d y1=%d outside screen boundaries\n",
00561                 x1, y1);
00562    } else if (x2 >= display->_width || y2 >= display->_height) {
00563      pynq_error("displayDrawRectAngle: x2=%d y2=%d outside screen boundaries\n",
00564                 x2, y2);
00565    } else if (x3 >= display->_width || y3 >= display->_height) {
00566      pynq_error("displayDrawRectAngle: x3=%d y3=%d outside screen boundaries\n",
00567                 x3, y3);
00568    }
00569
00570    // draw the lines for the basic triangle
00571    displayDrawLine(display, x1, y1, x2, y2, color);
00572    displayDrawLine(display, x2, y2, x3, y3, color);
00573    displayDrawLine(display, x3, y3, x1, y1, color);
00574 }
00575
00576 // when the origin is (0, 0), the point (x1, y1) after rotating the point (x, y)
00577 // by the angle is obtained by the following calculation.
00578 //   x1 = x * cos(angle) - y * sin(angle)
00579 //   y1 = x * sin(angle) + y * cos(angle)
00580 void displayDrawTriangleCenter(display_t *display, uint16_t xc, uint16_t yc,
00581                                uint16_t w, uint16_t h, uint16_t angle,
00582                                uint16_t color) {
00583    double xd, yd, rd;
00584    int x1, y1;
00585    int x2, y2;
00586    int x3, y3;
00587    rd = -angle * M_PI / 180.0;
00588    xd = 0.0;
00589    yd = h / 2;
00590    x1 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00591    y1 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
00592
00593    xd = w / 2;
00594    yd = 0.0 - yd;
00595    x2 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00596    y2 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
00597
00598    xd = 0.0 - w / 2;
00599    x3 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
00600    y3 = (int)(xd * sin(rd) + yd * cos(rd) + yc);
00601
00602    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00603      pynq_error("displayDrawTriangleCenter: display has not been initialized\n");
00604    }
```

```
00605    if (x1 >= display->_width || y1 >= display->_height) {
00606      pynq_error("displayDrawRectAngle: x1=%d y1=%d outside screen boundaries\n",
00607               x1, y1);
00608    } else if (x2 >= display->_width || y2 >= display->_height) {
00609      pynq_error("displayDrawRectAngle: x2=%d y2=%d outside screen boundaries\n",
00610               x2, y2);
00611    } else if (x3 >= display->_width || y3 >= display->_height) {
00612      pynq_error("displayDrawRectAngle: x3=%d y3=%d outside screen boundaries\n",
00613               x3, y3);
00614    }
00615
00616    displayDrawLine(display, x1, y1, x2, y2, color);
00617    displayDrawLine(display, x1, y1, x3, y3, color);
00618    displayDrawLine(display, x2, y2, x3, y3, color);
00619 }
00620
00621 void displayDrawCircle(display_t *display, uint16_t x_center, uint16_t y_center,
00622                      uint16_t r, uint16_t color) {
00623    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00624      pynq_error("displayDrawCircle: display has not been initialized\n");
00625    }
00626    if (r == 0) {
00627      pynq_error(
00628          "displayDrawCircle: x_center=%d y_center=%d r=%d r cannot be 0\n",
00629          x_center, y_center, r);
00630    }
00631
00632    int x_max = x_center + r, x_min = x_center - r, y_max = y_center + r,
00633        y_min = y_center - r;
00634
00635    if (x_max >= display->_width || x_min < 0 || y_max >= display->_height ||
00636        y_min < 0) {
00637      pynq_error("displayDrawCircle: x_center=%d y_center=%d r=%d outside screen "
00638                 "boundaries\n",
00639                 x_center, y_center, r);
00640    }
00641
00642    int x;
00643    int y;
00644    int err;
00645    int old_err;
00646
00647    x = 0;
00648    y = -r;
00649    err = 2 - 2 * r;
00650    do {
00651      displayDrawPixel(display, x_center - x, y_center + y, color);
00652      displayDrawPixel(display, x_center - y, y_center - x, color);
00653      displayDrawPixel(display, x_center + x, y_center - y, color);
00654      displayDrawPixel(display, x_center + y, y_center + x, color);
00655      if ((old_err = err) <= x)
00656        err += ++x * 2 + 1;
00657      if (old_err > y || err > x)
00658        err += ++y * 2 + 1;
00659    } while (y < 0);
00660 }
00661
00662 void displayDrawFillCircle(display_t *display, uint16_t x_center,
00663                          uint16_t y_center, uint16_t r, uint16_t color) {
00664    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00665      pynq_error("displayDrawFillCircle: display has not been initialized\n");
00666    }
00667    if (r == 0) {
00668      pynq_error(
00669          "displayDrawFillCircle: x_center=%d y_center=%d r=%d r cannot be 0\n",
00670          x_center, y_center, r);
00671    }
00672
00673    int x_max = x_center + r, x_min = x_center - r, y_max = y_center + r,
00674        y_min = y_center - r;
00675
00676    if (x_max >= display->_width || x_min < 0 || y_max >= display->_height ||
00677        y_min < 0) {
00678      pynq_error("displayDrawFillCircle: x_center=%d y_center=%d r=%d outside "
00679                 "screen boundaries\n",
00680                 x_center, y_center, r);
00681    }
00682
00683    int x;
00684    int y;
00685    int err;
00686    int old_err;
00687    int ChangeX;
00688
00689    x = 0;
00690    y = -r;
00691    err = 2 - 2 * r;
```

```
00692   ChangeX = 1;
00693   do {
00694     if (ChangeX) {
00695       displayDrawLine(display, x_center - x, y_center - y, x_center - x,
00696                       y_center + y, color);
00697       displayDrawLine(display, x_center + x, y_center - y, x_center + x,
00698                       y_center + y, color);
00699     } // endif
00700     ChangeX = (old_err = err) <= x;
00701     if (ChangeX)
00702       err += ++x * 2 + 1;
00703     if (old_err > y || err > x)
00704       err += ++y * 2 + 1;
00705   } while (y <= 0);
00706 }
00707
00708 void displayDrawRoundRect(display_t *display, uint16_t x1, uint16_t y1,
00709                           uint16_t x2, uint16_t y2, uint16_t r,
00710                           uint16_t color) {
00711   if (display == NULL || display->_width != DISPLAY_WIDTH) {
00712     pynq_error("displayDrawRoundRect: display has not been initialized\n");
00713   }
00714   if (r == 0) {
00715     pynq_error("displayDrawRoundRect: x_center=%d x1=%d y1=%d r cannot be 0\n",
00716                x1, y1, r);
00717   } else if (x1 >= display->_width || y1 >= display->_height) {
00718     pynq_error("displayDrawRoundRect: x1=%d y1=%d outside screen boundaries\n",
00719                x1, y1);
00720   } else if (x2 >= display->_width || y2 >= display->_height) {
00721     pynq_error("displayDrawRoundRect: x2=%d y2=%d outside screen boundaries\n",
00722                x2, y2);
00723   }
00724   int x;
00725   int y;
00726   int err;
00727   int old_err;
00728   unsigned char temp;
00729
00730   if (x1 > x2) {
00731     temp = x1;
00732     x1 = x2;
00733     x2 = temp;
00734   }
00735
00736   if (y1 > y2) {
00737     temp = y1;
00738     y1 = y2;
00739     y2 = temp;
00740   }
00741
00742   if (_DEBUG_)
00743     printf("x1=%d x2=%d delta=%d r=%d", x1, x2, x2 - x1, r);
00744   if (_DEBUG_)
00745     printf("y1=%d y2=%d delta=%d r=%d", y1, y2, y2 - y1, r);
00746   if (x2 - x1 < r)
00747     return; // TODO add 20190517?
00748   if (y2 - y1 < r)
00749     return; // TODO add 20190517?
00750
00751   x = 0;
00752   y = -r;
00753   err = 2 - 2 * r;
00754
00755   do {
00756     if (x) {
00757       displayDrawPixel(display, x1 + r - x, y1 + r + y, color);
00758       displayDrawPixel(display, x2 - r + x, y1 + r + y, color);
00759       displayDrawPixel(display, x1 + r - x, y2 - r - y, color);
00760       displayDrawPixel(display, x2 - r + x, y2 - r - y, color);
00761     }
00762     if ((old_err = err) <= x)
00763       err += ++x * 2 + 1;
00764     if (old_err > y || err > x)
00765       err += ++y * 2 + 1;
00766   } while (y < 0);
00767
00768   if (_DEBUG_)
00769     printf("x1+r=%d x2-r=%d", x1 + r, x2 - r);
00770   displayDrawLine(display, x1 + r, y1, x2 - r, y1, color);
00771   displayDrawLine(display, x1 + r, y2, x2 - r, y2, color);
00772   if (_DEBUG_)
00773     printf("y1+r=%d y2-r=%d", y1 + r, y2 - r);
00774   displayDrawLine(display, x1, y1 + r, x1, y2 - r, color);
00775   displayDrawLine(display, x2, y1 + r, x2, y2 - r, color);
00776 }
00777
00778 uint16_t rgb_conv(uint16_t r, uint16_t g, uint16_t b) {
```

```
00779    return (((r & 0xF8) « 8) | ((g & 0xFC) « 3) | (b » 3));
00780 }
00781
00782 int displayDrawChar(display_t *display, FontxFile *fxs, uint16_t x, uint16_t y,
00783                      uint8_t ascii, uint16_t color) {
00784    uint16_t xx, yy, bit, ofs;
00785    unsigned char fonts[128]; // font pattern
00786    unsigned char pw, ph;
00787    int h, w;
00788    uint16_t mask;
00789    bool rc = GetFontx(fxs, ascii, fonts, &pw, &ph);
00790
00791    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00792      pynq_error("displayDrawChar: display has not been initialized\n");
00793    }
00794    if (_DEBUG_) {
00795      printf("_font_direction=%d\n", display->_font_direction);
00796      printf("GetFontx rc=%d pw=%d ph=%d\n", rc, pw, ph);
00797    }
00798
00799    if (!rc) {
00800      pynq_error("displayDrawChar: cannot get font from font file\n");
00801    }
00802
00803    switch (display->_font_direction) {
00804    case TEXT_DIRECTION0:
00805      if (x + pw >= display->_width || y + ph >= display->_height) {
00806        pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
00807                   "direction=%d outside screen boundaries\n",
00808                   x, y, ph, pw, display->_font_direction);
00809      }
00810      break;
00811    case TEXT_DIRECTION90:
00812      if (x + ph >= display->_height || y + pw >= display->_width) {
00813        pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
00814                   "direction=%d outside screen boundaries\n",
00815                   x, y, ph, pw, display->_font_direction);
00816      }
00817      break;
00818    case TEXT_DIRECTION180:
00819      if (x - pw <= 0 || y - ph <= 0) {
00820        pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
00821                   "direction=%d outside screen boundaries\n",
00822                   x, y, ph, pw, display->_font_direction);
00823      }
00824      break;
00825    case TEXT_DIRECTION270:
00826      if (x - ph <= 0 || y - pw <= 0) {
00827        pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
00828                   "direction=%d outside screen boundaries\n",
00829                   x, y, ph, pw, display->_font_direction);
00830      }
00831      break;
00832    }
00833
00834    int16_t xd1 = 0, yd1 = 0, xd2 = 0, yd2 = 0;
00835    uint16_t xss = 0, yss = 0;
00836    int16_t xsd = 0, ysd = 0, next = 0;
00837    uint16_t x0 = 0, x1 = 0, y0 = 0, y1 = 0;
00838    if (display->_font_direction == 0) {
00839      xd1 = +1;
00840      yd1 = +1; //-1;
00841      xd2 = 0;
00842      yd2 = 0;
00843      xss = x;
00844      yss = y - (ph - 1);
00845      xsd = 1;
00846      ysd = 0;
00847      next = x + pw;
00848
00849      x0 = x;
00850      y0 = y - (ph - 1);
00851      x1 = x + (pw - 1);
00852      y1 = y;
00853    } else if (display->_font_direction == 2) {
00854      xd1 = -1;
00855      yd1 = -1; //+1;
00856      xd2 = 0;
00857      yd2 = 0;
00858      xss = x;
00859      yss = y + ph + 1;
00860      xsd = 1;
00861      ysd = 0;
00862      next = x - pw;
00863
00864      x0 = x - (pw - 1);
00865      y0 = y;
```

```
00866    x1 = x;
00867    y1 = y + (ph - 1);
00868    } else if (display->_font_direction == 1) {
00869      xd1 = 0;
00870      yd1 = 0;
00871      xd2 = -1;
00872      yd2 = +1; //-1;
00873      xss = x + ph;
00874      yss = y;
00875      xsd = 0;
00876      ysd = 1;
00877      next = y + pw; // y - pw;
00878
00879      x0 = x;
00880      y0 = y;
00881      x1 = x + (ph - 1);
00882      y1 = y + (pw - 1);
00883    } else if (display->_font_direction == 3) {
00884      xd1 = 0;
00885      yd1 = 0;
00886      xd2 = +1;
00887      yd2 = -1; //+1;
00888      xss = x - (ph - 1);
00889      yss = y;
00890      xsd = 0;
00891      ysd = 1;
00892      next = y - pw; // y + pw;
00893
00894      x0 = x - (ph - 1);
00895      y0 = y - (pw - 1);
00896      x1 = x;
00897      y1 = y;
00898    }
00899
00900    // TODO: fix the problem of underflow properly some time
00901    if (display->_font_fill && x0 < DISPLAY_WIDTH && y0 < DISPLAY_HEIGHT &&
00902        x1 < DISPLAY_WIDTH && y1 < DISPLAY_HEIGHT) {
00903      displayDrawFillRect(display, x0, y0, x1, y1, display->_font_fill_color);
00904    }
00905
00906    int bits;
00907    if (_DEBUG_)
00908      printf("xss=%d yss=%d\n", xss, yss);
00909    ofs = 0;
00910    yy = yss;
00911    xx = xss;
00912    for (h = 0; h < ph; h++) {
00913      if (xsd)
00914        xx = xss;
00915      if (ysd)
00916        yy = yss;
00917      bits = pw;
00918      for (w = 0; w < ((pw + 4) / 8); w++) {
00919        mask = 0x80;
00920        for (bit = 0; bit < 8; bit++) {
00921          bits--;
00922          if (bits < 0)
00923            continue;
00924          // TODO: fix the problem of underflow properly some time
00925          if (fonts[ofs] & mask && xx < DISPLAY_WIDTH && yy < DISPLAY_HEIGHT) {
00926            displayDrawPixel(display, xx, yy, color);
00927          }
00928          // TODO: fix the problem of underflow properly some time
00929          if (h == (ph - 2) && display->_font_underline && xx < DISPLAY_WIDTH &&
00930              yy < DISPLAY_HEIGHT)
00931            displayDrawPixel(display, xx, yy, display->_font_underline_color);
00932          // TODO: fix the problem of underflow properly some time
00933          if (h == (ph - 1) && display->_font_underline && xx < DISPLAY_WIDTH &&
00934              yy < DISPLAY_HEIGHT)
00935            displayDrawPixel(display, xx, yy, display->_font_underline_color);
00936          xx = xx + xd1;
00937          yy = yy + yd2;
00938          mask = mask » 1;
00939        }
00940        ofs++;
00941      }
00942      yy = yy + yd1;
00943      xx = xx + xd2;
00944    }
00945
00946    if (next < 0)
00947      next = 0;
00948    return next;
00949 }
00950
00951 int displayDrawString(display_t *display, FontxFile *fx, uint16_t x, uint16_t y,
00952                       uint8_t *ascii, uint16_t color) {
```

```
00953    int length = strlen((char *)ascii);
00954    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00955      pynq_error("displayDrawString: display has not been initialized\n");
00956    }
00957    if (_DEBUG_)
00958      printf("displayDrawString length=%d\n", length);
00959    for (int i = 0; i < length; i++) {
00960      if (_DEBUG_)
00961        printf("ascii[%d]=%x x=%d y=%d\n", i, ascii[i], x, y);
00962      if (display->_font_direction == 0)
00963        x = displayDrawChar(display, fx, x, y, ascii[i], color);
00964      if (display->_font_direction == 1)
00965        y = displayDrawChar(display, fx, x, y, ascii[i], color);
00966      if (display->_font_direction == 2)
00967        x = displayDrawChar(display, fx, x, y, ascii[i], color);
00968      if (display->_font_direction == 3)
00969        y = displayDrawChar(display, fx, x, y, ascii[i], color);
00970    }
00971    if (display->_font_direction == 0)
00972      return x;
00973    if (display->_font_direction == 2)
00974      return x;
00975    if (display->_font_direction == 1)
00976      return y;
00977    if (display->_font_direction == 3)
00978      return y;
00979    return 0;
00980 }
00981
00982 void displaySetFontDirection(display_t *display, uint16_t dir) {
00983    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00984      pynq_error("displaySetFontDirection: display has not been initialized\n");
00985    }
00986    display->_font_direction = dir;
00987 }
00988
00989 void displaySetFontFill(display_t *display, uint16_t color) {
00990    if (display == NULL || display->_width != DISPLAY_WIDTH) {
00991      pynq_error("displaySetFontFill: display has not been initialized\n");
00992    }
00993    display->_font_fill = true;
00994    display->_font_fill_color = color;
00995 }
00996
00997 void displayUnsetFontFill(display_t *display) { display->_font_fill = false; }
00998
00999 void displaySetFontUnderLine(display_t *display, uint16_t color) {
01000    if (display == NULL || display->_width != DISPLAY_WIDTH) {
01001      pynq_error("displaySetFontUnderLine: display has not been initialized\n");
01002    }
01003    display->_font_underline = true;
01004    display->_font_underline_color = color;
01005 }
01006
01007 void displayUnsetFontUnderLine(display_t *display) {
01008    if (display == NULL || display->_width != DISPLAY_WIDTH) {
01009      pynq_error("displayUnsetFontUnderLine: display has not been initialized\n");
01010    }
01011    display->_font_underline = false;
01012 }
01013
01014 void displayBacklightOff(display_t *display) {
01015    if (display == NULL || display->_width != DISPLAY_WIDTH) {
01016      pynq_error("displayBacklightOff: display has not been initialized\n");
01017    }
01018    if (display->_bl >= 0) {
01019      gpio_set_level(display->_bl, 0);
01020    }
01021 }
01022
01023 void displayBacklightOn(display_t *display) {
01024    if (display == NULL || display->_width != DISPLAY_WIDTH) {
01025      pynq_error("displayBacklightOn: display has not been initialized\n");
01026    }
01027    if (display->_bl >= 0) {
01028      gpio_set_level(display->_bl, 1);
01029    }
01030 }
01031
01032 void displayInversionOff(display_t *display) {
01033    if (display == NULL || display->_width != DISPLAY_WIDTH) {
01034      pynq_error("displayInversionOff: display has not been initialized\n");
01035    }
01036    spi_master_write_command(display, 0x21); // display Inversion Off
01037 }
01038
01039 void displayInversionOn(display_t *display) {
```

```
01040    if (display == NULL || display->_width != DISPLAY_WIDTH) {
01041      pynq_error("displayInversionOn: display has not been initialized\n");
01042    }
01043    spi_master_write_command(display, 0x20); // display Inversion On
01044 }
```

6.19 library/display.h File Reference

```
#include <fontx.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
```
Include dependency graph for display.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct display_t

Macros

- #define DISPLAY_HEIGHT 240
- #define DISPLAY_WIDTH 240

Enumerations

- enum colors {
  RGB_RED = 0xf800 , RGB_GREEN = 0x07e0 , RGB_BLUE = 0x001f , RGB_BLACK = 0x0000 ,
  RGB_WHITE = 0xffff , RGB_GRAY = 0x8c51 , RGB_YELLOW = 0xFFE0 , RGB_CYAN = 0x07FF ,
  RGB_PURPLE = 0xF81F }
- enum directions {
  TEXT_DIRECTION0 = 0 , TEXT_DIRECTION90 = 1 , TEXT_DIRECTION180 = 2 , TEXT_DIRECTION270 =
  3 ,
  NUM_TEXT_DIRECTIONS }

Functions

- void display_init (display_t ∗display)
- void display_destroy (display_t ∗display)
- void displayDrawPixel (display_t ∗display, uint16_t x, uint16_t y, uint16_t color)
- void displayDrawFillRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayFillScreen (display_t ∗display, uint16_t color)
- void displayDrawLine (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDrawRect (display_t ∗display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void displayDrawRectAngle (display_t ∗display, uint16_t xc, uint16_t yc, uint16_t w, uint16_t h, uint16_t angle,
  uint16_t color)
- void displayDrawTriangleCenter (display_t ∗display, uint16_t xc, uint16_t yc, uint16_t w, uint16_t h, uint16_t
  angle, uint16_t color)
- void displayDrawCircle (display_t ∗display, uint16_t x_center, uint16_t y_center, uint16_t r, uint16_t color)
- void displayDrawFillCircle (display_t ∗display, uint16_t x_center, uint16_t y_center, uint16_t r, uint16_t color)

- void displayDrawRoundRect (display_t *display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t r, uint16_t color)
- uint16_t rgb_conv (uint16_t r, uint16_t g, uint16_t b)
- int displayDrawChar (display_t *display, FontxFile *fx, uint16_t x, uint16_t y, uint8_t ascii, uint16_t color)
- int displayDrawString (display_t *display, FontxFile *fx, uint16_t x, uint16_t y, uint8_t *ascii, uint16_t color)
- void displaySetFontDirection (display_t *display, uint16_t dir)
- void displaySetFontFill (display_t *display, uint16_t color)
- void displayUnsetFontFill (display_t *display)
- void displaySetFontUnderLine (display_t *display, uint16_t color)
- void displayUnsetFontUnderLine (display_t *display)
- void displayDisplayOff (display_t *display)
- void displayDisplayOn (display_t *display)
- void displayBacklightOff (display_t *display)
- void displayBacklightOn (display_t *display)
- void displayInversionOff (display_t *display)
- void displayInversionOn (display_t *display)
- void displayDrawTriangle (display_t *display, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t x3, uint16_t y3, uint16_t color)
- void display_set_flip (display_t *display, bool xflip, bool yflip)

6.20 display.h

Go to the documentation of this file.
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2020
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023
00024 Modified by Eindhoven University of Technology 2023.
00025 */
00026 #ifndef SCREEN_H
00027 #define SCREEN_H
00028
00029 #include <fontx.h>
00030 #include <stdbool.h>
00031 #include <stdint.h>
00032 #include <stdio.h>
00033 #include <string.h>
00034
00083 #define DISPLAY_HEIGHT 240
00084 #define DISPLAY_WIDTH 240
00085
00089 enum colors {
00090   RGB_RED = 0xf800,
00091   RGB_GREEN = 0x07e0,
00092   RGB_BLUE = 0x001f,
00093   RGB_BLACK = 0x0000,
00094   RGB_WHITE = 0xffff,
00095   RGB_GRAY = 0x8c51,
00096   RGB_YELLOW = 0xFFE0,
00097   RGB_CYAN = 0x07FF,
00098   RGB_PURPLE = 0xF81F
```

```
00099 };
00100
00104 enum directions {
00105   TEXT_DIRECTION0 = 0,
00106   TEXT_DIRECTION90 = 1,
00107   TEXT_DIRECTION180 = 2,
00108   TEXT_DIRECTION270 = 3,
00109   NUM_TEXT_DIRECTIONS
00110 };
00111
00116 typedef struct {
00117   uint16_t _width;
00118   uint16_t _height;
00119   uint16_t _offsetx;
00120   uint16_t _offsety;
00121   uint16_t _font_direction;
00122   uint16_t _font_fill;
00123   uint16_t _font_fill_color;
00124   uint16_t _font_underline;
00125   uint16_t _font_underline_color;
00126   int16_t _dc;
00127   int16_t _bl;
00128 } display_t;
00129
00134 extern void display_init(display_t *display);
00135
00140 extern void display_destroy(display_t *display);
00141
00149 extern void displayDrawPixel(display_t *display, uint16_t x, uint16_t y,
00150                              uint16_t color);
00151
00161 extern void displayDrawFillRect(display_t *display, uint16_t x1, uint16_t y1,
00162                                 uint16_t x2, uint16_t y2, uint16_t color);
00163
00170 extern void displayFillScreen(display_t *display, uint16_t color);
00171
00181 extern void displayDrawLine(display_t *display, uint16_t x1, uint16_t y1,
00182                             uint16_t x2, uint16_t y2, uint16_t color);
00183
00193 extern void displayDrawRect(display_t *display, uint16_t x1, uint16_t y1,
00194                             uint16_t x2, uint16_t y2, uint16_t color);
00195
00208 extern void displayDrawRectAngle(display_t *display, uint16_t xc, uint16_t yc,
00209                                  uint16_t w, uint16_t h, uint16_t angle,
00210                                  uint16_t color);
00211
00222 extern void displayDrawTriangleCenter(display_t *display, uint16_t xc,
00223                                       uint16_t yc, uint16_t w, uint16_t h,
00224                                       uint16_t angle, uint16_t color);
00225
00234 extern void displayDrawCircle(display_t *display, uint16_t x_center,
00235                               uint16_t y_center, uint16_t r, uint16_t color);
00236
00245 extern void displayDrawFillCircle(display_t *display, uint16_t x_center,
00246                                   uint16_t y_center, uint16_t r,
00247                                   uint16_t color);
00248
00259 extern void displayDrawRoundRect(display_t *display, uint16_t x1, uint16_t y1,
00260                                  uint16_t x2, uint16_t y2, uint16_t r,
00261                                  uint16_t color);
00262
00269 extern uint16_t rgb_conv(uint16_t r, uint16_t g, uint16_t b);
00270
00285 extern int displayDrawChar(display_t *display, FontxFile *fx, uint16_t x,
00286                            uint16_t y, uint8_t ascii, uint16_t color);
00287
00303 extern int displayDrawString(display_t *display, FontxFile *fx, uint16_t x,
00304                              uint16_t y, uint8_t *ascii, uint16_t color);
00305
00311 extern void displaySetFontDirection(display_t *display, uint16_t dir);
00312
00319 extern void displaySetFontFill(display_t *display, uint16_t color);
00320
00327 extern void displayUnsetFontFill(display_t *display);
00328
00336 extern void displaySetFontUnderLine(display_t *display, uint16_t color);
00337
00342 extern void displayUnsetFontUnderLine(display_t *display);
00343
00348 extern void displayDisplayOff(display_t *display);
00349
00358 extern void displayDisplayOn(display_t *display);
00359
00364 extern void displayBacklightOff(display_t *display);
00365
00370 extern void displayBacklightOn(display_t *display);
00371
```

```
00376 extern void displayInversionOff(display_t *display);
00377
00382 extern void displayInversionOn(display_t *display);
00383
00397 extern void displayDrawTriangle(display_t *display, uint16_t x1, uint16_t y1,
00398                                 uint16_t x2, uint16_t y2, uint16_t x3,
00399                                 uint16_t y3, uint16_t color);
00400
00407 void display_set_flip(display_t *display, bool xflip, bool yflip);
00412 #endif /* MAIN_ST7789_H_ */
```

6.21 library/fontx.c File Reference

```
#include "fontx.h"
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/unistd.h>
```
Include dependency graph for fontx.c:

Macros

- #define FontxDebug 0

Functions

- void AddFontx (FontxFile ∗fx, const char ∗path)
- void InitFontx (FontxFile ∗fxs, const char ∗f0, const char ∗f1)
- bool OpenFontx (FontxFile ∗fx)
- void CloseFontx (FontxFile ∗fx)
- void DumpFontx (FontxFile ∗fxs)
- uint8_t getFortWidth (FontxFile ∗fx)
- uint8_t getFortHeight (FontxFile ∗fx)
- bool GetFontx (FontxFile ∗fxs, uint8_t ascii, uint8_t ∗pGlyph, uint8_t ∗pw, uint8_t ∗ph)
- void Font2Bitmap (uint8_t ∗fonts, uint8_t ∗line, uint8_t w, uint8_t h, uint8_t inverse)
- void UnderlineBitmap (uint8_t ∗line, uint8_t w, uint8_t h)
- void ReversBitmap (uint8_t ∗line, uint8_t w, uint8_t h)
- void ShowFont (uint8_t ∗fonts, uint8_t pw, uint8_t ph)
- void ShowBitmap (uint8_t ∗bitmap, uint8_t pw, uint8_t ph)
- uint8_t RotateByte (uint8_t ch1)

6.21.1 Macro Definition Documentation

6.21.1.1 FontxDebug

```
#define FontxDebug 0
```

Definition at line 9 of file fontx.c.

6.21.2 Function Documentation

6.21.2.1 AddFontx()

```
void AddFontx (
            FontxFile * fx,
            const char * path )
```

Definition at line 11 of file fontx.c.

Here is the caller graph for this function:

6.21.2.2 getFortHeight()

```
uint8_t getFortHeight (
            FontxFile * fx )
```

Definition at line 93 of file fontx.c.

6.21.2.3 getFortWidth()

```
uint8_t getFortWidth (
            FontxFile * fx )
```

Definition at line 88 of file fontx.c.

6.22 fontx.c

Go to the documentation of this file.
```
00001 #include "fontx.h"
00002 #include <stdbool.h>
00003 #include <stdint.h>
00004 #include <stdio.h>
00005 #include <string.h>
00006 #include <sys/stat.h>
00007 #include <sys/unistd.h>
00008
00009 #define FontxDebug 0
00010
00011 void AddFontx(FontxFile *fx, const char *path) {
00012   memset(fx, 0, sizeof(FontxFile));
00013   fx->path = path;
00014   fx->opened = false;
00015 }
00016
00017 void InitFontx(FontxFile *fxs, const char *f0, const char *f1) {
00018   AddFontx(&fxs[0], f0);
00019   AddFontx(&fxs[1], f1);
00020 }
00021
00022 bool OpenFontx(FontxFile *fx) {
00023   FILE *f;
00024   if (!fx->opened) {
00025     if (FontxDebug)
00026       printf("[openFont]fx->path=[%s]\n", fx->path);
00027     f = fopen(fx->path, "r");
00028     if (FontxDebug)
00029       printf("[openFont]fopen=%p\n", f);
00030     if (f == NULL) {
00031       fx->valid = false;
00032       printf("Fontx:%s not found.\n", fx->path);
00033       return fx->valid;
```

```
00034       }
00035       fx->opened = true;
00036       fx->file = f;
00037       char buf[18];
00038       if (fread(buf, 1, sizeof(buf), fx->file) != sizeof(buf)) {
00039         fx->valid = false;
00040         printf("Fontx:%s not FONTX format.\n", fx->path);
00041         fclose(fx->file);
00042         return fx->valid;
00043       }
00044
00045       if (FontxDebug) {
00046         for (uint32_t i = 0; i < strlen(buf); i++) {
00047           printf("buf[%d]=0x%x\n", i, buf[i]);
00048         }
00049       }
00050       memcpy(fx->fxname, &buf[6], 8);
00051       fx->w = buf[14];
00052       fx->h = buf[15];
00053       fx->is_ank = (buf[16] == 0);
00054       fx->bc = buf[17];
00055       fx->fsz = (fx->w + 7) / 8 * fx->h;
00056       if (fx->fsz > FontxGlyphBufSize) {
00057         printf("Fontx:%s is too big font size.\n", fx->path);
00058         fx->valid = false;
00059         fclose(fx->file);
00060         return fx->valid;
00061       }
00062       fx->valid = true;
00063   }
00064   return fx->valid;
00065 }
00066
00067 void CloseFontx(FontxFile *fx) {
00068   if (fx->opened) {
00069     fclose(fx->file);
00070     fx->opened = false;
00071   }
00072 }
00073
00074 void DumpFontx(FontxFile *fxs) {
00075   for (int i = 0; i < 2; i++) {
00076     printf("fxs[%d]->path=%s\n", i, fxs[i].path);
00077     printf("fxs[%d]->opened=%d\n", i, fxs[i].opened);
00078     printf("fxs[%d]->fxname=%s\n", i, fxs[i].fxname);
00079     printf("fxs[%d]->valid=%d\n", i, fxs[i].valid);
00080     printf("fxs[%d]->is_ank=%d\n", i, fxs[i].is_ank);
00081     printf("fxs[%d]->w=%d\n", i, fxs[i].w);
00082     printf("fxs[%d]->h=%d\n", i, fxs[i].h);
00083     printf("fxs[%d]->fsz=%d\n", i, fxs[i].fsz);
00084     printf("fxs[%d]->bc=%d\n", i, fxs[i].bc);
00085   }
00086 }
00087
00088 uint8_t getFortWidth(FontxFile *fx) {
00089   printf("fx->w=%d\n", fx->w);
00090   return (fx->w);
00091 }
00092
00093 uint8_t getFortHeight(FontxFile *fx) {
00094   printf("fx->h=%d\n", fx->h);
00095   return (fx->h);
00096 }
00097
00098 bool GetFontx(FontxFile *fxs, uint8_t ascii, uint8_t *pGlyph, uint8_t *pw,
00099               uint8_t *ph) {
00100   int i;
00101   uint32_t offset;
00102
00103   if (FontxDebug)
00104     printf("[GetFontx]ascii=0x%x\n", ascii);
00105   for (i = 0; i < 2; i++) {
00106     if (!OpenFontx(&fxs[i]))
00107       continue;
00108     if (FontxDebug)
00109       printf("[GetFontx]openFontxFile[%d] ok\n", i);
00110
00111     if (fxs[i].is_ank) {
00112       if (FontxDebug)
00113         printf("[GetFontx]fxs.is_ank fxs.fsz=%d\n", fxs[i].fsz);
00114       offset = 17 + ascii * fxs[i].fsz;
00115       if (FontxDebug)
00116         printf("[GetFontx]offset=%d\n", offset);
00117       if (fseek(fxs[i].file, offset, SEEK_SET)) {
00118         printf("Fontx:seek(%u) failed.\n", offset);
00119         return false;
00120       }
```

```
00121       if (fread(pGlyph, 1, fxs[i].fsz, fxs[i].file) != fxs[i].fsz) {
00122         printf("Fontx:fread failed.\n");
00123         return false;
00124       }
00125       if (pw)
00126         *pw = fxs[i].w;
00127       if (ph)
00128         *ph = fxs[i].h;
00129       return true;
00130     }
00131   }
00132   return false;
00133 }
00134
00135 void Font2Bitmap(uint8_t *fonts, uint8_t *line, uint8_t w, uint8_t h,
00136                  uint8_t inverse) {
00137   int x, y;
00138   for (y = 0; y < (h / 8); y++) {
00139     for (x = 0; x < w; x++) {
00140       line[y * 32 + x] = 0;
00141     }
00142   }
00143
00144   int mask = 7;
00145   int fontp;
00146   fontp = 0;
00147   for (y = 0; y < h; y++) {
00148     for (x = 0; x < w; x++) {
00149       uint8_t d = fonts[fontp + x / 8];
00150       uint8_t linep = (y / 8) * 32 + x;
00151       if (d & (0x80 >> (x % 8)))
00152         line[linep] = line[linep] + (1 << mask);
00153     }
00154     mask--;
00155     if (mask < 0)
00156       mask = 7;
00157     fontp += (w + 7) / 8;
00158   }
00159
00160   if (inverse) {
00161     for (y = 0; y < (h / 8); y++) {
00162       for (x = 0; x < w; x++) {
00163         line[y * 32 + x] = RotateByte(line[y * 32 + x]);
00164       }
00165     }
00166   }
00167 }
00168
00169 void UnderlineBitmap(uint8_t *line, uint8_t w, uint8_t h) {
00170   int x, y;
00171   uint8_t wk;
00172   for (y = 0; y < (h / 8); y++) {
00173     for (x = 0; x < w; x++) {
00174       wk = line[y * 32 + x];
00175       if ((y + 1) == (h / 8))
00176         line[y * 32 + x] = wk + 0x80;
00177     }
00178   }
00179 }
00180
00181 void ReversBitmap(uint8_t *line, uint8_t w, uint8_t h) {
00182   int x, y;
00183   uint8_t wk;
00184   for (y = 0; y < (h / 8); y++) {
00185     for (x = 0; x < w; x++) {
00186       wk = line[y * 32 + x];
00187       line[y * 32 + x] = ~wk;
00188     }
00189   }
00190 }
00191
00192 void ShowFont(uint8_t *fonts, uint8_t pw, uint8_t ph) {
00193   int x, y, fpos;
00194   printf("[ShowFont pw=%d ph=%d]\n", pw, ph);
00195   fpos = 0;
00196   for (y = 0; y < ph; y++) {
00197     printf("%02d", y);
00198     for (x = 0; x < pw; x++) {
00199       if (fonts[fpos + x / 8] & (0x80 >> (x % 8))) {
00200         printf("*");
00201       } else {
00202         printf(".");
00203       }
00204     }
00205     printf("\n");
00206     fpos = fpos + (pw + 7) / 8;
00207   }
```

```
00208   printf("\n");
00209 }
00210
00211 void ShowBitmap(uint8_t *bitmap, uint8_t pw, uint8_t ph) {
00212   int x, y, fpos;
00213   printf("[ShowBitmap pw=%d ph=%d]\n", pw, ph);
00214
00215   fpos = 0;
00216   for (y = 0; y < ph; y++) {
00217     printf("%02d", y);
00218     for (x = 0; x < pw; x++) {
00219
00220       if (bitmap[x + (y / 8) * 32] & (0x80 >> fpos)) {
00221         printf("*");
00222       } else {
00223         printf(".");
00224       }
00225     }
00226     printf("\n");
00227     fpos++;
00228     if (fpos > 7)
00229       fpos = 0;
00230   }
00231   printf("\n");
00232 }
00233
00234 uint8_t RotateByte(uint8_t ch1) {
00235   uint8_t ch2 = 0;
00236   int j;
00237   for (j = 0; j < 8; j++) {
00238     ch2 = (ch2 << 1) + (ch1 & 0x01);
00239     ch1 = ch1 >> 1;
00240   }
00241   return ch2;
00242 }
```

6.23 library/fontx.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
```
Include dependency graph for fontx.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct FontxFile

Macros

- #define FontxGlyphBufSize (32 ∗ 32 / 8)

Typedefs

- typedef struct _IO_FILE FILE

Functions

- void AaddFontx (FontxFile ∗fx, const char ∗path)
- void InitFontx (FontxFile ∗fxs, const char ∗f0, const char ∗f1)
- bool OpenFontx (FontxFile ∗fx)
- void CloseFontx (FontxFile ∗fx)
- void DumpFontx (FontxFile ∗fxs)
- uint8_t GetFontWidth (FontxFile ∗fx)
- uint8_t GetFontHeight (FontxFile ∗fx)
- bool GetFontx (FontxFile ∗fxs, uint8_t ascii, uint8_t ∗pGlyph, uint8_t ∗pw, uint8_t ∗ph)
- void Font2Bitmap (uint8_t ∗fonts, uint8_t ∗line, uint8_t w, uint8_t h, uint8_t inverse)
- void UnderlineBitmap (uint8_t ∗line, uint8_t w, uint8_t h)
- void ReversBitmap (uint8_t ∗line, uint8_t w, uint8_t h)
- void ShowFont (uint8_t ∗fonts, uint8_t pw, uint8_t ph)
- void ShowBitmap (uint8_t ∗bitmap, uint8_t pw, uint8_t ph)
- uint8_t RotateByte (uint8_t ch)

6.23.1 Macro Definition Documentation

6.23.1.1 FontxGlyphBufSize

```
#define FontxGlyphBufSize (32 * 32 / 8)
```

Definition at line 3 of file fontx.h.

6.24 fontx.h

Go to the documentation of this file.
```
00001 #ifndef MAIN_FONTX_H_
00002 #define MAIN_FONTX_H_
00003 #define FontxGlyphBufSize (32 * 32 / 8)
00004 #include <stdbool.h>
00005 #include <stdint.h>
00006
00023 typedef struct _IO_FILE FILE;
00024
00028 typedef struct {
00029   const char *path;
00030   char fxname[10];
00031   bool opened;
00032   bool valid;
00033   bool is_ank;
00035   uint8_t w;
00036   uint8_t h;
00037   uint16_t fsz;
00038   uint8_t bc;
00039   FILE *file;
00040 } FontxFile;
00041
00048 void AaddFontx(FontxFile *fx, const char *path);
00049
00058 void InitFontx(FontxFile *fxs, const char *f0, const char *f1);
00059
00073 bool OpenFontx(FontxFile *fx);
00074
00080 void CloseFontx(FontxFile *fx);
00081
00087 void DumpFontx(FontxFile *fxs);
00088
00096 uint8_t GetFontWidth(FontxFile *fx);
00097
00105 uint8_t GetFontHeight(FontxFile *fx);
00106
00118 bool GetFontx(FontxFile *fxs, uint8_t ascii, uint8_t *pGlyph, uint8_t *pw,
00119               uint8_t *ph);
```

```
00120
00130 void Font2Bitmap(uint8_t *fonts, uint8_t *line, uint8_t w, uint8_t h,
00131                  uint8_t inverse);
00132
00140 void UnderlineBitmap(uint8_t *line, uint8_t w, uint8_t h);
00141
00149 void ReversBitmap(uint8_t *line, uint8_t w, uint8_t h);
00150
00158 void ShowFont(uint8_t *fonts, uint8_t pw, uint8_t ph);
00159
00167 void ShowBitmap(uint8_t *bitmap, uint8_t pw, uint8_t ph);
00168
00176 uint8_t RotateByte(uint8_t ch);
00177
00182 #endif
```

6.25 library/gpio.c File Reference

```
#include "gpio.h"
#include "arm_shared_memory_system.h"
#include <log.h>
#include <pinmap.h>
#include <platform.h>
#include <stdio.h>
#include <stdlib.h>
#include <version.h>
```
Include dependency graph for gpio.c:

Functions

- bool gpio_is_initialized (void)
- void gpio_init (void)
- void gpio_destroy (void)
- void gpio_reset_pin (const io_t pin)
- void gpio_reset (void)
- void gpio_set_direction (const io_t pin, const gpio_direction_t dir)
- gpio_direction_t gpio_get_direction (const io_t pin)
- void gpio_set_level (const io_t pin, const gpio_level_t level)
- gpio_level_t gpio_get_level (const io_t pin)

Variables

- volatile uint32_t ∗ gpio = NULL
- volatile uint32_t ∗ intc0 = NULL

6.25.1 Variable Documentation

6.25.1.1 gpio

```
volatile uint32_t* gpio = NULL
```

Definition at line 32 of file gpio.c.

6.25.1.2 intc0

```
volatile uint32_t* intc0 = NULL
```

Definition at line 33 of file gpio.c.

6.26 gpio.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include "gpio.h"
00023 #include "arm_shared_memory_system.h"
00024 #include <log.h>
00025 #include <pinmap.h>
00026 #include <platform.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029 #include <version.h>
00030
00031 static arm_shared gpio_handle, intc0_handle;
00032 volatile uint32_t *gpio = NULL;
00033 volatile uint32_t *intc0 = NULL;
00034
00035 bool gpio_is_initialized(void) {
00036   /* if gpio == NULL we know we are not inialized */
00037   return (gpio != NULL) ? true : false;
00038 }
00039
00040 void gpio_init(void) {
00041   pynq_info("Initialize");
00042   check_version();
00043   gpio = arm_shared_init(&gpio_handle, axi_gpio_0, 4096);
00044   intc0 = arm_shared_init(&intc0_handle, axi_intc_0, 4096);
00045 }
00046
00047 void gpio_destroy(void) {
00048   pynq_info("Destroy");
00049   arm_shared_close(&gpio_handle);
00050   arm_shared_close(&intc0_handle);
00051   gpio = NULL;
00052   intc0 = NULL;
00053 }
00054
00055 void gpio_reset_pin(const io_t pin) {
00056   PIN_CHECK(pin);
00057   pynq_info("Reset pin: %d", pin);
00058   gpio_set_direction(pin, GPIO_DIR_INPUT);
00059   gpio_set_level(pin, GPIO_LEVEL_LOW);
00060 }
00061
00062 void gpio_reset(void) {
00063   pynq_info("Reset all pins");
00064   // set all pins as input
00065   gpio[1] = 0xFFFFFFFF;
00066   // re-set all outputs to 0
00067   gpio[0] = 0x0;
00068
00069   // set all pins as input
```

```
00070    gpio[3] = 0xFFFFFFFF;
00071    // re-set all outputs to 0
00072    gpio[2] = 0x0;
00073    // disable all interrupts
00074    intc0[0] = 0;
00075    intc0[1] = 0;
00076    // remove all pending interrupts
00077    intc0[2] = 0;
00078    intc0[3] = 0;
00079 }
00080
00081 void gpio_set_direction(const io_t pin, const gpio_direction_t dir) {
00082    PIN_CHECK(pin);
00083    if (!(dir == GPIO_DIR_INPUT || dir == GPIO_DIR_OUTPUT)) {
00084      pynq_error("gpio_set_direction: invalid direction %d", dir);
00085    }
00086    int pin_bank = pin % 32;
00087    int bank = pin < 32 ? 1 : 3;
00088    if (dir == GPIO_DIR_INPUT) {
00089      gpio[bank] = gpio[bank] | (1 << pin_bank);
00090    } else {
00091      gpio[bank] = gpio[bank] & ~(1 << pin_bank);
00092    }
00093 }
00094
00095 gpio_direction_t gpio_get_direction(const io_t pin) {
00096    PIN_CHECK(pin);
00097    int pin_bank = pin % 32;
00098    int bank = pin < 32 ? 1 : 3;
00099    int dir =
00100        ((gpio[bank] & (1 << pin_bank)) != 0) ? GPIO_DIR_INPUT : GPIO_DIR_OUTPUT;
00101    return dir;
00102 }
00103
00104 void gpio_set_level(const io_t pin, const gpio_level_t level) {
00105    PIN_CHECK(pin);
00106    if (!(level == GPIO_LEVEL_HIGH || level == GPIO_LEVEL_LOW)) {
00107      pynq_error("gpio_set_level: level %d is invalid", level);
00108    }
00109    int pin_bank = pin % 32;
00110    int bank = pin < 32 ? 0 : 2;
00111    if (level == GPIO_LEVEL_HIGH) {
00112      gpio[bank] = gpio[bank] | (1 << pin_bank);
00113    } else {
00114      gpio[bank] = gpio[bank] & ~(1 << pin_bank);
00115    }
00116 }
00117
00118 gpio_level_t gpio_get_level(const io_t pin) {
00119    PIN_CHECK(pin);
00120    int pin_bank = pin % 32;
00121    int bank = pin < 32 ? 0 : 2;
00122    return (gpio[bank] & (1 << pin_bank)) != 0 ? GPIO_LEVEL_HIGH : GPIO_LEVEL_LOW;
00123 }
```

6.27 library/gpio.h File Reference

```
#include <pinmap.h>
#include <stdbool.h>
#include <stdint.h>
```
Include dependency graph for gpio.h: This graph shows which files directly or indirectly include this file:

Enumerations

- enum gpio_direction_t { GPIO_DIR_INPUT = 0 , GPIO_DIR_OUTPUT = 1 }
- enum gpio_level_t { GPIO_LEVEL_LOW = 0 , GPIO_LEVEL_HIGH = 1 }

Functions

- void gpio_init (void)
- void gpio_destroy (void)

- void gpio_reset_pin (const io_t pin)
- void gpio_set_direction (const io_t pin, const gpio_direction_t direction)
- gpio_direction_t gpio_get_direction (const io_t pin)
- void gpio_set_level (const io_t pin, const gpio_level_t level)
- gpio_level_t gpio_get_level (const io_t pin)
- void gpio_reset (void)
- bool gpio_is_initialized (void)

6.28 gpio.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef GPIO_H
00023 #define GPIO_H
00024 #include <pinmap.h>
00025 #include <stdbool.h>
00026 #include <stdint.h>
00027
00088 typedef enum {
00090   GPIO_DIR_INPUT = 0,
00092   GPIO_DIR_OUTPUT = 1
00093 } gpio_direction_t;
00094
00098 typedef enum {
00100   GPIO_LEVEL_LOW = 0,
00102   GPIO_LEVEL_HIGH = 1
00103 } gpio_level_t;
00104
00108 extern void gpio_init(void);
00113 extern void gpio_destroy(void);
00114
00121 extern void gpio_reset_pin(const io_t pin);
00122
00130 extern void gpio_set_direction(const io_t pin,
00131                                const gpio_direction_t direction);
00132
00139 extern gpio_direction_t gpio_get_direction(const io_t pin);
00140
00148 extern void gpio_set_level(const io_t pin, const gpio_level_t level);
00149
00156 extern gpio_level_t gpio_get_level(const io_t pin);
00157
00161 extern void gpio_reset(void);
00162
00168 extern bool gpio_is_initialized(void);
00172 #endif // GPIO_H
```

6.29 library/i2cps.c File Reference

```
#include "i2cps.h"
#include <fcntl.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>
```
Include dependency graph for i2cps.c:

Functions

- int setI2C (unsigned int index, long slave_addr)
- int unsetI2C (int i2c_fd)
- int writeI2C_asFile (int i2c_fd, unsigned char writebuffer[ ], unsigned char bytes)
- int readI2C_asFile (int i2c_fd, unsigned char readbuffer[ ], unsigned char bytes)

6.29.1 Function Documentation

6.29.1.1 readI2C_asFile()

```
int readI2C_asFile (
            int i2c_fd,
            unsigned char readbuffer[],
            unsigned char bytes )
```

Definition at line 88 of file i2cps.c.

Here is the caller graph for this function:

6.29.1.2 setI2C()

```
int setI2C (
            unsigned int index,
            long slave_addr )
```

Definition at line 60 of file i2cps.c.

Here is the caller graph for this function:

6.29.1.3 unsetI2C()

```
int unsetI2C (
            int i2c_fd )
```

Definition at line 74 of file i2cps.c.

Here is the caller graph for this function:

6.29.1.4 writeI2C_asFile()

```
int writeI2C_asFile (
            int i2c_fd,
            unsigned char writebuffer[],
            unsigned char bytes )
```

Definition at line 79 of file i2cps.c.

Here is the caller graph for this function:

6.30 i2cps.c

Go to the documentation of this file.
```
00001 /*******************************************************************************
00002  * Copyright (c) 2016, Xilinx, Inc.
00003  * All rights reserved.
00004  *
00005  * Redistribution and use in source and binary forms, with or without
00006  * modification, are permitted provided that the following conditions are met:
00007  *
00008  * 1.  Redistributions of source code must retain the above copyright notice,
00009  *     this list of conditions and the following disclaimer.
00010  *
00011  * 2.  Redistributions in binary form must reproduce the above copyright
00012  *     notice, this list of conditions and the following disclaimer in the
00013  *     documentation and/or other materials provided with the distribution.
00014  *
00015  * 3.  Neither the name of the copyright holder nor the names of its
00016  *     contributors may be used to endorse or promote products derived from
00017  *     this software without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00020  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00021  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
00022  * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00026  * OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00027  * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00028  * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00029  * ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  *******************************************************************************/
00032 /*******************************************************************************
00033  *
00034  *
00035  * @file i2cps.c
00036  *
00037  * Functions to interact with linux I2C. No safe checks here, so users must
00038  * know what they are doing.
00039  *
00040  * <pre>
00041  * MODIFICATION HISTORY:
00042  *
00043  * Ver   Who  Date     Changes
00044  * ----- --- ------- --------------------------------------------------
00045  * 1.00a gn  02/03/16 release
00046  * 1.00b yrq 08/31/16 add license header
00047  *
00048  * </pre>
00049  *
00050  *******************************************************************************/
00051
00052 #include "i2cps.h"
00053 #include <fcntl.h>
00054 #include <stdio.h>
00055 #include <stdlib.h>
00056 #include <string.h>
00057 #include <sys/ioctl.h>
00058 #include <unistd.h>
00059
00060 int setI2C(unsigned int index, long slave_addr) {
00061   int i2c_fd;
00062   char buf[50];
```

```
00063   sprintf(buf, "/dev/i2c-%d", index);
00064   // printf("buf = %s \n",buf);
00065   if ((i2c_fd = open(buf, O_RDWR)) < 0) {
00066     return -1;
00067   }
00068   if (ioctl(i2c_fd, I2C_SLAVE, slave_addr) < 0) {
00069     return -1;
00070   }
00071   return i2c_fd;
00072 }
00073
00074 int unsetI2C(int i2c_fd) {
00075   close(i2c_fd);
00076   return 0;
00077 }
00078
00079 int writeI2C_asFile(int i2c_fd, unsigned char writebuffer[],
00080                     unsigned char bytes) {
00081   unsigned char bytesWritten = write(i2c_fd, writebuffer, bytes);
00082   if (bytes != bytesWritten) {
00083     return -1;
00084   }
00085   return 0;
00086 }
00087
00088 int readI2C_asFile(int i2c_fd, unsigned char readbuffer[],
00089                    unsigned char bytes) {
00090   unsigned char bytesRead = read(i2c_fd, readbuffer, bytes);
00091   if (bytes != bytesRead)
00092     return -1;
00093   return 0;
00094 }
```

6.31 library/i2cps.h File Reference

```
#include <linux/i2c-dev.h>
```
Include dependency graph for i2cps.h: This graph shows which files directly or indirectly include this file:

Macros

- #define writeI2C_byte(i2c_fd, u8RegAddr, u8Data) i2c_smbus_write_byte_data(i2c_fd, u8RegAddr, u8↩Data);
- #define writeI2C_word(i2c_fd, u8RegAddr, u16Data) i2c_smbus_write_word_data(i2c_fd, u8RegAddr, u16↩Data);

Functions

- int setI2C (unsigned int index, long slave_addr)
- int unsetI2C (int i2c_fd)
- int writeI2C_asFile (int i2c_fd, unsigned char writebuffer[ ], unsigned char bytes)
- int readI2C_asFile (int i2c_fd, unsigned char readbuffer[ ], unsigned char bytes)

6.31.1 Detailed Description

Functions to interact with linux I2C.

```
MODIFICATION HISTORY:

Ver    Who      Date      Changes
-----  --------  --------  ----------------------------------------------
1.00a gn        01/24/15 First release
1.00b yrq       08/31/16 Added license header
```

Definition in file i2cps.h.

6.31.2 Macro Definition Documentation

6.31.2.1 writeI2C_byte

```
#define writeI2C_byte(
            i2c_fd,
            u8RegAddr,
            u8Data )  i2c_smbus_write_byte_data(i2c_fd, u8RegAddr, u8Data);
```

Definition at line 63 of file i2cps.h.

6.31.2.2 writeI2C_word

```
#define writeI2C_word(
            i2c_fd,
            u8RegAddr,
            u16Data )  i2c_smbus_write_word_data(i2c_fd, u8RegAddr, u16Data);
```

Definition at line 66 of file i2cps.h.

6.31.3 Function Documentation

6.31.3.1 readI2C_asFile()

```
int readI2C_asFile (
            int i2c_fd,
            unsigned char readbuffer[],
            unsigned char bytes )
```

Definition at line 88 of file i2cps.c.

Here is the caller graph for this function:

6.31.3.2 setI2C()

```
int setI2C (
            unsigned int index,
            long slave_addr )
```

Definition at line 60 of file i2cps.c.

Here is the caller graph for this function:

6.31.3.3 unsetI2C()

```
int unsetI2C (
            int i2c_fd )
```

Definition at line 74 of file i2cps.c.

Here is the caller graph for this function:

6.31.3.4 writeI2C_asFile()

```
int writeI2C_asFile (
            int i2c_fd,
            unsigned char writebuffer[],
            unsigned char bytes )
```

Definition at line 79 of file i2cps.c.

Here is the caller graph for this function:

6.32   i2cps.h

Go to the documentation of this file.
```
00001 /*******************************************************************************
00002  *  Copyright (c) 2016, Xilinx, Inc.
00003  *  All rights reserved.
00004  *
00005  *  Redistribution and use in source and binary forms, with or without
00006  *  modification, are permitted provided that the following conditions are met:
00007  *
00008  *  1.  Redistributions of source code must retain the above copyright notice,
00009  *      this list of conditions and the following disclaimer.
00010  *
00011  *  2.  Redistributions in binary form must reproduce the above copyright
00012  *      notice, this list of conditions and the following disclaimer in the
00013  *      documentation and/or other materials provided with the distribution.
00014  *
00015  *  3.  Neither the name of the copyright holder nor the names of its
00016  *      contributors may be used to endorse or promote products derived from
00017  *      this software without specific prior written permission.
00018  *
00019  *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00020  *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00021  *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
00022  *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
00023  *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00026  *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00027  *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00028  *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00029  *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  *******************************************************************************/
00032
00033 /*******************************************************************************/
00052 #ifndef __I2CPS_H__
00053 #define __I2CPS_H__
00054
00055 #include <linux/i2c-dev.h>
00056
00057 int setI2C(unsigned int index, long slave_addr);
00058 int unsetI2C(int i2c_fd);
00059 int writeI2C_asFile(int i2c_fd, unsigned char writebuffer[],
00060                     unsigned char bytes);
00061 int readI2C_asFile(int i2c_fd, unsigned char readbuffer[], unsigned char bytes);
00062
00063 #define writeI2C_byte(i2c_fd, u8RegAddr, u8Data)                            \
00064   i2c_smbus_write_byte_data(i2c_fd, u8RegAddr, u8Data);
00065
00066 #define writeI2C_word(i2c_fd, u8RegAddr, u16Data)                           \
00067   i2c_smbus_write_word_data(i2c_fd, u8RegAddr, u16Data);
00068
00069 #endif // __I2CPS_H__
```

6.33   library/iic.c File Reference

```
#include "iic.h"
#include "arm_shared_memory_system.h"
```

```
#include "log.h"
#include <platform.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <xiic_l.h>
#include <unistd.h>
```
Include dependency graph for iic.c:

Data Structures

- struct IICHandle

Macros

- #define IIC_TIMEOUT 5
- #define IIC_STOP 0x00
- #define IIC_REPEATED_START 0x01
- #define IIC_DGIER_OFFSET 0x1C
- #define IIC_IISR_OFFSET 0x20
- #define IIC_IIER_OFFSET 0x28
- #define IIC_RESETR_OFFSET 0x40
- #define IIC_CR_REG_OFFSET 0x100
- #define IIC_SR_REG_OFFSET 0x104
- #define IIC_DTR_REG_OFFSET 0x108
- #define IIC_DRR_REG_OFFSET 0x10C
- #define IIC_ADR_REG_OFFSET 0x110
- #define IIC_TFO_REG_OFFSET 0x114
- #define IIC_RFO_REG_OFFSET 0x118
- #define IIC_TBA_REG_OFFSET 0x11C
- #define IIC_RFD_REG_OFFSET 0x120
- #define IIC_GPO_REG_OFFSET 0x124
- #define IIC_CR_ENABLE_DEVICE_MASK 0x00000001
- #define IIC_CR_TX_FIFO_RESET_MASK 0x00000002
- #define IIC_CR_MSMS_MASK 0x00000004
- #define IIC_CR_DIR_IS_TX_MASK 0x00000008
- #define IIC_CR_NO_ACK_MASK 0x00000010
- #define IIC_CR_REPEATED_START_MASK 0x00000020
- #define IIC_CR_GENERAL_CALL_MASK 0x00000040
- #define IIC_INTR_ARB_LOST_MASK 0x00000001
- #define IIC_INTR_TX_ERROR_MASK 0x00000002
- #define IIC_INTR_TX_EMPTY_MASK 0x00000004
- #define IIC_INTR_RX_FULL_MASK 0x00000008
- #define IIC_INTR_BNB_MASK 0x00000010
- #define IIC_INTR_AAS_MASK 0x00000020
- #define IIC_INTR_NAAS_MASK 0x00000040
- #define IIC_INTR_TX_HALF_MASK 0x00000080
- #define IIC_SR_BUS_BUSY_MASK 0x00000004
- #define IIC_SR_RX_FIFO_EMPTY 0x00000040
- #define IIC_REG_SOFT_RESET (0x40)
- #define IIC_SR_MSTR_RDING_SLAVE_MASK 0x00000008

Typedefs

- typedef struct IICHandle IICHandle

Enumerations

- enum IICState { IIC_IDLE = 0 , IIC_ADDRESS = 1 , IIC_READ = 2 , IIC_WRITE = 3 }

Functions

- void iic_init (const iic_index_t iic)
- void iic_destroy (const iic_index_t iic)
- bool iic_set_slave_mode (const iic_index_t iic, const uint8_t addr, uint32_t ∗register_map, const uint32_t rm_length)
- void iic_slave_mode_handler (const iic_index_t iic)
- void iic_reset (const iic_index_t iic)
- bool iic_read_register (const iic_index_t iic, const uint8_t addr, const uint8_t reg, uint8_t ∗data, uint16_↩ t data_length)
- bool iic_write_register (const iic_index_t iic, const uint8_t addr, const uint8_t reg, uint8_t ∗data, uint16_t data_length)

6.33.1 Macro Definition Documentation

6.33.1.1 IIC_ADR_REG_OFFSET

```
#define IIC_ADR_REG_OFFSET 0x110
```

Address Register

Definition at line 80 of file iic.c.

6.33.1.2 IIC_CR_DIR_IS_TX_MASK

```
#define IIC_CR_DIR_IS_TX_MASK 0x00000008
```

Dir of Tx. Txing=1

Definition at line 90 of file iic.c.

6.33.1.3 IIC_CR_ENABLE_DEVICE_MASK

```
#define IIC_CR_ENABLE_DEVICE_MASK 0x00000001
```

Device enable = 1

Definition at line 87 of file iic.c.

6.33.1.4 IIC_CR_GENERAL_CALL_MASK

`#define IIC_CR_GENERAL_CALL_MASK 0x00000040`

Gen Call enabled = 1

Definition at line 93 of file iic.c.

6.33.1.5 IIC_CR_MSMS_MASK

`#define IIC_CR_MSMS_MASK 0x00000004`

Master starts Txing=1

Definition at line 89 of file iic.c.

6.33.1.6 IIC_CR_NO_ACK_MASK

`#define IIC_CR_NO_ACK_MASK 0x00000010`

Tx Ack. NO ack = 1

Definition at line 91 of file iic.c.

6.33.1.7 IIC_CR_REG_OFFSET

`#define IIC_CR_REG_OFFSET 0x100`

Control Register

Definition at line 76 of file iic.c.

6.33.1.8 IIC_CR_REPEATED_START_MASK

`#define IIC_CR_REPEATED_START_MASK 0x00000020`

Repeated start = 1

Definition at line 92 of file iic.c.

6.33.1.9 IIC_CR_TX_FIFO_RESET_MASK

`#define IIC_CR_TX_FIFO_RESET_MASK 0x00000002`

Transmit FIFO reset=1

Definition at line 88 of file iic.c.

6.33.1.10 IIC_DGIER_OFFSET

`#define IIC_DGIER_OFFSET 0x1C`

Global Interrupt Enable Register

Definition at line 72 of file iic.c.

6.33.1.11 IIC_DRR_REG_OFFSET

`#define IIC_DRR_REG_OFFSET 0x10C`

Data Rx Register

Definition at line 79 of file iic.c.

6.33.1.12 IIC_DTR_REG_OFFSET

`#define IIC_DTR_REG_OFFSET 0x108`

Data Tx Register

Definition at line 78 of file iic.c.

6.33.1.13 IIC_GPO_REG_OFFSET

`#define IIC_GPO_REG_OFFSET 0x124`

Output Register

Definition at line 85 of file iic.c.

6.33.1.14 IIC_IIER_OFFSET

`#define IIC_IIER_OFFSET 0x28`

Interrupt Enable Register

Definition at line 74 of file iic.c.

6.33.1.15 IIC_IISR_OFFSET

`#define IIC_IISR_OFFSET 0x20`

Interrupt Status Register

Definition at line 73 of file iic.c.

6.33.1.16 IIC_INTR_AAS_MASK

```
#define IIC_INTR_AAS_MASK 0x00000020
```

1 = When addr as slave

Definition at line 100 of file iic.c.

6.33.1.17 IIC_INTR_ARB_LOST_MASK

```
#define IIC_INTR_ARB_LOST_MASK 0x00000001
```

1 = Arbitration lost

Definition at line 95 of file iic.c.

6.33.1.18 IIC_INTR_BNB_MASK

```
#define IIC_INTR_BNB_MASK 0x00000010
```

1 = Bus not busy

Definition at line 99 of file iic.c.

6.33.1.19 IIC_INTR_NAAS_MASK

```
#define IIC_INTR_NAAS_MASK 0x00000040
```

1 = Not addr as slave

Definition at line 101 of file iic.c.

6.33.1.20 IIC_INTR_RX_FULL_MASK

```
#define IIC_INTR_RX_FULL_MASK 0x00000008
```

1 = Rx FIFO/reg=OCY level

Definition at line 98 of file iic.c.

6.33.1.21 IIC_INTR_TX_EMPTY_MASK

```
#define IIC_INTR_TX_EMPTY_MASK 0x00000004
```

1 = Tx FIFO/reg empty

Definition at line 97 of file iic.c.

6.33.1.22 IIC_INTR_TX_ERROR_MASK

`#define IIC_INTR_TX_ERROR_MASK 0x00000002`

1 = Tx error/msg complete

Definition at line 96 of file iic.c.

6.33.1.23 IIC_INTR_TX_HALF_MASK

`#define IIC_INTR_TX_HALF_MASK 0x00000080`

1 = Tx FIFO half empty

Definition at line 102 of file iic.c.

6.33.1.24 IIC_REG_SOFT_RESET

`#define IIC_REG_SOFT_RESET (0x40)`

Definition at line 105 of file iic.c.

6.33.1.25 IIC_REPEATED_START

`#define IIC_REPEATED_START 0x01`

Definition at line 70 of file iic.c.

6.33.1.26 IIC_RESETR_OFFSET

`#define IIC_RESETR_OFFSET 0x40`

Reset Register

Definition at line 75 of file iic.c.

6.33.1.27 IIC_RFD_REG_OFFSET

`#define IIC_RFD_REG_OFFSET 0x120`

Rx FIFO Depth reg

Definition at line 84 of file iic.c.

6.33.1.28 IIC_RFO_REG_OFFSET

```
#define IIC_RFO_REG_OFFSET 0x118
```

Rx FIFO Occupancy

Definition at line 82 of file iic.c.

6.33.1.29 IIC_SR_BUS_BUSY_MASK

```
#define IIC_SR_BUS_BUSY_MASK 0x00000004
```

1 = Bus is busy

Definition at line 103 of file iic.c.

6.33.1.30 IIC_SR_MSTR_RDING_SLAVE_MASK

```
#define IIC_SR_MSTR_RDING_SLAVE_MASK 0x00000008
```

Definition at line 106 of file iic.c.

6.33.1.31 IIC_SR_REG_OFFSET

```
#define IIC_SR_REG_OFFSET 0x104
```

Status Register

Definition at line 77 of file iic.c.

6.33.1.32 IIC_SR_RX_FIFO_EMPTY

```
#define IIC_SR_RX_FIFO_EMPTY 0x00000040
```

Definition at line 104 of file iic.c.

6.33.1.33 IIC_STOP

```
#define IIC_STOP 0x00
```

Definition at line 69 of file iic.c.

6.33.1.34 IIC_TBA_REG_OFFSET

```
#define IIC_TBA_REG_OFFSET 0x11C
```

10 Bit Address reg

Definition at line 83 of file iic.c.

6.33.1.35 IIC_TFO_REG_OFFSET

```
#define IIC_TFO_REG_OFFSET 0x114
```

Tx FIFO Occupancy

Definition at line 81 of file iic.c.

6.33.1.36 IIC_TIMEOUT

```
#define IIC_TIMEOUT 5
```

Definition at line 33 of file iic.c.

6.33.2 Typedef Documentation

6.33.2.1 IICHandle

```
typedef struct IICHandle IICHandle
```

6.33.3 Enumeration Type Documentation

6.33.3.1 IICState

```
enum IICState
```

Enumerator

IIC_IDLE	
IIC_ADDRESS	
IIC_READ	
IIC_WRITE	

Definition at line 34 of file iic.c.

6.34 iic.c

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
```

```
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include "iic.h"
00023 #include "arm_shared_memory_system.h"
00024 #include "log.h"
00025 #include <platform.h>
00026 #include <stdio.h>
00027 #include <string.h>
00028 #include <time.h>
00029 #include <xiic_l.h>
00030
00031 #include <unistd.h>
00032
00033 #define IIC_TIMEOUT 5
00034 typedef enum {
00035   IIC_IDLE = 0,
00036   IIC_ADDRESS = 1,
00037   IIC_READ = 2,
00038   IIC_WRITE = 3
00039
00040 } IICState;
00041
00042 typedef struct IICHandle {
00043   arm_shared mem_handle;
00044   volatile uint32_t *ptr;
00045
00046   // Register interface for slave mode.
00047   uint32_t *register_map;
00048   uint32_t register_map_length;
00049
00050   uint8_t saddr;
00051   uint32_t selected_register;
00052   uint32_t new_val;
00053   uint32_t recv_cnt;
00054   IICState state;
00055   int addressed;
00056 } IICHandle;
00057
00058 static IICHandle iic_handles[NUM_IICS] = {
00059     {.ptr = NULL,
00060
00061      .saddr = 0,
00062      .register_map = NULL,
00063      .register_map_length = 0,
00064      .selected_register = 0,
00065      .state = IIC_IDLE,
00066      .addressed = 0},
00067 };
00068
00069 #define IIC_STOP 0x00
00070 #define IIC_REPEATED_START 0x01
00071
00072 #define IIC_DGIER_OFFSET 0x1C
00073 #define IIC_IISR_OFFSET 0x20
00074 #define IIC_IIER_OFFSET 0x28
00075 #define IIC_RESETR_OFFSET 0x40
00076 #define IIC_CR_REG_OFFSET 0x100
00077 #define IIC_SR_REG_OFFSET 0x104
00078 #define IIC_DTR_REG_OFFSET 0x108
00079 #define IIC_DRR_REG_OFFSET 0x10C
00080 #define IIC_ADR_REG_OFFSET 0x110
00081 #define IIC_TFO_REG_OFFSET 0x114
00082 #define IIC_RFO_REG_OFFSET 0x118
00083 #define IIC_TBA_REG_OFFSET 0x11C
00084 #define IIC_RFD_REG_OFFSET 0x120
00085 #define IIC_GPO_REG_OFFSET 0x124
00087 #define IIC_CR_ENABLE_DEVICE_MASK 0x00000001
00088 #define IIC_CR_TX_FIFO_RESET_MASK 0x00000002
00089 #define IIC_CR_MSMS_MASK 0x00000004
00090 #define IIC_CR_DIR_IS_TX_MASK 0x00000008
00091 #define IIC_CR_NO_ACK_MASK 0x00000010
00092 #define IIC_CR_REPEATED_START_MASK 0x00000020
00093 #define IIC_CR_GENERAL_CALL_MASK 0x00000040
00094 #define IIC_INTR_ARB_LOST_MASK 0x00000001
00095 #define IIC_INTR_ARB_LOST_MASK 0x00000001
00096 #define IIC_INTR_TX_ERROR_MASK 0x00000002
00097 #define IIC_INTR_TX_EMPTY_MASK 0x00000004
00098 #define IIC_INTR_RX_FULL_MASK 0x00000008
00099 #define IIC_INTR_BNB_MASK 0x00000010
00100 #define IIC_INTR_AAS_MASK 0x00000020
```

```
00101 #define IIC_INTR_NAAS_MASK 0x00000040
00102 #define IIC_INTR_TX_HALF_MASK 0x00000080
00103 #define IIC_SR_BUS_BUSY_MASK 0x00000004
00104 #define IIC_SR_RX_FIFO_EMPTY 0x00000040
00105 #define IIC_REG_SOFT_RESET (0x40)
00106 #define IIC_SR_MSTR_RDING_SLAVE_MASK 0x00000008
00107
00108 void iic_init(const iic_index_t iic) {
00109   if (!(iic >= IIC0 && iic < NUM_IICS)) {
00110     pynq_error("invalid IIC %d, must be 0..%d\n", iic, NUM_IICS);
00111   }
00112   if (iic == IIC0) {
00113     iic_handles[iic].ptr =
00114         arm_shared_init(&((iic_handles[iic].mem_handle)), axi_iic_0, 4096);
00115   } else if (iic == IIC1) {
00116     iic_handles[iic].ptr =
00117         arm_shared_init(&((iic_handles[iic].mem_handle)), axi_iic_1, 4096);
00118   }
00119   // Reset
00120   (iic_handles[iic].ptr[IIC_REG_SOFT_RESET / 4]) = 0xA;
00121   usleep(1000);
00122 }
00123
00124 void iic_destroy(const iic_index_t iic) {
00125   if (!(iic >= IIC0 && iic < NUM_IICS)) {
00126     pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
00127   }
00128   if (iic_handles[iic].ptr == NULL) {
00129     pynq_error("IIC%d has not been initialized.\n", iic);
00130   }
00131   arm_shared_close(&((iic_handles[iic].mem_handle)));
00132   iic_handles[iic].ptr = NULL;
00133 }
00134
00135 bool iic_set_slave_mode(const iic_index_t iic, const uint8_t addr,
00136                         uint32_t *register_map, const uint32_t rm_length) {
00137   if (!(iic >= IIC0 && iic < NUM_IICS)) {
00138     pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
00139   }
00140   if (iic_handles[iic].ptr == NULL) {
00141     pynq_error("IIC%d has not been initialized.\n", iic);
00142   }
00143   (iic_handles[iic].saddr) = addr;
00144   (iic_handles[iic].ptr[IIC_ADR_REG_OFFSET / 4]) = addr << 1;
00145   uint32_t ctr_reg = (iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4]);
00146   // Clear the master bit.
00147   ctr_reg &= ~(IIC_CR_MSMS_MASK);
00148   // Enable IIC
00149   ctr_reg |= IIC_CR_ENABLE_DEVICE_MASK;
00150
00151   (iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4]) = ctr_reg;
00152   (iic_handles[iic].ptr[IIC_RFD_REG_OFFSET / 4]) = 0x0;
00153
00154   iic_handles[iic].register_map = register_map;
00155   iic_handles[iic].register_map_length = rm_length;
00156
00157   return true;
00158 }
00159
00160 static inline void iic_clear_isr_mask(const iic_index_t iic, uint32_t mask) {
00161
00162   (iic_handles[iic].ptr[IIC_IISR_OFFSET / 4]) =
00163       (iic_handles[iic].ptr[IIC_IISR_OFFSET / 4]) & mask;
00164 }
00165
00166 static void iic_flush_tx_fifo(const iic_index_t iic) {
00167   IICHandle *handle = &(iic_handles[iic]);
00168   uint32_t reg = handle->ptr[IIC_CR_REG_OFFSET / 4];
00169   handle->ptr[IIC_CR_REG_OFFSET / 4] = reg | IIC_CR_TX_FIFO_RESET_MASK;
00170   handle->ptr[IIC_CR_REG_OFFSET / 4] = reg;
00171 }
00172
00173 static void iic_tx_error_handler(const iic_index_t iic) {
00174   IICHandle *handle = &(iic_handles[iic]);
00175   iic_flush_tx_fifo(iic);
00176   iic_clear_isr_mask(iic, IIC_INTR_RX_FULL_MASK | IIC_INTR_TX_HALF_MASK |
00177                           IIC_INTR_TX_ERROR_MASK | IIC_INTR_TX_EMPTY_MASK);
00178
00179   uint32_t reg = handle->ptr[IIC_CR_REG_OFFSET / 4];
00180   handle->ptr[IIC_CR_REG_OFFSET / 4] = reg & ~IIC_CR_MSMS_MASK;
00181 }
00182 static void iic_slave_master_write(const iic_index_t iic, const uint32_t c) {
00183   IICHandle *handle = &(iic_handles[iic]);
00184   uint32_t v = (c << (handle->recv_cnt) * 8);
00185   handle->new_val |= v;
00186   handle->recv_cnt++;
00187   // If we have one full word, write it back to register.
```

```
00188    if (handle->recv_cnt == 4) {
00189      handle->register_map[handle->selected_register %
00190                         handle->register_map_length] = handle->new_val;
00191      // go to idle mode.
00192      handle->state = IIC_IDLE;
00193    }
00194 }
00195
00196 static void iic_slave_master_read(const iic_index_t iic) {
00197    IICHandle *handle = &(iic_handles[iic]);
00198    if (handle->state == IIC_ADDRESS) {
00199      handle->state = IIC_WRITE;
00200    }
00201    if (handle->state == IIC_WRITE) {
00202      uint32_t r = (handle->register_map[handle->selected_register %
00203                                    handle->register_map_length]);
00204      uint8_t c = (r » ((handle->recv_cnt) * 8)) & 0xFF;
00205      (iic_handles[iic].ptr[IIC_DTR_REG_OFFSET / 4]) = c;
00206      handle->recv_cnt++;
00207      if (handle->recv_cnt == 4) {
00208        // printf("1\n");
00209        handle->state = IIC_IDLE;
00210      }
00211      // modulo 4;
00212      handle->recv_cnt &= 0x03;
00213    }
00214 };
00215 static void iic_interrupt_handle(const iic_index_t iic) {
00216    time_t start = time(NULL);
00217    IICHandle *handle = &(iic_handles[iic]);
00218    int loop = 1;
00219    uint32_t sr_reg = (handle->ptr[IIC_SR_REG_OFFSET / 4]);
00220    do {
00221      time_t now = time(NULL);
00222      uint32_t nisr = (handle->ptr[IIC_IISR_OFFSET / 4]);
00223      uint32_t clear = 0;
00224      uint32_t isr = 0;
00225      isr = nisr;
00226      if (isr & IIC_INTR_ARB_LOST_MASK) {
00227
00228        clear = IIC_INTR_ARB_LOST_MASK;
00229      } else if (isr & IIC_INTR_TX_ERROR_MASK) {
00230        iic_tx_error_handler(iic);
00231        handle->state = IIC_IDLE;
00232        clear = IIC_INTR_TX_ERROR_MASK;
00233      } else if (isr & IIC_INTR_RX_FULL_MASK) {
00234        // if there is data in outgoing fifo, flush this.
00235        uint8_t d = handle->ptr[IIC_DRR_REG_OFFSET / 4];
00236
00237        uint32_t reg = handle->ptr[IIC_CR_REG_OFFSET / 4];
00238        reg &= ~IIC_CR_NO_ACK_MASK;
00239        handle->ptr[IIC_CR_REG_OFFSET / 4] = reg;
00240        switch (handle->state) {
00241        case IIC_IDLE:
00242          handle->recv_cnt = 0;
00243          handle->new_val = 0;
00244          handle->selected_register = d;
00245          handle->state = IIC_ADDRESS;
00246          break;
00247        case IIC_ADDRESS:
00248          handle->state = IIC_WRITE;
00249          // FALLTHROUGH
00250        case IIC_WRITE:
00251          iic_slave_master_write(iic, d);
00252          start = now;
00253          break;
00254        default:
00255          pynq_warning("unhandled");
00256          break;
00257        }
00258
00259        clear = IIC_INTR_RX_FULL_MASK;
00260      } else if (handle->addressed && (isr & IIC_INTR_NAAS_MASK)) {
00261        handle->addressed = 0;
00262
00263        clear = IIC_INTR_NAAS_MASK;
00264      } else if (!handle->addressed && (isr & IIC_INTR_AAS_MASK)) {
00265        handle->addressed = 1;
00266        clear = IIC_INTR_AAS_MASK;
00267      } else if (isr & IIC_INTR_BNB_MASK) {
00268        loop = 0;
00269
00270        clear = IIC_INTR_BNB_MASK;
00271      } else if (isr & (IIC_INTR_TX_EMPTY_MASK | IIC_INTR_TX_HALF_MASK)) {
00272
00273        if (handle->state == IIC_ADDRESS || handle->state == IIC_WRITE) {
00274          if (sr_reg & IIC_SR_MSTR_RDING_SLAVE_MASK) {
```

```
00275              iic_slave_master_read(iic);
00276              start = now;
00277            }
00278          }
00279          clear = isr & (IIC_INTR_TX_EMPTY_MASK | IIC_INTR_TX_HALF_MASK);
00280        }
00281
00282      if ((now - start) > IIC_TIMEOUT) {
00283        pynq_warning("IIC timeout, resetting bus.");
00284        iic_reset(iic);
00285        iic_clear_isr_mask(iic, 0xFF);
00286        uint32_t ctr_reg = (handle->ptr[IIC_CR_REG_OFFSET / 4]);
00287        (iic_handles[iic].ptr[IIC_ADR_REG_OFFSET / 4]) = handle->saddr << 1;
00288        // Clear the master bit.
00289        ctr_reg &= ~(IIC_CR_MSMS_MASK);
00290        // Enable IIC
00291        ctr_reg |= IIC_CR_ENABLE_DEVICE_MASK;
00292
00293        (handle->ptr[IIC_CR_REG_OFFSET / 4]) = ctr_reg;
00294        loop = 0;
00295      }
00296      //(iic_handles[iic].ptr[IIC_IISR_OFFSET / 4]) = nisr;
00297      iic_clear_isr_mask(iic, clear);
00298      sr_reg = (handle->ptr[IIC_SR_REG_OFFSET / 4]);
00299    } while (loop && (sr_reg & IIC_SR_BUS_BUSY_MASK));
00300    // iic_clear_isr_mask(iic, 0xFF);
00301 }
00302 void iic_slave_mode_handler(const iic_index_t iic) {
00303
00304    if (!(iic >= IIC0 && iic < NUM_IICS)) {
00305      pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
00306    }
00307    if (iic_handles[iic].ptr == NULL) {
00308      pynq_error("IIC%d has not been initialized.\n", iic);
00309    }
00310    iic_interrupt_handle(iic);
00311    return;
00312 }
00313
00314 void iic_reset(const iic_index_t iic) {
00315    if (!(iic >= IIC0 && iic < NUM_IICS)) {
00316      pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
00317    }
00318    if (iic_handles[iic].ptr == NULL) {
00319      pynq_error("IIC%d has not been initialized.\n", iic);
00320    }
00321    iic_handles[iic].ptr[IIC_REG_SOFT_RESET / 4] = 0x0A;
00322    uint32_t reg = iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4];
00323    iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4] =
00324        reg & ~IIC_CR_REPEATED_START_MASK;
00325 }
00326
00327 bool iic_read_register(const iic_index_t iic, const uint8_t addr,
00328                        const uint8_t reg, uint8_t *data, uint16_t data_length) {
00329    if (!(iic >= IIC0 && iic < NUM_IICS)) {
00330      pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
00331    }
00332    if (iic_handles[iic].ptr == NULL) {
00333      pynq_error("IIC%d has not been initialized.\n", iic);
00334    }
00335    if (XIic_Send((UINTPTR)iic_handles[iic].ptr, addr, (u8 *)&reg, 1,
00336                  XIIC_REPEATED_START) != 1) {
00337      return 1;
00338    }
00339    uint8_t ByteCount = XIic_Recv((UINTPTR)iic_handles[iic].ptr, addr, data,
00340                                  data_length, XIIC_STOP);
00341    return (ByteCount == data_length) ? 0 : 1;
00342 }
00343
00344 bool iic_write_register(const iic_index_t iic, const uint8_t addr,
00345                         const uint8_t reg, uint8_t *data,
00346                         uint16_t data_length) {
00347    if (!(iic >= IIC0 && iic < NUM_IICS)) {
00348      pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
00349    }
00350    if (iic_handles[iic].ptr == NULL) {
00351      pynq_error("IIC%d has not been initialized.\n", iic);
00352    }
00353    uint8_t buffer[1 + data_length];
00354    buffer[0] = reg;
00355    memcpy(&(buffer[1]), data, data_length);
00356    uint8_t ByteCount = XIic_Send((UINTPTR)iic_handles[iic].ptr, addr,
00357                                  &(buffer[0]), 1 + data_length, XIIC_STOP);
00358    return (ByteCount == (data_length + 1)) ? 0 : 1;
00359 }
```

6.35 library/iic.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
```
Include dependency graph for iic.h: This graph shows which files directly or indirectly include this file:

Enumerations

- enum iic_index_t { IIC0 = 0 , IIC1 = 1 , NUM_IICS = 2 }

Functions

- void iic_init (const iic_index_t iic)
- void iic_destroy (const iic_index_t iic)
- bool iic_read_register (const iic_index_t iic, const uint8_t addr, const uint8_t reg, uint8_t ∗data, uint16_↩
  t length)
- bool iic_write_register (const iic_index_t iic, const uint8_t addr, const uint8_t reg, uint8_t ∗data, uint16_t
  length)
- bool iic_set_slave_mode (const iic_index_t iic, const uint8_t addr, uint32_t ∗register_map, const uint32_t
  rm_length)
- void iic_slave_mode_handler (const iic_index_t iic)
- void iic_reset (const iic_index_t iic)

6.36 iic.h

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef IIC_H
00023 #define IIC_H
00024 #include <stdbool.h>
00025 #include <stdint.h>
00026
00104 typedef enum { IIC0 = 0, IIC1 = 1, NUM_IICS = 2 } iic_index_t;
00105
00113 extern void iic_init(const iic_index_t iic);
00114
00120 extern void iic_destroy(const iic_index_t iic);
00121
00133 extern bool iic_read_register(const iic_index_t iic, const uint8_t addr,
00134                               const uint8_t reg, uint8_t *data,
00135                               uint16_t length);
00136
00148 extern bool iic_write_register(const iic_index_t iic, const uint8_t addr,
00149                                const uint8_t reg, uint8_t *data,
00150                                uint16_t length);
```

```
00151
00152 extern bool iic_set_slave_mode(const iic_index_t iic, const uint8_t addr,
00153                                uint32_t *register_map,
00154                                const uint32_t rm_length);
00155
00162 extern void iic_slave_mode_handler(const iic_index_t iic);
00163
00169 extern void iic_reset(const iic_index_t iic);
00173 #endif
```

6.37 library/interrupt.c File Reference

```
#include "arm_shared_memory_system.h"
#include <fcntl.h>
#include <gpio.h>
#include <log.h>
#include <platform.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <util.h>
```
Include dependency graph for interrupt.c:

Macros

- #define DOMAIN "Interrupt"

Functions

- void check_initialization (void)
- int gpio_interrupt_init (void)
- void gpio_enable_interrupt (const io_t pin)
- void gpio_disable_interrupt (const io_t pin)
- void gpio_disable_all_interrupts (void)
- uint64_t gpio_get_interrupt (void)
- void gpio_ack_interrupt (void)
- void verify_interrupt_request (const io_t pin)
- void gpio_print_interrupt (void)
- void findSetBitPositions (uint64_t word, uint8_t ∗positions)
- void gpio_wait_for_interrupt (const io_t pin)
- uint8_t ∗ gpio_get_interrupt_pins (uint8_t ∗positions)

Variables

- uint32_t ∗ gpio
- uint32_t ∗ intc0

6.37.1 Macro Definition Documentation

6.37.1.1 DOMAIN

```
#define DOMAIN "Interrupt"
```

Definition at line 34 of file interrupt.c.

6.37.2 Function Documentation

6.37.2.1 check_initialization()

```
void check_initialization (
            void )
```

Definition at line 41 of file interrupt.c.

Here is the caller graph for this function:

6.37.2.2 findSetBitPositions()

```
void findSetBitPositions (
            uint64_t word,
            uint8_t * positions )
```

Definition at line 126 of file interrupt.c.

Here is the caller graph for this function:

6.37.3 Variable Documentation

6.37.3.1 gpio

```
uint32_t* gpio  [extern]
```

Definition at line 32 of file gpio.c.

6.37.3.2 intc0

```
uint32_t* intc0  [extern]
```

Definition at line 33 of file gpio.c.

6.38   interrupt.c

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include "arm_shared_memory_system.h"
00023 #include <fcntl.h>
00024 #include <gpio.h>
00025 #include <log.h>
00026 #include <platform.h>
00027 #include <stdbool.h>
00028 #include <stdint.h>
00029 #include <stdio.h>
00030 #include <stdlib.h>
00031 #include <unistd.h>
00032 #include <util.h>
00033
00034 #define DOMAIN "Interrupt"
00035
00036 extern uint32_t *gpio;
00037 extern uint32_t *intc0;
00038
00039 static bool gpio_initialized = false;
00040
00041 void check_initialization(void) {
00042   if (gpio_initialized == false) {
00043     pynq_error("Interrupts have not been initialized. Call "
00044               "gpio_interupt_init() first.\n");
00045   }
00046 }
00047
00048 int gpio_interrupt_init(void) {
00049   int fd = open("/dev/uio1", O_RDWR, O_CLOEXEC);
00050   if (fd < 0) {
00051     pynq_error("failed to open interrupts\n");
00052   }
00053   int32_t m = 1;
00054   write(fd, &m, 4);
00055   gpio_initialized = true;
00056   return fd;
00057 }
00058
00059 void gpio_enable_interrupt(const io_t pin) {
00060   check_initialization();
00061   int pin_bank = pin % 32;
00062   int bank = pin < 32 ? 0 : 1;
00063   if (bank == 0) {
00064     printf("interrupt set 0: %08X %08X\r\n", pin, pin_bank);
00065     intc0[0] |= (1 << pin_bank);
00066   } else {
00067     printf("interrupt set 1: %08X %08X\r\n", pin, pin_bank);
00068     intc0[1] |= (1 << (pin_bank));
00069   }
00070 }
00071
00072 void gpio_disable_interrupt(const io_t pin) {
00073   check_initialization();
00074   intc0[0] &= ~(1 << pin);
00075 }
00076
00077 void gpio_disable_all_interrupts(void) {
00078   check_initialization();
00079   intc0[0] = 0;
00080   intc0[1] = 0;
00081 }
00082
```

```
00083 uint64_t gpio_get_interrupt(void) {
00084   check_initialization();
00085   uint64_t retv = intc0[3];
00086   retv <<= 32;
00087   retv |= intc0[2];
00088   return retv;
00089 }
00090
00091 void gpio_ack_interrupt(void) {
00092   check_initialization();
00093   intc0[2] = 1;
00094 }
00095
00096 void verify_interrupt_request(const io_t pin) {
00097   // TODO check if interrupts are initialized when using other interrupt
00098   // functions
00099   uint64_t retv = intc0[1];
00100   retv <<= 32;
00101   retv |= intc0[0];
00102   if (pin < 64) {
00103     uint64_t bitMask = 1ULL << pin;
00104     if (!(bitMask & retv)) {
00105       pynq_error("Pin %d is not enabled. Enable by using "
00106                  "gpio_enable_interrupt(pin). \n",
00107                  pin);
00108     }
00109   } else {
00110     if (retv == 0) {
00111       pynq_error("No interrupts enabled. Enable by using "
00112                  "gpio_enable_interrupt(pin). \n");
00113     }
00114   }
00115 }
00116
00117 void gpio_print_interrupt(void) {
00118   check_initialization();
00119   //  printf("11c: %08X\r\n", gpio[0x11c / 4]);
00120   //  printf("128: %08X\r\n", gpio[0x128 / 4]);
00121   //  printf("120: %08X\r\n", gpio[0x120 / 4]);
00122   printf("interrupt 0: %08X %08X\r\n", intc0[0], intc0[2]);
00123   printf("interrupt 1: %08X %08X\r\n", intc0[1], intc0[3]);
00124 }
00125
00126 void findSetBitPositions(uint64_t word, uint8_t *positions) {
00127   int index = 0;
00128   int count = 0;
00129   while (word) {
00130     if (word & 1) {
00131       positions[count++] = index;
00132     }
00133     word >>= 1;
00134     index++;
00135   }
00136 }
00137
00138 void gpio_wait_for_interrupt(const io_t pin) {
00139   check_initialization();
00140   verify_interrupt_request(pin);
00141   if (pin > 63) {
00142     while (1) {
00143       uint64_t interrupt = gpio_get_interrupt();
00144       if (interrupt != 0) {
00145         break;
00146       }
00147     }
00148   } else {
00149     while (1) {
00150       uint64_t interrupt = gpio_get_interrupt();
00151       uint64_t bitMask = 1ULL << pin;
00152       if (bitMask & interrupt) {
00153         break;
00154       }
00155       sleep_msec(100);
00156     }
00157   }
00158 }
00159
00160 uint8_t *gpio_get_interrupt_pins(uint8_t *positions) {
00161   check_initialization();
00162   verify_interrupt_request(64); // check if any interupt pin is enabled
00163   // uint8_t *positions = (uint8_t *)malloc(64 * sizeof(uint8_t));
00164   uint64_t pin = (uint64_t)((uint64_t)(intc0[3]) << 32 | intc0[2]);
00165   findSetBitPositions(pin, positions);
00166   // printf("Interrupted pin(s): ");
00167   bool empty = true;
00168   for (int i = 0; i < 64; i++) {
00169     if (positions[i] != 0) {
```

```
00170      empty = false;
00171      // printf("%d ", positions[i]);
00172      break;
00173    }
00174  }
00175  if (empty) {
00176    printf("WARNING: gpio_get_interrupt_pins: No pins interrupted. ");
00177  }
00178  printf("\n");
00179  return (positions);
00180 }
```

6.39   library/interrupt.h File Reference

`#include <gpio.h>`
Include dependency graph for interrupt.h: This graph shows which files directly or indirectly include this file:

Functions

- int gpio_interrupt_init (void)
- void gpio_ack_interrupt (void)
- void verify_interrupt_request (const io_t pin)
- void gpio_print_interrupt (void)
- void gpio_enable_interrupt (const io_t pin)
- void gpio_disable_interrupt (const io_t pin)
- void gpio_disable_all_interrupts (void)
- uint64_t gpio_get_interrupt (void)
- uint8_t ∗ gpio_get_interrupt_pins (uint8_t ∗positions)
- void gpio_wait_for_interrupt (const io_t pin)

6.40   interrupt.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef INTERRUPT_H
00023 #define INTERRUPT_H
00024
00025 #include <gpio.h>
00026
00072 extern int gpio_interrupt_init(void);
00073
00079 extern void gpio_ack_interrupt(void);
00080
00089 extern void verify_interrupt_request(const io_t pin);
00090
```

```
00094 extern void gpio_print_interrupt(void);
00095
00101 extern void gpio_enable_interrupt(const io_t pin);
00102
00109 extern void gpio_disable_interrupt(const io_t pin);
00110
00114 extern void gpio_disable_all_interrupts(void);
00115
00121 extern uint64_t gpio_get_interrupt(void);
00122
00129 extern uint8_t *gpio_get_interrupt_pins(uint8_t *positions);
00130
00137 extern void gpio_wait_for_interrupt(const io_t pin);
00138
00142 #endif
```

6.41 library/leds.c File Reference

```
#include <gpio.h>
#include <leds.h>
#include <log.h>
#include <pinmap.h>
#include <pwm.h>
#include <stdio.h>
#include <stdlib.h>
```
Include dependency graph for leds.c:

Macros

- #define LOG_DOMAIN "leds"

Typedefs

- typedef enum _led_mode led_mode

Enumerations

- enum _led_mode { uninitialized , binary , pwm_green , pwm_color }

Functions

- void leds_init_onoff (void)
- void green_leds_init_pwm (void)
- void color_leds_init_pwm (void)
- void leds_destroy (void)
- void green_led_onoff (const int led, const int onoff)
- void green_led_on (const int led)
- void green_led_off (const int led)
- void color_led_red_onoff (const int onoff)
- void color_led_green_onoff (const int onoff)
- void color_led_blue_onoff (const int onoff)
- void color_led_onoff (const int red_onoff, const int green_onoff, const int blue_onoff)
- void color_led_on (void)
- void color_led_off (void)

6.41.1 Macro Definition Documentation

6.41.1.1 LOG_DOMAIN

```
#define LOG_DOMAIN "leds"
```

Definition at line 31 of file leds.c.

6.41.2 Typedef Documentation

6.41.2.1 led_mode

```
typedef enum _led_mode led_mode
```

6.41.3 Enumeration Type Documentation

6.41.3.1 _led_mode

```
enum _led_mode
```

Enumerator

uninitialized	
binary	
pwm_green	
pwm_color	

Definition at line 33 of file leds.c.

6.42 leds.c

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <gpio.h>
00023 #include <leds.h>
```

```
00024 #include <log.h>
00025 #include <pinmap.h>
00026 #include <pwm.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029
00030 #undef LOG_DOMAIN
00031 #define LOG_DOMAIN "leds"
00032
00033 typedef enum _led_mode { uninitialized, binary, pwm_green, pwm_color } led_mode;
00034 static led_mode mode = uninitialized;
00035
00036 // LEDs are either on or off
00037 void leds_init_onoff(void) {
00038   if (mode == binary)
00039     return;
00040   if (mode != uninitialized) {
00041     pynq_error("leds_init_onoff: mode=%d should be uninitialized\n", mode);
00042   }
00043   gpio_set_direction(IO_LD0, GPIO_DIR_OUTPUT);
00044   gpio_set_direction(IO_LD1, GPIO_DIR_OUTPUT);
00045   gpio_set_direction(IO_LD2, GPIO_DIR_OUTPUT);
00046   gpio_set_direction(IO_LD3, GPIO_DIR_OUTPUT);
00047   gpio_set_direction(IO_LD4B, GPIO_DIR_OUTPUT);
00048   gpio_set_direction(IO_LD4G, GPIO_DIR_OUTPUT);
00049   gpio_set_direction(IO_LD4R, GPIO_DIR_OUTPUT);
00050   gpio_set_direction(IO_LD5B, GPIO_DIR_OUTPUT);
00051   gpio_set_direction(IO_LD5G, GPIO_DIR_OUTPUT);
00052   gpio_set_direction(IO_LD5R, GPIO_DIR_OUTPUT);
00053   mode = binary;
00054 }
00055
00056 // can change the intensity of LEDs, the onoff parameters are then in the range
00057 // 0..255
00058 void green_leds_init_pwm(void) {
00059   if (mode == pwm_green)
00060     return;
00061   if (mode != uninitialized) {
00062     pynq_error("green_leds_init_pwm: mode=%d should be uninitialized\n", mode);
00063   }
00064   // initialize switchbox and routing PWM to LEDs
00065   switchbox_set_pin(IO_LD0, SWB_PWM0);
00066   switchbox_set_pin(IO_LD1, SWB_PWM1);
00067   switchbox_set_pin(IO_LD2, SWB_PWM2);
00068   switchbox_set_pin(IO_LD3, SWB_PWM3);
00069   // initialize the PWM channels
00070   pwm_init(PWM0, 256);
00071   pwm_init(PWM1, 256);
00072   pwm_init(PWM2, 256);
00073   pwm_init(PWM3, 256);
00074   mode = pwm_green;
00075 }
00076
00077 // can change the intensity of LEDs, the onoff parameters are then in the range
00078 // 0..255
00079 void color_leds_init_pwm(void) {
00080   if (mode == pwm_color)
00081     return;
00082   if (mode != uninitialized) {
00083     pynq_error("color_leds_init_pwm: mode=%d should be uninitialized\n", mode);
00084   }
00085   // initialize switchbox and routing PWM to LEDs
00086   switchbox_set_pin(IO_LD4R, SWB_PWM0);
00087   switchbox_set_pin(IO_LD4G, SWB_PWM1);
00088   switchbox_set_pin(IO_LD4B, SWB_PWM2);
00089   // initialize the PWM channels
00090   pwm_init(PWM0, 256);
00091   pwm_init(PWM1, 256);
00092   pwm_init(PWM2, 256);
00093   mode = pwm_color;
00094 }
00095
00096 void leds_destroy(void) {
00097   // note that pynq_destroy will also reset all GPIO and switch off all LEDs
00098   if (mode == binary) {
00099     for (int i = 0; i < NUM_GREEN_LEDS; i++)
00100       green_led_off(i);
00101   }
00102   if (mode == pwm_green || mode == pwm_color) {
00103     green_led_off(0);
00104     green_led_off(1);
00105     green_led_off(2);
00106     pwm_destroy(PWM0);
00107     pwm_destroy(PWM1);
00108     pwm_destroy(PWM2);
00109   }
00110   if (mode == pwm_green) {
```

```
00111       green_led_off(3);
00112       pwm_destroy(PWM3);
00113     }
00114   mode = uninitialized;
00115 }
00116
00117 void green_led_onoff(const int led, const int onoff) {
00118   if (led < 0 || led >= NUM_GREEN_LEDS) {
00119     pynq_error("green_led_onoff: invalid led=%d, must be 0..%d-1\n",
00120               NUM_GREEN_LEDS);
00121   }
00122   int oo = onoff;
00123   switch (mode) {
00124   case binary:
00125     gpio_set_level(IO_LD0 + led,
00126                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
00127     break;
00128   case pwm_green:
00129   case pwm_color:
00130     if (onoff < 0) {
00131       oo = 0;
00132     } else {
00133       if (onoff > 255) {
00134         oo = 255;
00135       }
00136     }
00137     pwm_set_duty_cycle(PWM0 + led, oo);
00138     break;
00139   default:
00140     pynq_error("green_led_onoff: LEDs have not been initialized with "
00141               "green_leds_init_pwm\n");
00142     break;
00143   }
00144 }
00145
00146 void green_led_on(const int led) { green_led_onoff(led, LED_ON); }
00147 void green_led_off(const int led) { green_led_onoff(led, LED_OFF); }
00148 void color_led_red_onoff(const int onoff) {
00149   int oo = onoff;
00150   switch (mode) {
00151   case binary:
00152     gpio_set_level(IO_LD4R,
00153                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
00154     break;
00155   case pwm_green:
00156   case pwm_color:
00157     if (onoff < 0) {
00158       oo = 0;
00159     } else {
00160       if (onoff > 255) {
00161         oo = 255;
00162       }
00163     }
00164     pwm_set_duty_cycle(PWM0, oo);
00165     break;
00166   default:
00167     pynq_error("color_led_red_onoff: LEDs have not been initialized with "
00168               "color_leds_init_pwm\n");
00169   }
00170 }
00171
00172 void color_led_green_onoff(const int onoff) {
00173   int oo = onoff;
00174   switch (mode) {
00175   case binary:
00176     gpio_set_level(IO_LD4G,
00177                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
00178     break;
00179   case pwm_color:
00180     if (onoff < 0) {
00181       oo = 0;
00182     } else {
00183       if (onoff > 255) {
00184         oo = 255;
00185       }
00186     }
00187     pwm_set_duty_cycle(PWM1, oo);
00188     break;
00189   default:
00190     pynq_error("color_led_green_onoff: LEDs have not been initialized with "
00191               "color_leds_init_pwm\n");
00192   }
00193 }
00194
00195 void color_led_blue_onoff(const int onoff) {
00196   int oo = onoff;
00197   switch (mode) {
```

```
00198    case binary:
00199      gpio_set_level(IO_LD4B,
00200                     (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
00201      break;
00202    case pwm_color:
00203      if (onoff < 0) {
00204        oo = 0;
00205      } else {
00206        if (onoff > 255) {
00207          oo = 255;
00208        }
00209      }
00210      pwm_set_duty_cycle(PWM2, oo);
00211      break;
00212    default:
00213      pynq_error("color_led_blue_onoff: LEDs have not been initialized with "
00214                 "color_leds_init_pwm\n");
00215    }
00216 }
00217
00218 void color_led_onoff(const int red_onoff, const int green_onoff,
00219                      const int blue_onoff) {
00220    color_led_red_onoff(red_onoff);
00221    color_led_green_onoff(green_onoff);
00222    color_led_blue_onoff(blue_onoff);
00223 }
00224
00225 void color_led_on(void) { color_led_onoff(LED_ON, LED_ON, LED_ON); }
00226 void color_led_off(void) { color_led_onoff(LED_OFF, LED_OFF, LED_OFF); }
```

6.43 library/leds.h File Reference

#include <gpio.h>
#include <pinmap.h>
Include dependency graph for leds.h: This graph shows which files directly or indirectly include this file:

Macros

- #define NUM_LED_COLORS 3 /∗ # colors per color LED (RGB) ∗/
- #define NUM_LEDS (NUM_GREEN_LEDS + NUM_COLOR_LEDS)
- #define LED_OFF 0
- #define LED_ON 255

Enumerations

- enum green_led_index_t {
  LED0 , LED1 , LED2 , LED3 ,
  NUM_GREEN_LEDS }
- enum color_led_index_t { COLOR_LED0 , COLOR_LED1 , NUM_COLOR_LEDS }

Functions

- void leds_init_onoff (void)
- void green_leds_init_pwm (void)
- void color_leds_init_pwm (void)
- void leds_destroy (void)
- void green_led_onoff (const int led, const int onoff)
- void green_led_on (const int led)
- void green_led_off (const int led)
- void color_led_red_onoff (const int onoff)
- void color_led_green_onoff (const int onoff)
- void color_led_blue_onoff (const int onoff)
- void color_led_onoff (const int red_onoff, const int green_onoff, const int blue_onoff)
- void color_led_on (void)
- void color_led_off (void)

6.44   leds.h

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef LED_H
00023 #define LED_H
00024
00025 #include <gpio.h>
00026 #include <pinmap.h>
00027
00080 typedef enum {
00081   LED0,
00082   LED1,
00083   LED2,
00084   LED3,
00085   NUM_GREEN_LEDS,
00086 } green_led_index_t;
00087
00094 typedef enum {
00095   COLOR_LED0,
00096   COLOR_LED1,
00097   NUM_COLOR_LEDS,
00098 } color_led_index_t;
00099
00100 #define NUM_LED_COLORS 3 /* # colors per color LED (RGB) */
00101 #define NUM_LEDS (NUM_GREEN_LEDS + NUM_COLOR_LEDS)
00102 #define LED_OFF 0
00103 #define LED_ON 255
00104
00109 extern void leds_init_onoff(void);
00110
00116 extern void green_leds_init_pwm(void);
00117
00123 extern void color_leds_init_pwm(void);
00124
00129 extern void leds_destroy(void);
00130
00139 extern void green_led_onoff(const int led, const int onoff);
00140
00148 extern void green_led_on(const int led);
00149
00157 extern void green_led_off(const int led);
00158
00166 extern void color_led_red_onoff(const int onoff);
00167
00175 extern void color_led_green_onoff(const int onoff);
00176
00184 extern void color_led_blue_onoff(const int onoff);
00185
00194 extern void color_led_onoff(const int red_onoff, const int green_onoff,
00195                             const int blue_onoff);
00196
00203 extern void color_led_on(void);
00204
00211 extern void color_led_off(void);
00212
00217 #endif
```

6.45 library/libpynq.c File Reference

```
#include "libpynq.h"
```
Include dependency graph for libpynq.c:

Functions

- void pynq_init (void)
- void pynq_destroy (void)

6.45.1 Function Documentation

6.45.1.1 pynq_destroy()

```
void pynq_destroy (
            void )
```

Reset and destroy the switchbox and GPIO of the PYNQ.

Definition at line 35 of file libpynq.c.

Here is the call graph for this function:

6.45.1.2 pynq_init()

```
void pynq_init (
            void )
```

Initialise the switchbox and GPIO of the PYNQ.

Definition at line 24 of file libpynq.c.

Here is the call graph for this function:

6.46 libpynq.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
```

```
00020 SOFTWARE.
00021 */
00022 #include "libpynq.h"
00023
00024 void pynq_init(void) {
00025   gpio_init();
00026   gpio_reset();
00027   switchbox_init();
00028   switchbox_reset();
00029
00030   // set line buffering on the output, should help with logging
00031   //  setlinebuf(stdout);
00032   //  setlinebuf(stderr);
00033 }
00034
00035 void pynq_destroy(void) {
00036   gpio_reset();
00037   gpio_destroy();
00038   switchbox_reset();
00039   switchbox_destroy();
00040 }
```

6.47 library/libpynq.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <adc.h>
#include <arm_shared_memory_system.h>
#include <audio.h>
#include <buttons.h>
#include <display.h>
#include <fontx.h>
#include <gpio.h>
#include <i2cps.h>
#include <iic.h>
#include <interrupt.h>
#include <leds.h>
#include <log.h>
#include <pinmap.h>
#include <pwm.h>
#include <switchbox.h>
#include <uart.h>
#include <uio.h>
#include <util.h>
#include <version.h>
#include <lcdconfig.h>
#include <platform.h>
```
Include dependency graph for libpynq.h: This graph shows which files directly or indirectly include this file:

Functions

- void pynq_init (void)
- void pynq_destroy (void)

6.47.1 Function Documentation

6.47.1.1 pynq_destroy()

```
void pynq_destroy (
        void ) [extern]
```

Reset and destroy the switchbox and GPIO of the PYNQ.

Definition at line 35 of file libpynq.c.

Here is the call graph for this function:

6.47.1.2 pynq_init()

```
void pynq_init (
            void ) [extern]
```

Initialise the switchbox and GPIO of the PYNQ.

Definition at line 24 of file libpynq.c.

Here is the call graph for this function:

6.48 libpynq.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef PYNQLIB_H
00023 #define PYNQLIB_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif // all of your legacy C code here
00028
00029 // standard libraries
00030 #include <stdbool.h>
00031 #include <stdint.h>
00032
00033 // library > (...)
00034 #include <adc.h>
00035 #include <arm_shared_memory_system.h>
00036 #include <audio.h>
00037 #include <buttons.h>
00038 #include <display.h>
00039 #include <fontx.h>
00040 #include <gpio.h>
00041 #include <i2cps.h>
00042 #include <iic.h>
00043 #include <interrupt.h>
00044 #include <leds.h>
00045 #include <log.h>
00046 #include <pinmap.h>
00047 #include <pwm.h>
00048 #include <switchbox.h>
00049 #include <uart.h>
00050 #include <uio.h>
00051 #include <util.h>
```

```
00052 #include <version.h>
00053
00054 // platform > (...)
00055 #include <lcdconfig.h>
00056 #include <platform.h>
00057
00061 extern void pynq_init(void);
00062
00066 extern void pynq_destroy(void);
00067
00068 #ifdef __cplusplus
00069 }
00070 #endif
00071
00072 #endif
```

6.49 library/log.c File Reference

```
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "log.h"
```
Include dependency graph for log.c:

Macros

- #define DOMAIN "LOGGER"

Functions

- void pynq_log (const LogLevel level, char const ∗domain, char const ∗location, unsigned int lineno, char const ∗fmt,...)

6.49.1 Macro Definition Documentation

6.49.1.1 DOMAIN

```
#define DOMAIN "LOGGER"
```

Logging domain for this file.

Definition at line 31 of file log.c.

6.50 log.c

```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <stdarg.h>
00023 #include <stdbool.h>
00024 #include <stdio.h>
00025 #include <stdlib.h>
00026 #include <string.h>
00027
00028 #include "log.h"
00029
00031 #define DOMAIN "LOGGER"
00032
00034 static const char color_escape_calls[NUM_LOG_LEVELS][8] = {
00036     "\033[1;32m",
00038     "\033[1;33m",
00040     "\033[1;31m"};
00042 static const char log_level_name[NUM_LOG_LEVELS][10] = {
00043     "INFO:   ", "WARNING: ", "ERROR:   "};
00045 static const char color_escape_blue[] = "\033[1;34m";
00046 static const char color_escape_reset[] = "\033[0m";
00047
00048 static bool pynq_log_init = false;
00049 static LogLevel critical_level = LOG_LEVEL_ERROR;
00050 static LogLevel min_log_level = LOG_LEVEL_WARNING;
00051
00052 void pynq_log(const LogLevel level, char const *domain, char const *location,
00053               unsigned int lineno, char const *fmt, ...) {
00054   va_list arg_list;
00055
00056   // on first call, initialize based on input arguments
00057   if (!pynq_log_init) {
00058     // if DEBUG is set, we also print log level INFO
00059     char const *env = getenv("DEBUG");
00060     if (env != NULL) {
00061       min_log_level = LOG_LEVEL_INFO;
00062     }
00063     // make warnings fatal
00064     env = getenv("FATAL_WARNING");
00065     if (env != NULL) {
00066       critical_level = LOG_LEVEL_WARNING;
00067     }
00068     pynq_log_init = true;
00069   }
00070   // check if the log level is valid
00071   if (level < LOG_LEVEL_INFO || level > LOG_LEVEL_ERROR) {
00072     printf("pynq_log: invalid log level specified (%d)\r\n", level);
00073     return;
00074   }
00075
00076   if (level < min_log_level) {
00077     return;
00078   }
00079   fputs(color_escape_calls[level], stderr);
00080   fputs(log_level_name[level], stderr);
00081
00082   fputs(color_escape_blue, stderr);
00083   if (domain != NULL) {
00084     fprintf(stderr, "%s::", domain);
00085   }
00086   fprintf(stderr, "%s:%d ", location, lineno);
00087   fputs(color_escape_reset, stderr);
00088
00089   va_start(arg_list, fmt);
```

```
00090   vfprintf(stderr, fmt, arg_list);
00091   va_end(arg_list);
00092   if (fmt[strlen(fmt) - 1] != '\n') {
00093     fputs("\n", stderr);
00094   }
00095
00096   if (level >= critical_level) {
00097     abort();
00098   }
00099 }
```

6.51   library/log.h File Reference

This graph shows which files directly or indirectly include this file:

Macros

- #define LOG_DOMAIN NULL
- #define pynq_info(...) pynq_log(LOG_LEVEL_INFO, LOG_DOMAIN, __FUNCTION__, __LINE__, __VA_↩ ARGS__)
- #define pynq_warning(...) pynq_log(LOG_LEVEL_WARNING, LOG_DOMAIN, __FUNCTION__, __LINE↩ __, __VA_ARGS__)
- #define pynq_error(...)

Typedefs

- typedef enum LogLevel LogLevel

Enumerations

- enum LogLevel { LOG_LEVEL_INFO , LOG_LEVEL_WARNING , LOG_LEVEL_ERROR , NUM_LOG_LEVELS }

Functions

- void pynq_log (const LogLevel level, char const *domain, char const *location, unsigned int lineno, char const *fmt,...)

6.51.1   Macro Definition Documentation

6.51.1.1   LOG_DOMAIN

```
#define LOG_DOMAIN NULL
```

Definition at line 25 of file log.h.

6.52 log.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef LOG_H
00023 #define LOG_H
00024
00025 #define LOG_DOMAIN NULL
00065 typedef enum LogLevel {
00067   LOG_LEVEL_INFO,
00069   LOG_LEVEL_WARNING,
00071   LOG_LEVEL_ERROR,
00073   NUM_LOG_LEVELS
00074 } LogLevel;
00075
00091 void pynq_log(const LogLevel level, char const *domain, char const *location,
00092               unsigned int lineno, char const *fmt, ...);
00093
00100 #define pynq_info(...)                                                      \
00101   pynq_log(LOG_LEVEL_INFO, LOG_DOMAIN, __FUNCTION__, __LINE__, __VA_ARGS__)
00102
00109 #define pynq_warning(...)                                                   \
00110   pynq_log(LOG_LEVEL_WARNING, LOG_DOMAIN, __FUNCTION__, __LINE__, __VA_ARGS__)
00111
00118 #define pynq_error(...)                                                     \
00119   do {                                                                      \
00120     pynq_log(LOG_LEVEL_ERROR, LOG_DOMAIN, __FUNCTION__, __LINE__,            \
00121              __VA_ARGS__);                                                  \
00122     for (;;)                                                                \
00123       ;                                                                     \
00124   } while (0)
00125
00127 #endif // LOG_H
```

6.53 library/pinmap.c File Reference

```
#include <pinmap.h>
```
Include dependency graph for pinmap.c:

Variables

- char *const pin_names []

6.54 pinmap.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
```

```
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <pinmap.h>
00023
00024 char *const pin_names[] = {
00026     "AR0",
00028     "AR1",
00030     "AR2",
00032     "AR3",
00034     "AR4",
00036     "AR5",
00038     "AR6",
00040     "AR7",
00042     "AR8",
00044     "AR9",
00046     "AR10",
00048     "AR11",
00050     "AR12",
00052     "AR13",
00053
00055     "A0",
00057     "A1",
00059     "A2",
00061     "A3",
00063     "A4",
00065     "A5",
00066
00068     "SW0",
00070     "SW1",
00073     "BTN0",
00075     "BTN1",
00077     "BTN2",
00079     "BTN3",
00082     "LD0",
00084     "LD1",
00086     "LD2",
00088     "LD3",
00089
00091     "AR_SDA",
00093     "AR_SCL",
00095     "LD4B",
00097     "LD4G",
00099     "LD4R",
00101     "LD5B",
00103     "LD5G",
00105     "LD5R",
00107     "RBPI40",
00109     "RBPI37",
00111     "RBPI38",
00113     "RBPI35",
00115     "RBPI36",
00117     "RBPI33",
00119     "RBPI18",
00121     "RBPI32",
00123     "RBPI10",
00125     "RBPI27",
00127     "RBPI28",
00129     "RBPI22",
00131     "RBPI23",
00133     "RBPI24",
00135     "RBPI21",
00137     "RBPI26",
00139     "RBPI19",
00141     "RBPI31",
00143     "RBPI15",
00145     "RBPI16",
00147     "RBPI13",
00149     "RBPI12",
00151     "RBPI29",
00153     "RBPI08",
00155     "RBPI07",
```

```
00157    "RBPI05",
00158 };
```

6.55 library/pinmap.h File Reference

This graph shows which files directly or indirectly include this file:

Macros

- #define NUM_ANALOG_REFERENCE_PINS 14 /∗ # analog reference pins ∗/
- #define NUM_ANALOG_IN_PINS 6 /∗ # analog input pins ∗/
- #define IO_PMODA1 IO_RBPI07
- #define IO_PMODA2 IO_RBPI29
- #define IO_PMODA3 IO_RBPI27
- #define IO_PMODA4 IO_RBPI28
- #define IO_PMODA7 IO_RBPI31
- #define IO_PMODA8 IO_RBPI26
- #define PIN_CHECK(pin)

Enumerations

- enum io_t {
  IO_AR0 = 0 , IO_AR1 = 1 , IO_AR2 = 2 , IO_AR3 = 3 ,
  IO_AR4 = 4 , IO_AR5 = 5 , IO_AR6 = 6 , IO_AR7 = 7 ,
  IO_AR8 = 8 , IO_AR9 = 9 , IO_AR10 = 10 , IO_AR11 = 11 ,
  IO_AR12 = 12 , IO_AR13 = 13 , IO_A0 = 14 , IO_A1 = 15 ,
  IO_A2 = 16 , IO_A3 = 17 , IO_A4 = 18 , IO_A5 = 19 ,
  IO_SW0 = 20 , IO_SW1 = 21 , IO_BTN0 = 22 , IO_BTN1 = 23 ,
  IO_BTN2 = 24 , IO_BTN3 = 25 , IO_LD0 = 26 , IO_LD1 = 27 ,
  IO_LD2 = 28 , IO_LD3 = 29 , IO_AR_SCL = 31 , IO_AR_SDA = 30 ,
  IO_LD4B = 32 , IO_LD4R = 33 , IO_LD4G = 34 , IO_LD5B = 35 ,
  IO_LD5R = 36 , IO_LD5G = 37 , IO_RBPI40 = 38 , IO_RBPI37 = 39 ,
  IO_RBPI38 = 40 , IO_RBPI35 = 41 , IO_RBPI36 = 42 , IO_RBPI33 = 43 ,
  IO_RBPI18 = 44 , IO_RBPI32 = 45 , IO_RBPI10 = 46 , IO_RBPI27 = 47 ,
  IO_RBPI28 = 48 , IO_RBPI22 = 49 , IO_RBPI23 = 50 , IO_RBPI24 = 51 ,
  IO_RBPI21 = 52 , IO_RBPI26 = 53 , IO_RBPI19 = 54 , IO_RBPI31 = 55 ,
  IO_RBPI15 = 56 , IO_RBPI16 = 57 , IO_RBPI13 = 58 , IO_RBPI12 = 59 ,
  IO_RBPI29 = 60 , IO_RBPI08 = 61 , IO_RBPI07 = 62 , IO_RBPI05 = 63 ,
  IO_NUM_PINS = 64 }

Variables

- char ∗const pin_names [64]

6.56 pinmap.h

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef PINMAP_H
00023 #define PINMAP_H
00024
00042 #define NUM_ANALOG_REFERENCE_PINS 14 /* # analog reference pins */
00043 #define NUM_ANALOG_IN_PINS 6        /* # analog input pins */
00044
00045 typedef enum {
00049   IO_AR0 = 0,   /* reference pin 0 */
00050   IO_AR1 = 1,   /* reference pin 1 */
00051   IO_AR2 = 2,   /* reference pin 2 */
00052   IO_AR3 = 3,   /* reference pin 3 */
00053   IO_AR4 = 4,   /* reference pin 4 */
00054   IO_AR5 = 5,   /* reference pin 5 */
00055   IO_AR6 = 6,   /* reference pin 6 */
00056   IO_AR7 = 7,   /* reference pin 7 */
00057   IO_AR8 = 8,   /* reference pin 8 */
00058   IO_AR9 = 9,   /* reference pin 9 */
00059   IO_AR10 = 10, /* reference pin 10 */
00060   IO_AR11 = 11, /* reference pin 11 */
00061   IO_AR12 = 12, /* reference pin 12 */
00062   IO_AR13 = 13, /* reference pin 13 */
00063
00067   IO_A0 = 14, /* analog input pin 0 */
00068   IO_A1 = 15, /* analog input pin 1 */
00069   IO_A2 = 16, /* analog input pin 2 */
00070   IO_A3 = 17, /* analog input pin 3 */
00071   IO_A4 = 18, /* analog input pin 4 */
00072   IO_A5 = 19, /* analog input pin 5 */
00073
00077   IO_SW0 = 20, /* switch input pin 0 */
00078   IO_SW1 = 21, /* switch input pin 1 */
00079
00083   IO_BTN0 = 22, /* button input pin 0 */
00084   IO_BTN1 = 23, /* button input pin 1 */
00085   IO_BTN2 = 24, /* button input pin 2 */
00086   IO_BTN3 = 25, /* button input pin 3 */
00087
00091   IO_LD0 = 26, /* LED output pin 0 */
00092   IO_LD1 = 27, /* LED output pin 1 */
00093   IO_LD2 = 28, /* LED output pin 2 */
00094   IO_LD3 = 29, /* LED output pin 3 */
00095
00099   IO_AR_SCL = 31, /* I2C clock pin */
00100   IO_AR_SDA = 30, /* I2C data pin */
00101
00106   IO_LD4B = 32, /* color LED 0 blue input pin */
00107   IO_LD4R = 33, /* color LED 0 red input pin */
00108   IO_LD4G = 34, /* color LED 0 green input pin */
00109
00110   IO_LD5B = 35, /* color LED 1 blue input pin */
00111   IO_LD5R = 36, /* color LED 1 red input pin */
00112   IO_LD5G = 37, /* color LED 1 green input pin */
00113
00117   IO_RBPI40 = 38, /* RaspberryPi header pin */
00118   IO_RBPI37 = 39, /* RaspberryPi header pin */
00119   IO_RBPI38 = 40, /* RaspberryPi header pin */
00120   IO_RBPI35 = 41, /* RaspberryPi header pin */
00121   IO_RBPI36 = 42, /* RaspberryPi header pin */
00122   IO_RBPI33 = 43, /* RaspberryPi header pin */
00123   IO_RBPI18 = 44, /* RaspberryPi header pin */
00124   IO_RBPI32 = 45, /* RaspberryPi header pin */
```

```
00125   IO_RBPI10 = 46, /* RaspberryPi header pin */
00126   IO_RBPI27 = 47, /* RaspberryPi header pin */
00127   IO_RBPI28 = 48, /* RaspberryPi header pin */
00128   IO_RBPI22 = 49, /* RaspberryPi header pin */
00129   IO_RBPI23 = 50, /* RaspberryPi header pin */
00130   IO_RBPI24 = 51, /* RaspberryPi header pin */
00131   IO_RBPI21 = 52, /* RaspberryPi header pin */
00132   IO_RBPI26 = 53, /* RaspberryPi header pin */
00133   IO_RBPI19 = 54, /* RaspberryPi header pin */
00134   IO_RBPI31 = 55, /* RaspberryPi header pin */
00135   IO_RBPI15 = 56, /* RaspberryPi header pin */
00136   IO_RBPI16 = 57, /* RaspberryPi header pin */
00137   IO_RBPI13 = 58, /* RaspberryPi header pin */
00138   IO_RBPI12 = 59, /* RaspberryPi header pin */
00139   IO_RBPI29 = 60, /* RaspberryPi header pin */
00140   IO_RBPI08 = 61, /* RaspberryPi header pin */
00141   IO_RBPI07 = 62, /* RaspberryPi header pin */
00142   IO_RBPI05 = 63, /* RaspberryPi header pin */
00143
00144   IO_NUM_PINS = 64,
00145 } io_t;
00146
00150 #define IO_PMODA1       IO_RBPI07
00151 #define IO_PMODA2       IO_RBPI29
00152 #define IO_PMODA3       IO_RBPI27
00153 #define IO_PMODA4       IO_RBPI28
00154 #define IO_PMODA7       IO_RBPI31
00155 #define IO_PMODA8       IO_RBPI26
00156
00160 #define PIN_CHECK(pin)                                                       \
00161   do {                                                                       \
00162     if (pin >= IO_NUM_PINS) {                                                \
00163       pynq_error("pin %u is invalid, must be 0..%u-1.\n", pin, IO_NUM_PINS); \
00164     }                                                                        \
00165   } while (0);
00166
00170 extern char *const pin_names[64];
00174 #endif // PINMAP_H
```

6.57 library/pwm.c File Reference

#include <libpynq.h>
Include dependency graph for pwm.c:

Enumerations

- enum PWM_Regs { PWM_REG_DUTY = 0 , PWM_REG_PERIOD = 1 , PWM_REG_NEW_STEP_COUNT = 2 , PWM_REG_CUR_STEP_COUNT = 3 }

Functions

- bool pwm_initialized (const int pwm)
- bool check_initialized_pwm (const int pwm)
- void pwm_init (const int pwm, const uint32_t period)
- void pwm_destroy (const int pwm)
- uint32_t pwm_get_duty_cycle (const int pwm)
- uint32_t pwm_get_period (const int pwm)
- void pwm_set_period (const int pwm, const uint32_t period)
- void pwm_set_duty_cycle (const int pwm, const uint32_t duty)
- uint32_t pwm_get_steps (const int pwm)
- void pwm_set_steps (const int pwm, const uint32_t steps)

6.57.1 Enumeration Type Documentation

6.57.1.1 PWM_Regs

enum PWM_Regs

Enumerator

PWM_REG_DUTY	
PWM_REG_PERIOD	
PWM_REG_NEW_STEP_COUNT	
PWM_REG_CUR_STEP_COUNT	

Definition at line 24 of file pwm.c.

6.57.2 Function Documentation

6.57.2.1 check_initialized_pwm()

```
bool check_initialized_pwm (
            const int pwm )
```

Definition at line 49 of file pwm.c.

Here is the caller graph for this function:

6.58 pwm.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <libpynq.h>
00023
00024 enum PWM_Regs {
00025   PWM_REG_DUTY = 0,
00026   PWM_REG_PERIOD = 1,
00027   PWM_REG_NEW_STEP_COUNT = 2,
00028   PWM_REG_CUR_STEP_COUNT = 3,
00029 };
00030
00031 static struct arm_shared_t channels[NUM_PWMS] = {
00032     0,
00033 };
00034 static volatile uint32_t *initializedChannel[NUM_PWMS] = {
00035     NULL,
00036 };
00037
00038 bool pwm_initialized(const int pwm) {
00039   if (pwm < 0 || pwm >= NUM_PWMS) {
00040     pynq_error("pwm_initialized: invalid pwm=%d, must be 0..%d-1\n", pwm,
00041               NUM_PWMS);
00042   }
00043   if (initializedChannel[pwm] == NULL) {
```

```
00044      return false;
00045    }
00046    return true;
00047 }
00048
00049 bool check_initialized_pwm(const int pwm) {
00050    if (pwm < 0 || pwm >= NUM_PWMS) {
00051      pynq_error("pwm_initialized: invalid pwm=%d, must be 0..%d-1\n", pwm,
00052                 NUM_PWMS);
00053    }
00054    if (initializedChannel[pwm] == NULL) {
00055      pynq_error("pwm_initialized: channel of pwm %d has not been initialized\n",
00056                 pwm);
00057    }
00058    return true;
00059 }
00060
00061 void pwm_init(const int pwm, const uint32_t period) {
00062    if (pwm < 0 || pwm >= NUM_PWMS) {
00063      pynq_error("pwm_init: invalid pwm=%d, must be 0..%d-1\n", pwm, NUM_PWMS);
00064    }
00065    uint32_t channelAddr = axi_pwm_base + (pwm * 0x10000);
00066    initializedChannel[pwm] = arm_shared_init(&channels[pwm], channelAddr, 512);
00067    initializedChannel[pwm][PWM_REG_DUTY] = 0;
00068    initializedChannel[pwm][PWM_REG_PERIOD] = period;
00069    initializedChannel[pwm][PWM_REG_NEW_STEP_COUNT] = -1;
00070 }
00071
00072 void pwm_destroy(const int pwm) {
00073    (void)check_initialized_pwm(pwm);
00074    arm_shared_close(&channels[pwm]);
00075    initializedChannel[pwm] = NULL;
00076 }
00077
00078 uint32_t pwm_get_duty_cycle(const int pwm) {
00079    (void)check_initialized_pwm(pwm);
00080    return initializedChannel[pwm][PWM_REG_DUTY];
00081 }
00082
00083 uint32_t pwm_get_period(const int pwm) {
00084    (void)check_initialized_pwm(pwm);
00085    return initializedChannel[pwm][PWM_REG_PERIOD];
00086 }
00087
00088 void pwm_set_period(const int pwm, const uint32_t period) {
00089    (void)check_initialized_pwm(pwm);
00090    initializedChannel[pwm][PWM_REG_PERIOD] = period;
00091 }
00092
00093 void pwm_set_duty_cycle(const int pwm, const uint32_t duty) {
00094    (void)check_initialized_pwm(pwm);
00095    initializedChannel[pwm][PWM_REG_DUTY] = duty;
00096 }
00097
00098 uint32_t pwm_get_steps(const int pwm) {
00099    (void)check_initialized_pwm(pwm);
00100    return initializedChannel[pwm][PWM_REG_NEW_STEP_COUNT];
00101 }
00102
00103 void pwm_set_steps(const int pwm, const uint32_t steps) {
00104    (void)check_initialized_pwm(pwm);
00105    initializedChannel[pwm][PWM_REG_NEW_STEP_COUNT] = steps;
00106 }
```

6.59 library/pwm.h File Reference

```
#include <libpynq.h>
```
Include dependency graph for pwm.h: This graph shows which files directly or indirectly include this file:

Enumerations

- enum pwm_index_t {
  PWM0 , PWM1 , PWM2 , PWM3 ,
  PWM4 , PWM5 , NUM_PWMS }

Functions

- bool pwm_initialized (const int pwm)
- void pwm_init (const int pwm, const uint32_t period)
- void pwm_destroy (const int pwm)
- void pwm_set_duty_cycle (const int pwm, const uint32_t duty)
- void pwm_set_period (const int pwm, const uint32_t period)
- uint32_t pwm_get_period (const int pwm)
- uint32_t pwm_get_duty_cycle (const int pwm)
- void pwm_set_steps (const int pwm, const uint32_t steps)
- uint32_t pwm_get_steps (const int pwm)

6.60   pwm.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef PWM_H
00023 #define PWM_H
00024 #include <libpynq.h>
00025
00047 typedef enum { PWM0, PWM1, PWM2, PWM3, PWM4, PWM5, NUM_PWMS } pwm_index_t;
00048
00055 extern bool pwm_initialized(const int pwm);
00056
00063 extern void pwm_init(const int pwm, const uint32_t period);
00064
00070 extern void pwm_destroy(const int pwm);
00071
00079 extern void pwm_set_duty_cycle(const int pwm, const uint32_t duty);
00080
00088 extern void pwm_set_period(const int pwm, const uint32_t period);
00089
00097 uint32_t pwm_get_period(const int pwm);
00098
00106 extern uint32_t pwm_get_duty_cycle(const int pwm);
00107
00116 extern void pwm_set_steps(const int pwm, const uint32_t steps);
00117
00126 extern uint32_t pwm_get_steps(const int pwm);
00127
00131 #endif
```

6.61   library/switchbox.c File Reference

```
#include "switchbox.h"
#include <libpynq.h>
```
Include dependency graph for switchbox.c:

Data Structures

- struct pin

Functions

- void switchbox_init (void)
- void switchbox_destroy (void)
- void switchbox_reset (void)
- void switchbox_set_pin (const io_t pin_number, const io_configuration_t io_type)
- io_configuration_t switchbox_get_pin (const io_t pin_number)

Variables

- char ∗const switchbox_names [NUM_SWITCHBOX_NAMES]
- arm_shared ioswitch_handle
- volatile uint32_t ∗ ioswitch = NULL

6.61.1 Variable Documentation

6.61.1.1 ioswitch

```
volatile uint32_t* ioswitch = NULL
```

Definition at line 97 of file switchbox.c.

6.61.1.2 ioswitch_handle

```
arm_shared ioswitch_handle
```

Definition at line 96 of file switchbox.c.

6.62 switchbox.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
```

```
00022 #include "switchbox.h"
00023 #include <libpynq.h>
00024
00025 char *const switchbox_names[NUM_SWITCHBOX_NAMES] = {
00027     "SWB_GPIO",
00029     "SWB_Interrupt_In",
00031     "SWB_UART0_TX",
00033     "SWB_UART0_RX",
00035     "SWB_SPI0_CLK",
00037     "SWB_SPI0_MISO",
00039     "SWB_SPI0_MOSI",
00041     "SWB_SPI0_SS",
00043     "SWB_SPI1_CLK",
00045     "SWB_SPI1_MISO",
00047     "SWB_SPI1_MOSI",
00049     "SWB_SPI1_SS",
00051     "SWB_IIC0_SDA",
00053     "SWB_IIC0_SCL",
00055     "SWB_IIC1_SDA",
00057     "SWB_IIC1_SCL",
00059     "SWB_PWM0",
00061     "SWB_PWM1",
00063     "SWB_PWM2",
00065     "SWB_PWM3",
00067     "SWB_PWM4",
00069     "SWB_PWM5",
00070     "SWB_TIMER_G0",
00071     "SWB_TIMER_G1",
00073     "SWB_TIMER_G2",
00075     "SWB_TIMER_G3",
00077     "SWB_TIMER_G4",
00079     "SWB_TIMER_G5",
00081     "SWB_TIMER_G6",
00083     "SWB_TIMER_G7",
00084     "SWB_UART1_TX",
00085     "SWB_UART1_RX",
00086     "SWB_TIMER_IC0",
00087     "SWB_TIMER_IC1",
00088     "SWB_TIMER_IC2",
00089     "SWB_TIMER_IC3",
00090     "SWB_TIMER_IC4",
00091     "SWB_TIMER_IC5",
00092     "SWB_TIMER_IC6",
00093     "SWB_TIMER_IC7",
00094 };
00095
00096 arm_shared ioswitch_handle;
00097 volatile uint32_t *ioswitch = NULL;
00098
00099 typedef struct {
00100   char *name;
00101   char *state;
00102   io_configuration_t channel; // was uint8_t
00103 } pin;
00104
00105 void switchbox_init(void) {
00106   // allocate shared memory for the switchbox and store the pointer in
00107   // `ioswitch`
00108   check_version();
00109   ioswitch = arm_shared_init(&ioswitch_handle, io_switch_0, 4096);
00110 }
00111
00112 void switchbox_destroy(void) {
00113   // free the sared memory in the switchbox
00114   arm_shared_close(&ioswitch_handle);
00115 }
00116
00117 // reset all switchbox pins to 0
00118 void switchbox_reset(void) {
00119   // 32 pins to remap, 4 per word.
00120   for (uint_fast32_t i = 0; i < (64 / 4); i++) {
00121     // set all words to 0
00122     ioswitch[i] = 0;
00123   }
00124 }
00125
00126 void switchbox_set_pin(const io_t pin_number,
00127                       const io_configuration_t io_type) {
00128   int numWordstoPass, byteNumber;
00129   uint32_t switchConfigValue;
00130
00131   PIN_CHECK(pin_number);
00132
00133   // If gpio is initialized, set the pin as input, if PIN_TYPE is
00134   // not gpio
00135   if (io_type != SWB_GPIO && gpio_is_initialized()) {
00136     // set pin as input.
```

```
00137    if (gpio_get_direction(pin_number) != GPIO_DIR_INPUT) {
00138      pynq_warning("pin: %s is set as GPIO ouput, but not mapped as GPIO. "
00139                   "Reconfiguring as input.",
00140                   pin_names[pin_number]);
00141      gpio_set_direction(pin_number, GPIO_DIR_INPUT);
00142    }
00143  }
00144
00145  // calculate the word and byte number for the given pin number
00146  numWordstoPass = pin_number / 4;
00147  byteNumber = pin_number % 4;
00148
00149  // get the current value of the word containing the pin
00150  switchConfigValue = ioswitch[numWordstoPass];
00151
00152  // clear the byte containing the pin type and set it to the new value
00153  switchConfigValue = (switchConfigValue & (~(0xFF « (byteNumber * 8)))) |
00154                      (io_type « (byteNumber * 8));
00155
00156  // update the word in the switchbox with the new value
00157  ioswitch[numWordstoPass] = switchConfigValue;
00158 }
00159
00160 // pin_number: the number of the pin to get
00161 // returns: the type of the given pin
00162 io_configuration_t switchbox_get_pin(const io_t pin_number) {
00163   int numWordstoPass, byteNumber;
00164   uint32_t switchConfigValue;
00165
00166   PIN_CHECK(pin_number);
00167
00168   // calculate the word and byte number for the given pin number
00169   numWordstoPass = pin_number / 4;
00170   byteNumber = pin_number % 4;
00171
00172   // get the value of the word containing the pin and extract the value of the
00173   // byte containing the pin type
00174   switchConfigValue = ioswitch[numWordstoPass];
00175   switchConfigValue = (switchConfigValue » (byteNumber * 8)) & 0xFF;
00176
00177   // return pintype
00178   return switchConfigValue;
00179 }
```

6.63 library/switchbox.h File Reference

```
#include <pinmap.h>
#include <stdbool.h>
#include <stdint.h>
```
Include dependency graph for switchbox.h: This graph shows which files directly or indirectly include this file:

Macros

- #define NUM_SWITCHBOX_NAMES 40

Typedefs

- typedef enum io_configuration io_configuration_t

Enumerations

- enum io_configuration {
  SWB_GPIO = 0x00 , SWB_Interrupt_In = 0x01 , SWB_UART0_TX = 0x02 , SWB_UART0_RX = 0x03 ,
  SWB_SPI0_CLK = 0x04 , SWB_SPI0_MISO = 0x05 , SWB_SPI0_MOSI = 0x06 , SWB_SPI0_SS = 0x07 ,
  SWB_SPI1_CLK = 0x08 , SWB_SPI1_MISO = 0x09 , SWB_SPI1_MOSI = 0x0A , SWB_SPI1_SS = 0x0B ,
  SWB_IIC0_SDA = 0x0C , SWB_IIC0_SCL = 0x0D , SWB_IIC1_SDA = 0x0E , SWB_IIC1_SCL = 0x0F ,

SWB_PWM0 = 0x10 , SWB_PWM1 = 0x11 , SWB_PWM2 = 0x12 , SWB_PWM3 = 0x13 ,
SWB_PWM4 = 0x14 , SWB_PWM5 = 0x15 , SWB_TIMER_G0 = 0x18 , SWB_TIMER_G1 = 0x19 ,
SWB_TIMER_G2 = 0x1A , SWB_TIMER_G3 = 0x1B , SWB_TIMER_G4 = 0x1C , SWB_TIMER_G5 = 0x1D
,
SWB_TIMER_G6 = 0x1E , SWB_TIMER_G7 = 0x1F , SWB_UART1_TX = 0x22 , SWB_UART1_RX = 0x23 ,
SWB_TIMER_IC0 = 0x38 , SWB_TIMER_IC1 = 0x39 , SWB_TIMER_IC2 = 0x3A , SWB_TIMER_IC3 = 0x3B
,
SWB_TIMER_IC4 = 0x3C , SWB_TIMER_IC5 = 0x3D , SWB_TIMER_IC6 = 0x3E , SWB_TIMER_IC7 = 0x3F
,
NUM_IO_CONFIGURATIONS }

Functions

- void switchbox_init (void)
- void switchbox_set_pin (const io_t pin_number, const io_configuration_t pin_type)
- void switchbox_reset (void)
- void switchbox_destroy (void)
- io_configuration_t switchbox_get_pin (const io_t pin_number)

Variables

- char ∗const switchbox_names [NUM_SWITCHBOX_NAMES]

6.64 switchbox.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef SWITCHBOX_H
00023 #define SWITCHBOX_H
00024 #include <pinmap.h>
00025 #include <stdbool.h>
00026 #include <stdint.h>
00027
00062 typedef enum io_configuration {
00064   SWB_GPIO = 0x00,
00066   SWB_Interrupt_In = 0x01,
00068   SWB_UART0_TX = 0x02,
00070   SWB_UART0_RX = 0x03,
00072   SWB_SPI0_CLK = 0x04,
00074   SWB_SPI0_MISO = 0x05,
00076   SWB_SPI0_MOSI = 0x06,
00078   SWB_SPI0_SS = 0x07,
00080   SWB_SPI1_CLK = 0x08,
00082   SWB_SPI1_MISO = 0x09,
00084   SWB_SPI1_MOSI = 0x0A,
00086   SWB_SPI1_SS = 0x0B,
00088   SWB_IIC0_SDA = 0x0C,
```

```
00090    SWB_IIC0_SCL = 0x0D,
00092    SWB_IIC1_SDA = 0x0E,
00094    SWB_IIC1_SCL = 0x0F,
00096    SWB_PWM0 = 0x10,
00098    SWB_PWM1 = 0x11,
00100    SWB_PWM2 = 0x12,
00102    SWB_PWM3 = 0x13,
00104    SWB_PWM4 = 0x14,
00106    SWB_PWM5 = 0x15,
00107    SWB_TIMER_G0 = 0x18,
00108    SWB_TIMER_G1 = 0x19,
00110    SWB_TIMER_G2 = 0x1A,
00112    SWB_TIMER_G3 = 0x1B,
00114    SWB_TIMER_G4 = 0x1C,
00116    SWB_TIMER_G5 = 0x1D,
00118    SWB_TIMER_G6 = 0x1E,
00120    SWB_TIMER_G7 = 0x1F,
00121    SWB_UART1_TX = 0x22,
00122    SWB_UART1_RX = 0x23,
00123    SWB_TIMER_IC0 = 0x38,
00124    SWB_TIMER_IC1 = 0x39,
00125    SWB_TIMER_IC2 = 0x3A,
00126    SWB_TIMER_IC3 = 0x3B,
00127    SWB_TIMER_IC4 = 0x3C,
00128    SWB_TIMER_IC5 = 0x3D,
00129    SWB_TIMER_IC6 = 0x3E,
00130    SWB_TIMER_IC7 = 0x3F,
00132    NUM_IO_CONFIGURATIONS,
00133  } io_configuration_t;
00134
00135  #define NUM_SWITCHBOX_NAMES 40
00140  extern char *const switchbox_names[NUM_SWITCHBOX_NAMES];
00141
00147  extern void switchbox_init(void);
00148
00155  extern void switchbox_set_pin(const io_t pin_number,
00156                                const io_configuration_t pin_type);
00157
00162  extern void switchbox_reset(void);
00163
00167  extern void switchbox_destroy(void);
00168
00175  extern io_configuration_t switchbox_get_pin(const io_t pin_number);
00176
00180  #endif // SWITCHBOX_H
```

6.65 library/uart.c File Reference

```
#include "uart.h"
#include "arm_shared_memory_system.h"
#include "log.h"
#include <platform.h>
#include <stdio.h>
```
Include dependency graph for uart.c:

Macros

- #define UART_REG_RECEIVE_FIFO 0
- #define UART_REG_TRANSMIT_FIFO 1
- #define UART_REG_STATUS 2
- #define UART_REG_CONTROL 3
- #define UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA 1
- #define UART_REG_STATUS_BIT_RX_FIFO_FULL 2
- #define UART_REG_STATUS_BIT_TX_FIFO_EMPTY 4
- #define UART_REG_STATUS_BIT_TX_FIFO_FULL 8
- #define UART_REG_CONTROL_BIT_CLEAR_TX_FIFO 1
- #define UART_REG_CONTROL_BIT_CLEAR_RX_FIFO 2
- #define UART_REG_CONTROL_BIT_CLEAR_FIFOS (UART_REG_CONTROL_BIT_CLEAR_RX_FIFO | UART_REG_CONTROL_BIT_CLEAR_TX_FIFO)

Functions

- void [uart_init](const int uart)
- void [uart_destroy](const int uart)
- void [uart_send](const int uart, const uint8_t data)
- uint8_t [uart_recv](const int uart)
- bool [uart_has_data](const int uart)
- bool [uart_has_space](const int uart)
- void [uart_reset_fifos](const int uart)

6.65.1 Macro Definition Documentation

6.65.1.1 UART_REG_CONTROL

```
#define UART_REG_CONTROL 3
```

Definition at line 31 of file uart.c.

6.65.1.2 UART_REG_CONTROL_BIT_CLEAR_FIFOS

```
#define UART_REG_CONTROL_BIT_CLEAR_FIFOS  (UART_REG_CONTROL_BIT_CLEAR_RX_FIFO | UART_REG_CONTROL_BIT_CLEAR_TX_
```

Definition at line 40 of file uart.c.

6.65.1.3 UART_REG_CONTROL_BIT_CLEAR_RX_FIFO

```
#define UART_REG_CONTROL_BIT_CLEAR_RX_FIFO 2
```

Definition at line 39 of file uart.c.

6.65.1.4 UART_REG_CONTROL_BIT_CLEAR_TX_FIFO

```
#define UART_REG_CONTROL_BIT_CLEAR_TX_FIFO 1
```

Definition at line 38 of file uart.c.

6.65.1.5 UART_REG_RECEIVE_FIFO

```
#define UART_REG_RECEIVE_FIFO 0
```

Definition at line 28 of file uart.c.

6.65.1.6 UART_REG_STATUS

```
#define UART_REG_STATUS 2
```

Definition at line 30 of file uart.c.

6.65.1.7 UART_REG_STATUS_BIT_RX_FIFO_FULL

```
#define UART_REG_STATUS_BIT_RX_FIFO_FULL 2
```

Definition at line 34 of file uart.c.

6.65.1.8 UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA

```
#define UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA 1
```

Definition at line 33 of file uart.c.

6.65.1.9 UART_REG_STATUS_BIT_TX_FIFO_EMPTY

```
#define UART_REG_STATUS_BIT_TX_FIFO_EMPTY 4
```

Definition at line 35 of file uart.c.

6.65.1.10 UART_REG_STATUS_BIT_TX_FIFO_FULL

```
#define UART_REG_STATUS_BIT_TX_FIFO_FULL 8
```

Definition at line 36 of file uart.c.

6.65.1.11 UART_REG_TRANSMIT_FIFO

```
#define UART_REG_TRANSMIT_FIFO 1
```

Definition at line 29 of file uart.c.

6.66 uart.c

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include "uart.h"
00023 #include "arm_shared_memory_system.h"
```

```
00024 #include "log.h"
00025 #include <platform.h>
00026 #include <stdio.h>
00027
00028 #define UART_REG_RECEIVE_FIFO 0
00029 #define UART_REG_TRANSMIT_FIFO 1
00030 #define UART_REG_STATUS 2
00031 #define UART_REG_CONTROL 3
00032
00033 #define UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA 1
00034 #define UART_REG_STATUS_BIT_RX_FIFO_FULL 2
00035 #define UART_REG_STATUS_BIT_TX_FIFO_EMPTY 4
00036 #define UART_REG_STATUS_BIT_TX_FIFO_FULL 8
00037
00038 #define UART_REG_CONTROL_BIT_CLEAR_TX_FIFO 1
00039 #define UART_REG_CONTROL_BIT_CLEAR_RX_FIFO 2
00040 #define UART_REG_CONTROL_BIT_CLEAR_FIFOS                                  \
00041   (UART_REG_CONTROL_BIT_CLEAR_RX_FIFO | UART_REG_CONTROL_BIT_CLEAR_TX_FIFO)
00042
00043 static arm_shared uart_handles[NUM_UARTS];
00044 static volatile uint32_t *uart_ptrs[NUM_UARTS] = {
00045     NULL,
00046 };
00047
00048 void uart_init(const int uart) {
00049   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00050     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00051   }
00052   if (uart == UART0) {
00053     uart_ptrs[uart] =
00054         arm_shared_init(&(uart_handles[uart]), axi_uartlite_0, 4096);
00055   } else if (uart == UART1) {
00056     uart_ptrs[uart] =
00057         arm_shared_init(&(uart_handles[uart]), axi_uartlite_1, 4096);
00058   }
00059 }
00060
00061 void uart_destroy(const int uart) {
00062   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00063     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00064   }
00065   if (uart_ptrs[uart] == NULL) {
00066     pynq_error("UART%d has not been initialized.\n", uart);
00067   }
00068   arm_shared_close(&(uart_handles[uart]));
00069   uart_ptrs[uart] = NULL;
00070 }
00071
00072 void uart_send(const int uart, const uint8_t data) {
00073   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00074     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00075   }
00076   if (uart_ptrs[uart] == NULL) {
00077     pynq_error("UART%d has not been initialized.\n", uart);
00078   }
00079   while ((uart_ptrs[uart][UART_REG_STATUS] &
00080           UART_REG_STATUS_BIT_TX_FIFO_FULL) == UART_REG_STATUS_BIT_TX_FIFO_FULL)
00081     ;
00082   uart_ptrs[uart][UART_REG_TRANSMIT_FIFO] = data;
00083 }
00084
00085 uint8_t uart_recv(const int uart) {
00086   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00087     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00088   }
00089   if (uart_ptrs[uart] == NULL) {
00090     pynq_error("UART%d has not been initialized.\n", uart);
00091   }
00092   while ((uart_ptrs[uart][UART_REG_STATUS] &
00093           UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA) == 0) {
00094   }
00095   return uart_ptrs[uart][UART_REG_RECEIVE_FIFO];
00096 }
00097
00098 bool uart_has_data(const int uart) {
00099   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00100     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00101   }
00102   if (uart_ptrs[uart] == NULL) {
00103     pynq_error("UART%d has not been initialized.\n", uart);
00104   }
00105   return ((uart_ptrs[uart][UART_REG_STATUS] &
00106            UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA) ==
00107           UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA);
00108 }
00109
00110 bool uart_has_space(const int uart) {
```

```
00111   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00112     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00113   }
00114   if (uart_ptrs[uart] == NULL) {
00115     pynq_error("UART%d has not been initialized.\n", uart);
00116   }
00117   return ((uart_ptrs[uart][UART_REG_STATUS] &
00118           UART_REG_STATUS_BIT_TX_FIFO_FULL) == 0);
00119 }
00120
00121 void uart_reset_fifos(const int uart) {
00122   if (!(uart >= UART0 && uart < NUM_UARTS)) {
00123     pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
00124   }
00125   if (uart_ptrs[uart] == NULL) {
00126     pynq_error("UART%d has not been initialized.\n", uart);
00127   }
00128   uart_ptrs[uart][UART_REG_CONTROL] = UART_REG_CONTROL_BIT_CLEAR_FIFOS;
00129 }
```

6.67 library/uart.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
```
Include dependency graph for uart.h: This graph shows which files directly or indirectly include this file:

Enumerations

- enum uart_index_t { UART0 = 0 , UART1 = 1 , NUM_UARTS }

Functions

- void uart_init (const int uart)
- void uart_destroy (const int uart)
- void uart_send (const int uart, const uint8_t data)
- uint8_t uart_recv (const int uart)
- bool uart_has_data (const int uart)
- bool uart_has_space (const int uart)
- void uart_reset_fifos (const int uart)

6.68 uart.h

Go to the documentation of this file.
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
```

```
00021 */
00022 #ifndef UART_H
00023 #define UART_H
00024 #include <stdbool.h>
00025 #include <stdint.h>
00026
00107 typedef enum { UART0 = 0, UART1 = 1, NUM_UARTS } uart_index_t;
00108
00116 extern void uart_init(const int uart);
00117
00123 extern void uart_destroy(const int uart);
00124
00132 extern void uart_send(const int uart, const uint8_t data);
00133
00142 extern uint8_t uart_recv(const int uart);
00143
00151 extern bool uart_has_data(const int uart);
00152
00160 extern bool uart_has_space(const int uart);
00161
00174 extern void uart_reset_fifos(const int uart);
00175
00180 #endif // UART_H
```

6.69 library/uio.c File Reference

```
#include "uio.h"
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <unistd.h>
```
Include dependency graph for uio.c:

Functions

- void ∗ setUIO (int uio_index, int length)
- int unsetUIO (void ∗uio_ptr, int length)

6.69.1 Function Documentation

6.69.1.1 setUIO()

```
void * setUIO (
            int uio_index,
            int length )
```

Definition at line 65 of file uio.c.

Here is the caller graph for this function:

6.69.1.2 unsetUIO()

```
int unsetUIO (
            void * uio_ptr,
            int length )
```

Definition at line 86 of file uio.c.

Here is the caller graph for this function:

6.70 uio.c

```
00001 /******************************************************************************
00002  *  Copyright (c) 2016, Xilinx, Inc.
00003  *  All rights reserved.
00004  *
00005  *  Redistribution and use in source and binary forms, with or without
00006  *  modification, are permitted provided that the following conditions are met:
00007  *
00008  *  1.  Redistributions of source code must retain the above copyright notice,
00009  *     this list of conditions and the following disclaimer.
00010  *
00011  *  2.  Redistributions in binary form must reproduce the above copyright
00012  *      notice, this list of conditions and the following disclaimer in the
00013  *      documentation and/or other materials provided with the distribution.
00014  *
00015  *  3.  Neither the name of the copyright holder nor the names of its
00016  *      contributors may be used to endorse or promote products derived from
00017  *      this software without specific prior written permission.
00018  *
00019  *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00020  *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00021  *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
00022  *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
00023  *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00026  *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00027  *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00028  *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00029  *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032 /******************************************************************************
00033  *
00034  *
00035  * @file uio.c
00036  *
00037  * Functions to interact with linux UIO. No safe checks here, so users must
00038  * know what they are doing.
00039  *
00040  * <pre>
00041  * MODIFICATION HISTORY:
00042  *
00043  * Ver   Who  Date     Changes
00044  * ----- --- -------- -----------------------------------------------
00045  * 1.00a yrq 12/05/17 Initial release
00046  *
00047  * </pre>
00048  *
00049  ******************************************************************************/
00050
00051 #include "uio.h"
00052 #include <fcntl.h>
00053 #include <stdio.h>
00054 #include <stdlib.h>
00055 #include <string.h>
00056 #include <sys/mman.h>
00057 #include <unistd.h>
00058
00059 /******************************************************************************
00060  * Function to set the UIO device.
00061  * @param   uio_index is the uio index in /dev list.
00062  * @param   length is the length of the MMAP in bytes.
00063  * @return  A pointer pointing to the MMAP of the UIO.
00064  ******************************************************************************/
00065 void *setUIO(int uio_index, int length) {
00066   char uio_buf[32];
00067   int uio_fd;
00068   void *uio_ptr;
00069
00070   sprintf(uio_buf, "/dev/uio%d", uio_index);
00071   uio_fd = open(uio_buf, O_RDWR);
00072   if (uio_fd < 1) {
00073     printf("Invalid UIO device file: %s.\n", uio_buf);
00074   }
00075   // mmap the UIO devices
00076   uio_ptr = mmap(NULL, length, PROT_READ | PROT_WRITE, MAP_SHARED, uio_fd, 0);
00077   return uio_ptr;
00078 }
00079
00080 /******************************************************************************
00081  * Function to set the UIO device.
00082  * @param   uio_ptr is the uio pointer to be freed.
```

```
00083   * @param   length is the length of the MMAP.
00084   * @return  0 on success; -1 otherwise.
00085   ******************************************************************/
00086 int unsetUIO(void *uio_ptr, int length) { return munmap(uio_ptr, length); }
```

6.71   library/uio.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void * **setUIO** (int uio_index, int length)
- int **unsetUIO** (void *uio_ptr, int length)

6.71.1   Detailed Description

Functions to interact with linux UIO.

```
MODIFICATION HISTORY:

Ver   Who      Date      Changes
----_ --------  --------  ------------------------------------------------
1.00  yrq      12/05/17 Initial release
```

Definition in file uio.h.

6.71.2   Function Documentation

6.71.2.1   setUIO()

```
void * setUIO (
            int uio_index,
            int length )
```

Definition at line 65 of file uio.c.

Here is the caller graph for this function:

6.71.2.2   unsetUIO()

```
int unsetUIO (
            void * uio_ptr,
            int length )
```

Definition at line 86 of file uio.c.

Here is the caller graph for this function:

6.72 uio.h

Go to the documentation of this file.
```
00001 /*******************************************************************************
00002  *  Copyright (c) 2016, Xilinx, Inc.
00003  *  All rights reserved.
00004  *
00005  *  Redistribution and use in source and binary forms, with or without
00006  *  modification, are permitted provided that the following conditions are met:
00007  *
00008  *  1.  Redistributions of source code must retain the above copyright notice,
00009  *      this list of conditions and the following disclaimer.
00010  *
00011  *  2.  Redistributions in binary form must reproduce the above copyright
00012  *      notice, this list of conditions and the following disclaimer in the
00013  *      documentation and/or other materials provided with the distribution.
00014  *
00015  *  3.  Neither the name of the copyright holder nor the names of its
00016  *      contributors may be used to endorse or promote products derived from
00017  *      this software without specific prior written permission.
00018  *
00019  *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00020  *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00021  *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
00022  *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
00023  *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00026  *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00027  *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00028  *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00029  *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  *******************************************************************************/
00032
00033 /*******************************************************************************/
00051 #ifndef __UIO_H__
00052 #define __UIO_H__
00053
00054 void *setUIO(int uio_index, int length);
00055 int unsetUIO(void *uio_ptr, int length);
00056
00057 #endif // __UIO_H__
```

6.73 library/util.c File Reference

```
#include <libpynq.h>
#include <unistd.h>
```
Include dependency graph for util.c:

Data Structures

- struct pin_state_t

Functions

- void sleep_msec (int msec)
- void mapping_info (void)

6.74 util.c

```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <libpynq.h>
00023 #include <unistd.h>
00024
00025 typedef struct {
00026   char *name;
00027   gpio_direction_t state;
00028   uint8_t channel;
00029   char *level;
00030 } pin_state_t;
00031
00032 void sleep_msec(int msec) {
00033   if (msec > 0)
00034     usleep(msec * 1000);
00035 }
00036
00037 void mapping_info(void) {
00038   const char *const dir[2] = {"Input", "Output"};
00039   printf("Pin\tName\tI/O\tLevel\tChannel\tCh_Name\t\tState\n");
00040   for (int i = 0; i < IO_NUM_PINS; i++) {
00041     pin_state_t pin_array = {
00042         0,
00043     };
00044     pin_array.name = pin_names[i];
00045     pin_array.state = gpio_get_direction(i);
00046     if (gpio_get_level(i) == GPIO_LEVEL_HIGH) {
00047       pin_array.level = "high";
00048     } else if (gpio_get_level(i) == GPIO_LEVEL_LOW) {
00049       pin_array.level = "low";
00050     } else {
00051       pin_array.level = "undef";
00052     }
00053     // get the index of the channel the pin is mapped to, 0 for none
00054     pin_array.channel = switchbox_get_pin(i);
00055
00056     printf("%i\t%s\t%s\t%s\t%u\t", i, pin_array.name, dir[pin_array.state],
00057            pin_array.level, pin_array.channel);
00058
00059     printf("%s\t", switchbox_names[pin_array.channel]);
00060     if (pin_array.channel != SWB_GPIO && pin_array.state != GPIO_DIR_INPUT) {
00061       printf("Invalid\n");
00062     } else {
00063       printf("Valid\n");
00064     }
00065   }
00066 }
```

6.75 library/util.h File Reference

```
#include <stdlib.h>
#include <switchbox.h>
```
Include dependency graph for util.h: This graph shows which files directly or indirectly include this file:

Functions

- void sleep_msec (int msec)
- void mapping_info (void)

6.76 util.h

Go to the documentation of this file.

```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef UTIL_H
00023 #define UTIL_H
00024
00025 #include <stdlib.h>
00026 #include <switchbox.h>
00027
00041 extern void sleep_msec(int msec);
00042
00047 extern void mapping_info(void);
00048
00052 #endif
```

6.77 library/version.c File Reference

```
#include <libpynq.h>
```
Include dependency graph for version.c:

Macros

- #define LIBPYNQ_RELEASE "5EWC0-2023"
- #define LIBPYNQ_VERSION_MAJOR 0
- #define LIBPYNQ_VERSION_MINOR 2
- #define LIBPYNQ_VERSION_PATCH 6
- #define LOG_DOMAIN "version"

Functions

- void print_version (void)
- void check_version (void)

Variables

- const version_t libpynq_version

6.77.1 Macro Definition Documentation

6.77.1.1 LIBPYNQ_RELEASE

```
#define LIBPYNQ_RELEASE "5EWC0-2023"
```

Definition at line 30 of file version.c.

6.77.1.2 LIBPYNQ_VERSION_MAJOR

```
#define LIBPYNQ_VERSION_MAJOR 0
```

Definition at line 31 of file version.c.

6.77.1.3 LIBPYNQ_VERSION_MINOR

```
#define LIBPYNQ_VERSION_MINOR 2
```

Definition at line 32 of file version.c.

6.77.1.4 LIBPYNQ_VERSION_PATCH

```
#define LIBPYNQ_VERSION_PATCH 6
```

Definition at line 33 of file version.c.

6.77.1.5 LOG_DOMAIN

```
#define LOG_DOMAIN "version"
```

Definition at line 42 of file version.c.

6.78 version.c

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #include <libpynq.h>
00023
00024 /***********************
00025  * WARNING
00026  * only change the numbers in these 4 #defs; do not change anything else
00027  * the libpynq version in doxygen ryb.doxy is updated automatically based
00028  * on the next 4 lines
00029  ***********************/
00030 #define LIBPYNQ_RELEASE "5EWC0-2023"
00031 #define LIBPYNQ_VERSION_MAJOR 0
00032 #define LIBPYNQ_VERSION_MINOR 2
00033 #define LIBPYNQ_VERSION_PATCH 6
00034 const version_t libpynq_version = {
00035     LIBPYNQ_RELEASE,
00036     LIBPYNQ_VERSION_MAJOR,
00037     LIBPYNQ_VERSION_MINOR,
00038     LIBPYNQ_VERSION_PATCH,
00039 };
00040
00041 #undef LOG_DOMAIN
00042 #define LOG_DOMAIN "version"
00043
00044 void print_version(void) {
00045   arm_shared t;
00046   version_t volatile *hardwareVersion =
00047       (version_t volatile *)arm_shared_init(&t, axi_version_0, 4096);
00048   printf("Bitstream version: %d.%d.%d\r\n", hardwareVersion->major,
00049          hardwareVersion->minor, hardwareVersion->patch);
00050   printf("Libpynq release %s version %d.%d.%d\r\n", libpynq_version.release,
00051          libpynq_version.major, libpynq_version.minor, libpynq_version.patch);
00052   if (libpynq_version.major != hardwareVersion->major) {
00053     pynq_error(
00054         "ERROR: the bitstream (hardware) and the libpynq library versions "
00055         "are incompatible. Please update your SD-card image and libpynq "
00056        "library.\n");
00057   } else if (libpynq_version.minor > hardwareVersion->minor) {
00058     printf("INFO: the libpynq library is newer than the bitstream (hardware). "
00059            "Please check if there is a newer version of the SD-card image.\n");
00060   } else if (libpynq_version.minor < hardwareVersion->minor) {
00061     printf(
00062         "INFO: the bitstream (hardware) is newer than the libpynq library. "
00063         "Please check if there is a newer version of the libpynq library.\n");
00064   }
00065   arm_shared_close(&t);
00066 }
00067
00068 void check_version(void) {
00069   arm_shared t;
00070   version_t volatile *hardwareVersion =
00071       (version_t volatile *)arm_shared_init(&t, axi_version_0, 4096);
00072   if (libpynq_version.major != hardwareVersion->major) {
00073     print_version();
00074   }
00075   arm_shared_close(&t);
00076 }
```

6.79   library/version.h File Reference

`#include <stdint.h>`

Include dependency graph for version.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct version_t

Functions

- void print_version (void)
- void check_version (void)

Variables

- const version_t libpynq_version

6.80   version.h

[Go to the documentation of this file.](#)
```
00001 /*
00002 Copyright (c) 2023 Eindhoven University of Technology
00003
00004 Permission is hereby granted, free of charge, to any person obtaining a copy
00005 of this software and associated documentation files (the "Software"), to deal
00006 in the Software without restriction, including without limitation the rights
00007 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00008 copies of the Software, and to permit persons to whom the Software is
00009 furnished to do so, subject to the following conditions:
00010
00011 The above copyright notice and this permission notice shall be included in all
00012 copies or substantial portions of the Software.
00013
00014 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00015 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00016 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00017 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00018 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00019 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00020 SOFTWARE.
00021 */
00022 #ifndef VERSION_H
00023 #define VERSION_H
00024
00058 #include <stdint.h>
00059
00063 typedef struct {
00064   uint8_t release[64];
00065   uint32_t major;
00066   uint32_t minor;
00067   uint32_t patch;
00068 } version_t;
00069
00073 extern const version_t libpynq_version;
00074
00080 extern void print_version(void);
00081
00088 extern void check_version(void);
00089
00094 #endif
```

6.81 library/xiic_i.h File Reference

```
#include "xiic.h"
#include "xil_assert.h"
#include "xil_types.h"
#include "xstatus.h"
```
Include dependency graph for xiic_i.h:

Macros

- #define XIIC_I_H /∗ by using protection macros ∗/
- #define XIic_Send10BitAddrByte1(SlaveAddress, Operation)
- #define XIic_Send10BitAddrByte2(SlaveAddress)
- #define XIic_Send7BitAddr(SlaveAddress, Operation)
- #define XIic_DisableIntr(BaseAddress, InterruptMask) XIic_WriteIier((BaseAddress), XIic_ReadIier(Base↩ Address) & ∼(InterruptMask))
- #define XIic_EnableIntr(BaseAddress, InterruptMask) XIic_WriteIier((BaseAddress), XIic_ReadIier(Base↩ Address) | (InterruptMask))
- #define XIic_ClearIntr(BaseAddress, InterruptMask) XIic_WriteIisr((BaseAddress), XIic_ReadIisr(Base↩ Address) & (InterruptMask))
- #define XIic_ClearEnableIntr(BaseAddress, InterruptMask)
- #define XIic_FlushRxFifo(InstancePtr)
- #define XIic_FlushTxFifo(InstancePtr)
- #define XIic_ReadRecvByte(InstancePtr)
- #define XIic_WriteSendByte(InstancePtr)
- #define XIic_SetControlRegister(InstancePtr, ControlRegister, ByteCount)

Functions

- void XIic_TransmitFifoFill (XIic ∗InstancePtr, int Role)

Variables

- XIic_Config XIic_ConfigTable [ ]
- void(∗ XIic_AddrAsSlaveFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_NotAddrAsSlaveFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_RecvSlaveFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_SendSlaveFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_RecvMasterFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_SendMasterFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_ArbLostFuncPtr)(XIic ∗InstancePtr)
- void(∗ XIic_BusNotBusyFuncPtr)(XIic ∗InstancePtr)

6.81.1 Macro Definition Documentation

6.81.1.1 XIic_ClearEnableIntr

```
#define XIic_ClearEnableIntr(
              BaseAddress,
              InterruptMask )
```

Value:
```
{                                                                         \
   XIic_WriteIisr(BaseAddress,                                            \
                 (XIic_ReadIisr(BaseAddress) & (InterruptMask)));         \
                                                                          \
   XIic_WriteIier(BaseAddress,                                            \
                 (XIic_ReadIier(BaseAddress) | (InterruptMask)));         \
   }
```

Definition at line 206 of file xiic_i.h.

6.81.1.2 XIic_ClearIntr

```
#define XIic_ClearIntr(
            BaseAddress,
            InterruptMask )  XIic_WriteIisr((BaseAddress), XIic_ReadIisr(BaseAddress) &
(InterruptMask))
```

Definition at line 187 of file xiic_i.h.

6.81.1.3 XIic_DisableIntr

```
#define XIic_DisableIntr(
            BaseAddress,
            InterruptMask )  XIic_WriteIier((BaseAddress), XIic_ReadIier(BaseAddress) &
∼(InterruptMask))
```

Definition at line 151 of file xiic_i.h.

6.81.1.4 XIic_EnableIntr

```
#define XIic_EnableIntr(
            BaseAddress,
            InterruptMask )  XIic_WriteIier((BaseAddress), XIic_ReadIier(BaseAddress) | (Interrupt↩
Mask))
```

Definition at line 169 of file xiic_i.h.

6.81.1.5 XIic_FlushRxFifo

```
#define XIic_FlushRxFifo(
            InstancePtr )
```

Value:
```
  {                                                                    \
    int LoopCnt;                                                       \
    u8 BytesToRead =                                                   \
        XIic_ReadReg(InstancePtr->BaseAddress, XIIC_RFO_REG_OFFSET) + 1;   \
    for (LoopCnt = 0; LoopCnt < BytesToRead; LoopCnt++) {              \
      XIic_ReadReg(InstancePtr->BaseAddress, XIIC_DRR_REG_OFFSET);     \
    }                                                                  \
  }
```

Definition at line 229 of file xiic_i.h.

6.81.1.6 XIic_FlushTxFifo

```
#define XIic_FlushTxFifo(
            InstancePtr )
```

Value:
```
  ;                                                                    \
  {                                                                    \
    u32 CntlReg = XIic_ReadReg(InstancePtr->BaseAddress, XIIC_CR_REG_OFFSET);  \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_CR_REG_OFFSET,        \
                CntlReg | XIIC_CR_TX_FIFO_RESET_MASK);                 \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_CR_REG_OFFSET, CntlReg);   \
  }
```

Definition at line 253 of file xiic_i.h.

6.81.1.7 XIIC_I_H

```
#define XIIC_I_H /* by using protection macros */
```

This header file contains internal identifiers, which are those shared between XIic components. The identifiers in this file are not intended for use external to the driver.

```
MODIFICATION HISTORY:

Ver   Who  Date      Changes
----- ---- --------  ---------------------------------------------
1.01a rfp  10/19/01 release
1.01c ecm  12/05/02 new rev
1.13a wgr  03/22/07 Converted to new coding style.
2.00a sdm  10/22/09 Converted all register accesses to 32 bit access.
                    Removed the macro XIIC_CLEAR_STATS, user has to
                    use the the XIic_ClearStats API in its place.
                    Removed the macro XIic_mEnterCriticalRegion,
                    XIic_IntrGlobalDisable should be used in its place.
                    Removed the macro XIic_mExitCriticalRegion,
                    XIic_IntrGlobalEnable should be used in its place.
                    Removed the _m prefix from all the macros
                    XIic_mSend10BitAddrByte1 is now XIic_Send10BitAddrByte1
                    XIic_mSend10BitAddrByte2 is now XIic_Send10BitAddrByte2
                    XIic_mSend7BitAddr is now XIic_Send7BitAddr
                    XIic_mDisableIntr is now XIic_DisableIntr
                    XIic_mEnableIntr is now XIic_EnableIntr
                    XIic_mClearIntr is now XIic_ClearIntr
                    XIic_mClearEnableIntr is now XIic_ClearEnableIntr
                    XIic_mFlushRxFifo is now XIic_FlushRxFifo
                    XIic_mFlushTxFifo is now XIic_FlushTxFifo
                    XIic_mReadRecvByte is now XIic_ReadRecvByte
                    XIic_mWriteSendByte is now XIic_WriteSendByte
                    XIic_mSetControlRegister is now XIic_SetControlRegister
2.07a adk   18/04/13 Updated the code to avoid unused variable warnings when
                    compiling with the -Wextra -Wall flags.
                    Changes done in files xiic.c and xiic_i.h. CR:705001
```

Definition at line 51 of file xiic_i.h.

6.81.1.8 XIic_ReadRecvByte

```
#define XIic_ReadRecvByte(
              InstancePtr )
```

Value:
```
  {                                                                      \
    *InstancePtr->RecvBufferPtr++ =                                      \
        XIic_ReadReg(InstancePtr->BaseAddress, XIIC_DRR_REG_OFFSET);     \
    InstancePtr->RecvByteCount--;                                        \
    InstancePtr->Stats.RecvBytes++;                                      \
  }
```

Definition at line 275 of file xiic_i.h.

6.81.1.9 XIic_Send10BitAddrByte1

```
#define XIic_Send10BitAddrByte1(
                SlaveAddress,
                Operation )
```

Value:
```
  {                                                                         \
    u8 LocalAddr = (u8)((SlaveAddress) » 7);                                \
    LocalAddr = (LocalAddr & 0xF6) | 0xF0 | (Operation);                    \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,            \
                  (u32)LocalAddr);                                          \
  }
```

Definition at line 88 of file xiic_i.h.

6.81.1.10 XIic_Send10BitAddrByte2

```
#define XIic_Send10BitAddrByte2(
                SlaveAddress )
```

Value:
```
  XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,             \
                (u32)(SlaveAddress));
```

Definition at line 110 of file xiic_i.h.

6.81.1.11 XIic_Send7BitAddr

```
#define XIic_Send7BitAddr(
                SlaveAddress,
                Operation )
```

Value:
```
  {                                                                         \
    u8 LocalAddr = (u8)(SlaveAddress « 1);                                  \
    LocalAddr = (LocalAddr & 0xFE) | (Operation);                          \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,            \
                  (u32)LocalAddr);                                          \
  }
```

Definition at line 128 of file xiic_i.h.

6.81.1.12 XIic_SetControlRegister

```
#define XIic_SetControlRegister(
                InstancePtr,
                ControlRegister,
                ByteCount )
```

Value:
```
  {                                                                             \
    (ControlRegister) &= ~(XIIC_CR_NO_ACK_MASK | XIIC_CR_DIR_IS_TX_MASK);       \
    if (InstancePtr->Options & XII_SEND_10_BIT_OPTION) {                        \
      (ControlRegister) |= XIIC_CR_DIR_IS_TX_MASK;                              \
    } else {                                                                    \
      if ((ByteCount) == 1) {                                                   \
        (ControlRegister) |= XIIC_CR_NO_ACK_MASK;                               \
      }                                                                         \
    }                                                                           \
  }
```

Definition at line 323 of file xiic_i.h.

6.81.1.13 XIic_WriteSendByte

```
#define XIic_WriteSendByte(
            InstancePtr )
```

Value:
```
  {                                                                \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,   \
                *InstancePtr->SendBufferPtr++);                    \
    InstancePtr->SendByteCount--;                                  \
    InstancePtr->Stats.SendBytes++;                                \
  }
```

Definition at line 296 of file xiic_i.h.

6.81.2 Function Documentation

6.81.2.1 XIic_TransmitFifoFill()

```
void XIic_TransmitFifoFill (
            XIic * InstancePtr,
            int Role )
```

6.81.3 Variable Documentation

6.81.3.1 XIic_AddrAsSlaveFuncPtr

```
void(* XIic_AddrAsSlaveFuncPtr) (XIic *InstancePtr) (
            XIic * InstancePtr )  [extern]
```

6.81.3.2 XIic_ArbLostFuncPtr

```
void(* XIic_ArbLostFuncPtr) (XIic *InstancePtr) (
            XIic * InstancePtr )  [extern]
```

6.81.3.3 XIic_BusNotBusyFuncPtr

```
void(* XIic_BusNotBusyFuncPtr) (XIic *InstancePtr) (
            XIic * InstancePtr )  [extern]
```

6.81.3.4 XIic_ConfigTable

```
XIic_Config XIic_ConfigTable[]  [extern]
```

6.81.3.5 XIic_NotAddrAsSlaveFuncPtr

```
void(* XIic_NotAddrAsSlaveFuncPtr) (XIic *InstancePtr) (
            XIic * InstancePtr )  [extern]
```

6.81.3.6 XIic_RecvMasterFuncPtr

void(* XIic_RecvMasterFuncPtr) (XIic *InstancePtr) (
        XIic * *InstancePtr*) [extern]

6.81.3.7 XIic_RecvSlaveFuncPtr

void(* XIic_RecvSlaveFuncPtr) (XIic *InstancePtr) (
        XIic * *InstancePtr*) [extern]

6.81.3.8 XIic_SendMasterFuncPtr

void(* XIic_SendMasterFuncPtr) (XIic *InstancePtr) (
        XIic * *InstancePtr*) [extern]

6.81.3.9 XIic_SendSlaveFuncPtr

void(* XIic_SendSlaveFuncPtr) (XIic *InstancePtr) (
        XIic * *InstancePtr*) [extern]

6.82 xiic_i.h

Go to the documentation of this file.
```
00001 /*****************************************************************************
00002  * Copyright (C) 2002 - 2021 Xilinx, Inc.  All rights reserved.
00003  * SPDX-License-Identifier: MIT
00004  *****************************************************************************/
00005
00006 /*****************************************************************************/
00050 #ifndef XIIC_I_H /* prevent circular inclusions */
00051 #define XIIC_I_H /* by using protection macros */
00052
00053 #ifdef __cplusplus
00054 extern "C" {
00055 #endif
00056
00057 /***************************** Include Files *********************************/
00058
00059 #include "xiic.h"
00060 #include "xil_assert.h"
00061 #include "xil_types.h"
00062 #include "xstatus.h"
00063
00064 /************************** Constant Definitions *****************************/
00065
00066 /************************** Type Definitions *********************************/
00067
00068 /***************** Macros (Inline Functions) Definitions *********************/
00069
00070 /*****************************************************************************
00071  *
00072  * This macro sends the first byte of the address for a 10 bit address during
00073  * both read and write operations. It takes care of the details to format the
00074  * address correctly.
00075  *
00076  * address = 1111_0xxD   xx = address MSBits
00077  *                        D = Tx direction = 0 = write
00078  *
00079  * @param   SlaveAddress contains the address of the slave to send to.
00080  * @param   Operation indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION
00081  *
00082  * @return  None.
00083  *
```

```
00084  * @note      Signature:
00085  *      void XIic_Send10BitAddrByte1(u16 SlaveAddress, u8 Operation);
00086  *
00087  ******************************************************************************/
00088 #define XIic_Send10BitAddrByte1(SlaveAddress, Operation)                  \
00089   {                                                                       \
00090     u8 LocalAddr = (u8)((SlaveAddress) >> 7);                             \
00091     LocalAddr = (LocalAddr & 0xF6) | 0xF0 | (Operation);                  \
00092     XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,          \
00093                 (u32)LocalAddr);                                          \
00094   }
00095
00096 /******************************************************************************
00097  *
00098  * This macro sends the second byte of the address for a 10 bit address during
00099  * both read and write operations. It takes care of the details to format the
00100  * address correctly.
00101  *
00102  * @param   SlaveAddress contains the address of the slave to send to.
00103  *
00104  * @return  None.
00105  *
00106  * @note       Signature: void XIic_Send10BitAddrByte2(u16
00107  *SlaveAddress, u8 Operation);
00108  *
00109  ******************************************************************************/
00110 #define XIic_Send10BitAddrByte2(SlaveAddress)                             \
00111   XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,            \
00112                 (u32)(SlaveAddress));
00113
00114 /******************************************************************************
00115  *
00116  * This macro sends the address for a 7 bit address during both read and write
00117  * operations. It takes care of the details to format the address correctly.
00118  *
00119  * @param   SlaveAddress contains the address of the slave to send to.
00120  * @param   Operation indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION
00121  *
00122  * @return  None.
00123  *
00124  * @note       Signature:
00125  *      void XIic_Send7BitAddr(u16 SlaveAddress, u8 Operation);
00126  *
00127  ******************************************************************************/
00128 #define XIic_Send7BitAddr(SlaveAddress, Operation)                        \
00129   {                                                                       \
00130     u8 LocalAddr = (u8)(SlaveAddress << 1);                               \
00131     LocalAddr = (LocalAddr & 0xFE) | (Operation);                         \
00132     XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,          \
00133                 (u32)LocalAddr);                                          \
00134   }
00135
00136 /******************************************************************************
00137  *
00138  * This macro disables the specified interrupts in the Interrupt enable
00139  * register.  It is non-destructive in that the register is read and only the
00140  * interrupts specified is changed.
00141  *
00142  * @param   BaseAddress is the base address of the IIC device.
00143  * @param   InterruptMask contains the interrupts to be disabled
00144  *
00145  * @return  None.
00146  *
00147  * @note       Signature:
00148  *      void XIic_DisableIntr(u32 BaseAddress, u32 InterruptMask);
00149  *
00150  ******************************************************************************/
00151 #define XIic_DisableIntr(BaseAddress, InterruptMask)                      \
00152   XIic_WriteIier((BaseAddress), XIic_ReadIier(BaseAddress) & ~(InterruptMask))
00153
00154 /******************************************************************************
00155  *
00156  * This macro enables the specified interrupts in the Interrupt enable
00157  * register.  It is non-destructive in that the register is read and only the
00158  * interrupts specified is changed.
00159  *
00160  * @param   BaseAddress is the base address of the IIC device.
00161  * @param   InterruptMask contains the interrupts to be disabled
00162  *
00163  * @return  None.
00164  *
00165  * @note       Signature:
00166  *      void XIic_EnableIntr(u32 BaseAddress, u32 InterruptMask);
00167  *
00168  ******************************************************************************/
00169 #define XIic_EnableIntr(BaseAddress, InterruptMask)                       \
00170   XIic_WriteIier((BaseAddress), XIic_ReadIier(BaseAddress) | (InterruptMask))
```

```
00171
00172 /****************************************************************************
00173  *
00174  * This macro clears the specified interrupt in the Interrupt status
00175  * register.  It is non-destructive in that the register is read and only the
00176  * interrupt specified is cleared.  Clearing an interrupt acknowledges it.
00177  *
00178  * @param   BaseAddress is the base address of the IIC device.
00179  * @param   InterruptMask contains the interrupts to be disabled
00180  *
00181  * @return  None.
00182  *
00183  * @note        Signature:
00184  *      void XIic_ClearIntr(u32 BaseAddress, u32 InterruptMask);
00185  *
00186  ****************************************************************************/
00187 #define XIic_ClearIntr(BaseAddress, InterruptMask)                        \
00188   XIic_WriteIisr((BaseAddress), XIic_ReadIisr(BaseAddress) & (InterruptMask))
00189
00190 /****************************************************************************
00191  *
00192  * This macro clears and enables the specified interrupt in the Interrupt
00193  * status and enable registers.  It is non-destructive in that the registers are
00194  * read and only the interrupt specified is modified.
00195  * Clearing an interrupt acknowledges it.
00196  *
00197  * @param   BaseAddress is the base address of the IIC device.
00198  * @param   InterruptMask contains the interrupts to be cleared and enabled
00199  *
00200  * @return  None.
00201  *
00202  * @note        Signature:
00203  *      void XIic_ClearEnableIntr(u32 BaseAddress, u32 InterruptMask);
00204  *
00205  ****************************************************************************/
00206 #define XIic_ClearEnableIntr(BaseAddress, InterruptMask)                  \
00207   {                                                                       \
00208     XIic_WriteIisr(BaseAddress,                                           \
00209                    (XIic_ReadIisr(BaseAddress) & (InterruptMask)));       \
00210                                                                           \
00211     XIic_WriteIier(BaseAddress,                                           \
00212                    (XIic_ReadIier(BaseAddress) | (InterruptMask)));       \
00213   }
00214
00215 /****************************************************************************
00216  *
00217  * This macro flushes the receive FIFO such that all bytes contained within it
00218  * are discarded.
00219  *
00220  * @param   InstancePtr is a pointer to the IIC instance containing the FIFO
00221  *      to be flushed.
00222  *
00223  * @return  None.
00224  *
00225  * @note        Signature:
00226  *      void XIic_FlushRxFifo(XIic *InstancePtr);
00227  *
00228  ****************************************************************************/
00229 #define XIic_FlushRxFifo(InstancePtr)                                     \
00230   {                                                                       \
00231     int LoopCnt;                                                          \
00232     u8 BytesToRead =                                                      \
00233       XIic_ReadReg(InstancePtr->BaseAddress, XIIC_RFO_REG_OFFSET) + 1;    \
00234     for (LoopCnt = 0; LoopCnt < BytesToRead; LoopCnt++) {                 \
00235       XIic_ReadReg(InstancePtr->BaseAddress, XIIC_DRR_REG_OFFSET);        \
00236     }                                                                     \
00237   }
00238
00239 /****************************************************************************
00240  *
00241  * This macro flushes the transmit FIFO such that all bytes contained within it
00242  * are discarded.
00243  *
00244  * @param   InstancePtr is a pointer to the IIC instance containing the FIFO
00245  *      to be flushed.
00246  *
00247  * @return  None.
00248  *
00249  * @note        Signature:
00250  *      void XIic_FlushTxFifo(XIic *InstancePtr);
00251  *
00252  ****************************************************************************/
00253 #define XIic_FlushTxFifo(InstancePtr)                                     \
00254   ;                                                                       \
00255   {                                                                       \
00256     u32 CntlReg = XIic_ReadReg(InstancePtr->BaseAddress, XIIC_CR_REG_OFFSET);  \
00257     XIic_WriteReg(InstancePtr->BaseAddress, XIIC_CR_REG_OFFSET,           \
```

```
00258                 CntlReg | XIIC_CR_TX_FIFO_RESET_MASK);                    \
00259     XIic_WriteReg(InstancePtr->BaseAddress, XIIC_CR_REG_OFFSET, CntlReg);  \
00260   }
00261
00262 /********************************************************************************
00263 *
00264 * This macro reads the next available received byte from the receive FIFO
00265 * and updates all the data structures to reflect it.
00266 *
00267 * @param   InstancePtr is a pointer to the IIC instance to be operated on.
00268 *
00269 * @return  None.
00270 *
00271 * @note        Signature:
00272 *       void XIic_ReadRecvByte(XIic *InstancePtr);
00273 *
00274 ********************************************************************************/
00275 #define XIic_ReadRecvByte(InstancePtr)                                       \
00276   {                                                                          \
00277     *InstancePtr->RecvBufferPtr++ =                                          \
00278         XIic_ReadReg(InstancePtr->BaseAddress, XIIC_DRR_REG_OFFSET);         \
00279     InstancePtr->RecvByteCount--;                                            \
00280     InstancePtr->Stats.RecvBytes++;                                          \
00281   }
00282
00283 /********************************************************************************
00284 *
00285 * This macro writes the next byte to be sent to the transmit FIFO
00286 * and updates all the data structures to reflect it.
00287 *
00288 * @param   InstancePtr is a pointer to the IIC instance to be operated on.
00289 *
00290 * @return  None.
00291 *
00292 * @note        Signature:
00293 *       void XIic_WriteSendByte(XIic *InstancePtr);
00294 *
00295 ********************************************************************************/
00296 #define XIic_WriteSendByte(InstancePtr)                                      \
00297   {                                                                          \
00298     XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,             \
00299                    *InstancePtr->SendBufferPtr++);                           \
00300     InstancePtr->SendByteCount--;                                            \
00301     InstancePtr->Stats.SendBytes++;                                          \
00302   }
00303
00304 /********************************************************************************
00305 *
00306 * This macro sets up the control register for a master receive operation.
00307 * A write is necessary if a 10 bit operation is being performed.
00308 *
00309 * @param   InstancePtr is a pointer to the IIC instance to be operated on.
00310 * @param   ControlRegister contains the contents of the IIC device control
00311 *       register
00312 * @param   ByteCount contains the number of bytes to be received for the
00313 *       master receive operation
00314 *
00315 * @return  None.
00316 *
00317 * @note        Signature:
00318 *       void XIic_SetControlRegister(XIic *InstancePtr,
00319 *                       u8 ControlRegister,
00320 *                       int ByteCount);
00321 *
00322 ********************************************************************************/
00323 #define XIic_SetControlRegister(InstancePtr, ControlRegister, ByteCount)     \
00324   {                                                                          \
00325     (ControlRegister) &= ~(XIIC_CR_NO_ACK_MASK | XIIC_CR_DIR_IS_TX_MASK);     \
00326     if (InstancePtr->Options & XII_SEND_10_BIT_OPTION) {                     \
00327       (ControlRegister) |= XIIC_CR_DIR_IS_TX_MASK;                           \
00328     } else {                                                                 \
00329       if ((ByteCount) == 1) {                                                \
00330         (ControlRegister) |= XIIC_CR_NO_ACK_MASK;                            \
00331       }                                                                      \
00332     }                                                                        \
00333   }
00334
00335 /*********************** Function Prototypes ****************************/
00336
00337 extern XIic_Config XIic_ConfigTable[];
00338
00339 /* The following variables are shared across files of the driver and
00340 * are function pointers that are necessary to break dependencies allowing
00341 * optional parts of the driver to be used without condition compilation
00342 */
00343 extern void (*XIic_AddrAsSlaveFuncPtr)(XIic *InstancePtr);
00344 extern void (*XIic_NotAddrAsSlaveFuncPtr)(XIic *InstancePtr);
```

```
00345 extern void (*XIic_RecvSlaveFuncPtr)(XIic *InstancePtr);
00346 extern void (*XIic_SendSlaveFuncPtr)(XIic *InstancePtr);
00347 extern void (*XIic_RecvMasterFuncPtr)(XIic *InstancePtr);
00348 extern void (*XIic_SendMasterFuncPtr)(XIic *InstancePtr);
00349 extern void (*XIic_ArbLostFuncPtr)(XIic *InstancePtr);
00350 extern void (*XIic_BusNotBusyFuncPtr)(XIic *InstancePtr);
00351
00352 void XIic_TransmitFifoFill(XIic *InstancePtr, int Role);
00353
00354 #ifdef __cplusplus
00355 }
00356 #endif
00357
00358 #endif /* end of protection macro */
```

6.83 library/xiic_l.c File Reference

```
#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include "xiic_l.h"
#include "xil_types.h"
```
Include dependency graph for xiic_l.c:

Macros

- #define _DEFAULT_SOURCE
- #define IIC_TIMEOUT 5

Functions

- unsigned XIic_Recv (UINTPTR BaseAddress, u8 Address, u8 ∗BufferPtr, unsigned ByteCount, u8 Option)
- unsigned XIic_Send (UINTPTR BaseAddress, u8 Address, u8 ∗BufferPtr, unsigned ByteCount, u8 Option)
- u32 XIic_CheckIsBusBusy (UINTPTR BaseAddress)
- u32 XIic_WaitBusFree (UINTPTR BaseAddress)

6.83.1 Macro Definition Documentation

6.83.1.1 _DEFAULT_SOURCE

```
#define _DEFAULT_SOURCE
```

This file contains low-level driver functions that can be used to access the device in normal and dynamic controller mode. The user should refer to the hardware device specification for more details of the device operation.

```
MODIFICATION HISTORY:

Ver   Who  Date      Changes
----- ---  -------   -----------------------------------------------
1.01b jhl  05/13/02  First release
1.01b jhl  10/14/02  Corrected bug in the receive function, the setup of the
                     interrupt status mask was not being done in the loop such
                     that a read would sometimes fail on the last byte because
                     the transmit error which should have been ignored was
                     being used.  This would leave an extra byte in the FIFO
```

```
                        and the bus throttled such that the next operation would
                        also fail.  Also updated the receive function to not
                        disable the device after the last byte until after the
                        bus transitions to not busy which is more consistent
                        with the expected behavior.
1.01c ecm  12/05/02 new rev
1.02a mta  03/09/06 Implemented Repeated Start in the Low Level Driver.
1.03a mta  04/04/06 Implemented Dynamic IIC core routines.
1.03a ecm  06/15/06 Fixed the hang in low_level_eeprom_test with -O0
                        Added polling loops for BNB to allow the slave to
                        respond correctly. Also added polling loop prior
                        to reset in _Recv.
1.13a wgr  03/22/07 Converted to new coding style.
1.13b ecm  11/29/07 added BB polling loops to the DynSend and DynRecv
        routines to handle the race condition with BNB in IISR.
2.00a sdm  10/22/09 Converted all register accesses to 32 bit access.
        Updated to use the HAL APIs/macros.
        Some of the macros have been renamed to remove _m from
        the name and Some of the macros have been renamed to be
        consistent, see the xiic_i.h and xiic_l.h files for
        further information.
2.02a sdm  10/08/10 Updated to disable the device at the end of the transfer,
        only when addressed as slave in XIic_Send for CR565373.
2.04a sdm  07/22/11 Removed a compiler warning by adding parenthesis around &
        at line 479.
2.08a adk  29/07/13 In Low level driver In repeated start condition the
        Direction of Tx bit must be disabled in Receive
        condition It Fixes the CR:685759 Changes are done
        in the function XIic_Recv.
3.2   sk   11/10/15 Used UINTPTR instead of u32 for Baseaddress CR# 867425.
                        Changed the prototypes of RecvData, SendData,
                        DynRecvData, DynSendData APIs.
3.2 sd   18/02/16 In Low level driver in repeated start condition
                        NACK for last byte is added. Changes are done in
                        XIic_Recv for CR# 862303
3.3   sk   06/17/16 Added bus busy checks for slave send/recv and master
                        send/recv.
3.3   als  06/27/16 Added Low-level XIic_CheckIsBusBusy API.
3.3   als  06/27/16 Added low-level XIic_WaitBusFree API.
3.4 nk   16/11/16 Reduced sleeping time in Bus-busy check.
3.5   sd   08/29/18 Fix bus busy check for the NACK case.
```

Definition at line 71 of file xiic_l.c.

6.83.1.2 IIC_TIMEOUT

```
#define IIC_TIMEOUT 5
```

Definition at line 76 of file xiic_l.c.

6.83.2 Function Documentation

6.83.2.1 XIic_CheckIsBusBusy()

```
u32 XIic_CheckIsBusBusy (
            UINTPTR BaseAddress )
```

Definition at line 614 of file xiic_l.c.

Here is the caller graph for this function:

6.83.2.2 XIic_Recv()

```
unsigned XIic_Recv (
        UINTPTR BaseAddress,
        u8 Address,
        u8 * BufferPtr,
        unsigned ByteCount,
        u8 Option )
```

Receive data as a master on the IIC bus. This function receives the data using polled I/O and blocks until the data has been received. It only supports 7 bit addressing mode of operation. This function returns zero if bus is busy.

Parameters

BaseAddress	contains the base address of the IIC device.
Address	contains the 7 bit IIC address of the device to send the specified data to.
BufferPtr	points to the data to be sent.
ByteCount	is the number of bytes to be sent.
Option	indicates whether to hold or free the bus after reception of data, XIIC_STOP = end with STOP condition, XIIC_REPEATED_START = don't end with STOP condition.

Returns

> The number of bytes received.

Note

> None.

Definition at line 117 of file xiic_l.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.83.2.3 XIic_Send()

```
unsigned XIic_Send (
        UINTPTR BaseAddress,
        u8 Address,
        u8 * BufferPtr,
        unsigned ByteCount,
        u8 Option )
```

Send data as a master on the IIC bus. This function sends the data using polled I/O and blocks until the data has been sent. It only supports 7 bit addressing mode of operation. This function returns zero if bus is busy.

Parameters

BaseAddress	contains the base address of the IIC device.
Address	contains the 7 bit IIC address of the device to send the specified data to.
BufferPtr	points to the data to be sent.
ByteCount	is the number of bytes to be sent.
Option	indicates whether to hold or free the bus after transmitting the data.

Returns

>   The number of bytes sent.

Note

>   None.

Definition at line 373 of file xiic_l.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.83.2.4 XIic_WaitBusFree()

```
u32 XIic_WaitBusFree (
            UINTPTR BaseAddress )
```

This function will wait until the I2C bus is free or timeout.

Parameters

BaseAddress	contains the base address of the I2C device.

Returns

>   • XST_SUCCESS if the I2C bus was freed before the timeout.
>   • XST_FAILURE otherwise.

Note

>   None.

Definition at line 638 of file xiic_l.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.84 xiic_l.c

Go to the documentation of this file.
```
00001 /******************************************************************************
00002  * Copyright (C) 2002 - 2021 Xilinx, Inc.  All rights reserved.
00003  * SPDX-License-Identifier: MIT
00004  ******************************************************************************/
00005
00006 /*****************************************************************************/
00070 /*************************** Include Files ****************************/
00071 #define _DEFAULT_SOURCE
00072 #include <stdio.h>
00073 #include <time.h>
00074 #include <unistd.h>
00075
00076 #define IIC_TIMEOUT 5
00077
00078 #include "xiic_l.h"
00079 #include "xil_types.h"
```

```
00080
00081 /*************************** Constant Definitions ****************************/
00082
00083 /*************************** Type Definitions ********************************/
00084
00085 /***************** Macros (Inline Functions) Definitions *********************/
00086
00087 /*************************** Function Prototypes *****************************/
00088
00089 static unsigned RecvData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
00090                         u8 Option);
00091 static unsigned SendData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
00092                         u8 Option);
00093
00094 /*************************** Variable Definitions ****************************/
00095
00096 /*****************************************************************************/
00117 unsigned XIic_Recv(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
00118                     unsigned ByteCount, u8 Option) {
00119   u32 CntlReg;
00120   unsigned RemainingByteCount;
00121   volatile u32 StatusReg;
00122
00123   /* Tx error is enabled in case the address (7 or 10) has no device to
00124    * answer with Ack. When only one byte of data, must set NO ACK before
00125    * address goes out therefore Tx error must not be enabled as it will go
00126    * off immediately and the Rx full interrupt will be checked.  If full,
00127    * then the one byte was received and the Tx error will be disabled
00128    * without sending an error callback msg
00129    */
00130   XIic_ClearIisr(BaseAddress, XIIC_INTR_RX_FULL_MASK | XIIC_INTR_TX_ERROR_MASK |
00131                               XIIC_INTR_ARB_LOST_MASK);
00132
00133   /* Set receive FIFO occupancy depth for 1 byte (zero based) */
00134   XIic_WriteReg(BaseAddress, XIIC_RFD_REG_OFFSET, 0);
00135
00136   /* Check to see if already Master on the Bus.
00137    * If Repeated Start bit is not set send Start bit by setting MSMS bit
00138    * else Send the address
00139    */
00140   CntlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
00141   if ((CntlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
00142     /* 7 bit slave address, send the address for a read operation
00143      * and set the state to indicate the address has been sent
00144      */
00145     XIic_Send7BitAddress(BaseAddress, Address, XIIC_READ_OPERATION);
00146
00147     /* MSMS gets set after putting data in FIFO. Start the master
00148      * receive operation by setting CR Bits MSMS to Master, if the
00149      * buffer is only one byte, then it should not be acknowledged
00150      * to indicate the end of data
00151      */
00152     CntlReg = XIIC_CR_MSMS_MASK | XIIC_CR_ENABLE_DEVICE_MASK;
00153     if (ByteCount == 1) {
00154       CntlReg |= XIIC_CR_NO_ACK_MASK;
00155     }
00156
00157     /* Write out the control register to start receiving data and
00158      * call the function to receive each byte into the buffer
00159      */
00160     XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, CntlReg);
00161
00162     /* Clear the latched interrupt status for the bus not busy bit
00163      * which must be done while the bus is busy
00164      */
00165     StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00166
00167     while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) == 0) {
00168       StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00169     }
00170
00171     XIic_ClearIisr(BaseAddress, XIIC_INTR_BNB_MASK);
00172   } else {
00173     /* Before writing 7bit slave address the Direction of Tx bit
00174      * must be disabled
00175      */
00176     CntlReg &= ~XIIC_CR_DIR_IS_TX_MASK;
00177     if (ByteCount == 1) {
00178       CntlReg |= XIIC_CR_NO_ACK_MASK;
00179     }
00180     XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, CntlReg);
00181     /* Already owns the Bus indicating that its a Repeated Start
00182      * call. 7 bit slave address, send the address for a read
00183      * operation and set the state to indicate the address has been
00184      * sent
00185      */
00186     XIic_Send7BitAddress(BaseAddress, Address, XIIC_READ_OPERATION);
```

```
00187    }
00188    /* Try to receive the data from the IIC bus */
00189
00190    RemainingByteCount = RecvData(BaseAddress, BufferPtr, ByteCount, Option);
00191
00192    CntlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
00193    if ((CntlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
00194      /* The receive is complete, disable the IIC device if the Option
00195       * is to release the Bus after Reception of data and return the
00196       * number of bytes that was received
00197       */
00198      XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, 0);
00199    }
00200
00201    /* Wait until I2C bus is freed, exit if timed out. */
00202    if (XIic_WaitBusFree(BaseAddress) != XST_SUCCESS) {
00203      return 0;
00204    }
00205
00206    /* Return the number of bytes that was received */
00207    return ByteCount - RemainingByteCount;
00208  }
00209
00210  /*****************************************************************************
00211   *
00212   * Receive the specified data from the device that has been previously addressed
00213   * on the IIC bus.  This function assumes that the 7 bit address has been sent
00214   * and it should wait for the transmit of the address to complete.
00215   *
00216   * @param   BaseAddress contains the base address of the IIC device.
00217   * @param   BufferPtr points to the buffer to hold the data that is
00218   *      received.
00219   * @param   ByteCount is the number of bytes to be received.
00220   * @param   Option indicates whether to hold or free the bus after reception
00221   *      of data, XIIC_STOP = end with STOP condition,
00222   *      XIIC_REPEATED_START = don't end with STOP condition.
00223   *
00224   * @return  The number of bytes remaining to be received.
00225   *
00226   * @note
00227   *
00228   * This function does not take advantage of the receive FIFO because it is
00229   * designed for minimal code space and complexity.  It contains loops that
00230   * that could cause the function not to return if the hardware is not working.
00231   *
00232   * This function assumes that the calling function will disable the IIC device
00233   * after this function returns.
00234   *
00235   *****************************************************************************/
00236  static unsigned RecvData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
00237                           u8 Option) {
00238    u32 CntlReg;
00239    u32 IntrStatusMask;
00240    u32 IntrStatus;
00241
00242    /* Attempt to receive the specified number of bytes on the IIC bus */
00243
00244    while (ByteCount > 0) {
00245      /* Setup the mask to use for checking errors because when
00246       * receiving one byte OR the last byte of a multibyte message an
00247       * error naturally occurs when the no ack is done to tell the
00248       * slave the last byte
00249       */
00250      if (ByteCount == 1) {
00251        IntrStatusMask = XIIC_INTR_ARB_LOST_MASK | XIIC_INTR_BNB_MASK;
00252      } else {
00253        IntrStatusMask = XIIC_INTR_ARB_LOST_MASK | XIIC_INTR_TX_ERROR_MASK |
00254                         XIIC_INTR_BNB_MASK;
00255      }
00256
00257      /* Wait for the previous transmit and the 1st receive to
00258       * complete by checking the interrupt status register of the
00259       * IPIF
00260       */
00261      while (1) {
00262        IntrStatus = XIic_ReadIisr(BaseAddress);
00263        if (IntrStatus & XIIC_INTR_RX_FULL_MASK) {
00264          break;
00265        }
00266        /* Check the transmit error after the receive full
00267         * because when sending only one byte transmit error
00268         * will occur because of the no ack to indicate the end
00269         * of the data
00270         */
00271        if (IntrStatus & IntrStatusMask) {
00272          return ByteCount;
00273        }
```

```
00274       }
00275
00276       CntlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
00277
00278       /* Special conditions exist for the last two bytes so check for
00279        * them. Note that the control register must be setup for these
00280        * conditions before the data byte which was already received is
00281        * read from the receive FIFO (while the bus is throttled
00282        */
00283       if (ByteCount == 1) {
00284         if (Option == XIIC_STOP) {
00285
00286           /* If the Option is to release the bus after the
00287            * last data byte, it has already been read and
00288            * no ack has been done, so clear MSMS while
00289            * leaving the device enabled so it can get off
00290            * the IIC bus appropriately with a stop
00291            */
00292           XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00293                     XIIC_CR_ENABLE_DEVICE_MASK);
00294         }
00295       }
00296
00297       /* Before the last byte is received, set NOACK to tell the slave
00298        * IIC device that it is the end, this must be done before
00299        * reading the byte from the FIFO
00300        */
00301       if (ByteCount == 2) {
00302         /* Write control reg with NO ACK allowing last byte to
00303          * have the No ack set to indicate to slave last byte
00304          * read
00305          */
00306         XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00307                     CntlReg | XIIC_CR_NO_ACK_MASK);
00308       }
00309
00310       /* Read in data from the FIFO and unthrottle the bus such that
00311        * the next byte is read from the IIC bus
00312        */
00313       *BufferPtr++ = (u8)XIic_ReadReg(BaseAddress, XIIC_DRR_REG_OFFSET);
00314
00315       if ((ByteCount == 1) && (Option == XIIC_REPEATED_START)) {
00316
00317         /* RSTA bit should be set only when the FIFO is
00318          * completely Empty.
00319          */
00320         XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00321                     XIIC_CR_ENABLE_DEVICE_MASK | XIIC_CR_MSMS_MASK |
00322                       XIIC_CR_REPEATED_START_MASK);
00323       }
00324
00325       /* Clear the latched interrupt status so that it will be updated
00326        * with the new state when it changes, this must be done after
00327        * the receive register is read
00328        */
00329       XIic_ClearIisr(BaseAddress, XIIC_INTR_RX_FULL_MASK |
00330                                   XIIC_INTR_TX_ERROR_MASK |
00331                                   XIIC_INTR_ARB_LOST_MASK);
00332     ByteCount--;
00333   }
00334
00335   if (Option == XIIC_STOP) {
00336
00337     /* If the Option is to release the bus after Reception of data,
00338      * wait for the bus to transition to not busy before returning,
00339      * the IIC device cannot be disabled until this occurs. It
00340      * should transition as the MSMS bit of the control register was
00341      * cleared before the last byte was read from the FIFO
00342      */
00343     while (1) {
00344       if (XIic_ReadIisr(BaseAddress) & XIIC_INTR_BNB_MASK) {
00345         break;
00346       }
00347     }
00348   }
00349
00350   return ByteCount;
00351 }
00352
00353 /*****************************************************************************/
00373 unsigned XIic_Send(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
00374                 unsigned ByteCount, u8 Option) {
00375   unsigned RemainingByteCount;
00376   u32 ControlReg;
00377   volatile u32 StatusReg;
00378
00379   /* Wait until I2C bus is freed, exit if timed out. */
```

```
00380    if (XIic_WaitBusFree(BaseAddress) != XST_SUCCESS) {
00381      return 0;
00382    }
00383
00384    /* Check to see if already Master on the Bus.
00385     * If Repeated Start bit is not set send Start bit by setting
00386     * MSMS bit else Send the address.
00387     */
00388    ControlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
00389    if ((ControlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
00390      /*
00391       * Put the address into the FIFO to be sent and indicate
00392       * that the operation to be performed on the bus is a
00393       * write operation
00394       */
00395      XIic_Send7BitAddress(BaseAddress, Address, XIIC_WRITE_OPERATION);
00396      /* Clear the latched interrupt status so that it will
00397       * be updated with the new state when it changes, this
00398       * must be done after the address is put in the FIFO
00399       */
00400      XIic_ClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK |
00401                                  XIIC_INTR_TX_ERROR_MASK |
00402                                  XIIC_INTR_ARB_LOST_MASK);
00403
00404      /*
00405       * MSMS must be set after putting data into transmit FIFO,
00406       * indicate the direction is transmit, this device is master
00407       * and enable the IIC device
00408       */
00409      XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00410                    XIIC_CR_MSMS_MASK | XIIC_CR_DIR_IS_TX_MASK |
00411                       XIIC_CR_ENABLE_DEVICE_MASK);
00412
00413      /*
00414       * Clear the latched interrupt
00415       * status for the bus not busy bit which must be done while
00416       * the bus is busy
00417       */
00418      time_t s = time(NULL);
00419      StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00420      while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) == 0) {
00421        StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00422        time_t n = time(NULL);
00423        if ((n - s) > IIC_TIMEOUT) {
00424          printf("IIC timeout bus not busy.\n");
00425          return 0;
00426        }
00427      }
00428
00429      XIic_ClearIisr(BaseAddress, XIIC_INTR_BNB_MASK);
00430    } else {
00431      /*
00432       * Already owns the Bus indicating that its a Repeated Start
00433       * call. 7 bit slave address, send the address for a write
00434       * operation and set the state to indicate the address has
00435       * been sent.
00436       */
00437      XIic_Send7BitAddress(BaseAddress, Address, XIIC_WRITE_OPERATION);
00438    }
00439
00440    /* Send the specified data to the device on the IIC bus specified by the
00441     * the address
00442     */
00443    RemainingByteCount = SendData(BaseAddress, BufferPtr, ByteCount, Option);
00444
00445    ControlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
00446    if ((ControlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
00447      /*
00448       * The Transmission is completed, disable the IIC device if
00449       * the Option is to release the Bus after transmission of data
00450       * and return the number of bytes that was received. Only wait
00451       * if master, if addressed as slave just reset to release
00452       * the bus.
00453       */
00454      if ((ControlReg & XIIC_CR_MSMS_MASK) != 0) {
00455        XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00456                      (ControlReg & ~XIIC_CR_MSMS_MASK));
00457      }
00458
00459      if ((XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET) &
00460          XIIC_SR_ADDR_AS_SLAVE_MASK) != 0) {
00461        XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, 0);
00462      } else {
00463        StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00464        while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) != 0) {
00465          StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00466        }
```

```
00467      }
00468    }
00469
00470    return ByteCount - RemainingByteCount;
00471 }
00472
00473 /******************************************************************************
00474  *
00475  * Send the specified buffer to the device that has been previously addressed
00476  * on the IIC bus.  This function assumes that the 7 bit address has been sent
00477  * and it should wait for the transmit of the address to complete.
00478  *
00479  * @param   BaseAddress contains the base address of the IIC device.
00480  * @param   BufferPtr points to the data to be sent.
00481  * @param   ByteCount is the number of bytes to be sent.
00482  * @param   Option indicates whether to hold or free the bus after
00483  *      transmitting the data.
00484  *
00485  * @return  The number of bytes remaining to be sent.
00486  *
00487  * @note
00488  *
00489  * This function does not take advantage of the transmit FIFO because it is
00490  * designed for minimal code space and complexity.  It contains loops that
00491  * that could cause the function not to return if the hardware is not working.
00492  *
00493  ******************************************************************************/
00494 static unsigned SendData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
00495                          u8 Option) {
00496   u32 IntrStatus;
00497
00498   /*
00499    * Send the specified number of bytes in the specified buffer by polling
00500    * the device registers and blocking until complete
00501    */
00502   while (ByteCount > 0) {
00503     /*
00504      * Wait for the transmit to be empty before sending any more
00505      * data by polling the interrupt status register
00506      */
00507     while (1) {
00508       IntrStatus = XIic_ReadIisr(BaseAddress);
00509
00510       if (IntrStatus & (XIIC_INTR_TX_ERROR_MASK | XIIC_INTR_ARB_LOST_MASK |
00511                         XIIC_INTR_BNB_MASK)) {
00512         return ByteCount;
00513       }
00514
00515       if (IntrStatus & XIIC_INTR_TX_EMPTY_MASK) {
00516         break;
00517       }
00518     }
00519     /* If there is more than one byte to send then put the
00520      * next byte to send into the transmit FIFO
00521      */
00522     if (ByteCount > 1) {
00523       XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, *BufferPtr++);
00524     } else {
00525       if (Option == XIIC_STOP) {
00526         /*
00527          * If the Option is to release the bus after
00528          * the last data byte, Set the stop Option
00529          * before sending the last byte of data so
00530          * that the stop Option will be generated
00531          * immediately following the data. This is
00532          * done by clearing the MSMS bit in the
00533          * control register.
00534          */
00535         XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00536                       XIIC_CR_ENABLE_DEVICE_MASK | XIIC_CR_DIR_IS_TX_MASK);
00537       }
00538
00539       /*
00540        * Put the last byte to send in the transmit FIFO
00541        */
00542       XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, *BufferPtr++);
00543
00544       if (Option == XIIC_REPEATED_START) {
00545         XIic_ClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK);
00546         /*
00547          * Wait for the transmit to be empty before
00548          * setting RSTA bit.
00549          */
00550         while (1) {
00551           IntrStatus = XIic_ReadIisr(BaseAddress);
00552           if (IntrStatus & XIIC_INTR_TX_EMPTY_MASK) {
00553             /*
```

```
00554                  * RSTA bit should be set only
00555                  * when the FIFO is completely
00556                  * Empty.
00557                  */
00558                 XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
00559                             XIIC_CR_REPEATED_START_MASK |
00560                                 XIIC_CR_ENABLE_DEVICE_MASK |
00561                                 XIIC_CR_DIR_IS_TX_MASK | XIIC_CR_MSMS_MASK);
00562                 break;
00563             }
00564         }
00565       }
00566     }
00567
00568     /*
00569      * Clear the latched interrupt status register and this must be
00570      * done after the transmit FIFO has been written to or it won't
00571      * clear
00572      */
00573     XIic_ClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK);
00574
00575     /*
00576      * Update the byte count to reflect the byte sent and clear
00577      * the latched interrupt status so it will be updated for the
00578      * new state
00579      */
00580     ByteCount--;
00581   }
00582
00583   if (Option == XIIC_STOP) {
00584     /*
00585      * If the Option is to release the bus after transmission of
00586      * data, Wait for the bus to transition to not busy before
00587      * returning, the IIC device cannot be disabled until this
00588      * occurs. Note that this is different from a receive operation
00589      * because the stop Option causes the bus to go not busy.
00590      */
00591     while (1) {
00592       if (XIic_ReadIisr(BaseAddress) & XIIC_INTR_BNB_MASK) {
00593         break;
00594       }
00595     }
00596   }
00597
00598   return ByteCount;
00599 }
00600 /****************************************************************************
00601  *
00602  *
00603  * This is a function which tells whether the I2C bus is busy or free.
00604  *
00605  * @param   BaseAddr is the base address of the I2C core to work on.
00606  *
00607  * @return
00608  *      - TRUE if the bus is busy.
00609  *      - FALSE if the bus is NOT busy.
00610  *
00611  * @note        None.
00612  *
00613  ****************************************************************************/
00614 u32 XIic_CheckIsBusBusy(UINTPTR BaseAddress) {
00615   u32 StatusReg;
00616
00617   StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
00618   if (StatusReg & XIIC_SR_BUS_BUSY_MASK) {
00619     return TRUE;
00620   } else {
00621     return FALSE;
00622   }
00623 }
00624
00625 /****************************************************************************/
00638 u32 XIic_WaitBusFree(UINTPTR BaseAddress) {
00639   u32 BusyCount = 0;
00640
00641   while (XIic_CheckIsBusBusy(BaseAddress)) {
00642     if (BusyCount++ > 10000) {
00643       return XST_FAILURE;
00644     }
00645     usleep(100);
00646   }
00647
00648   return XST_SUCCESS;
00649 }
```

6.85 library/xiic_l.h File Reference

```
#include "xil_io.h"
#include "xil_types.h"
```
Include dependency graph for xiic_l.h: This graph shows which files directly or indirectly include this file:

Macros

- #define XIIC_L_H /∗ by using protection macros ∗/

Register Map

Register offsets for the XIic device.

- #define XIIC_DGIER_OFFSET 0x1C
- #define XIIC_IISR_OFFSET 0x20
- #define XIIC_IIER_OFFSET 0x28
- #define XIIC_RESETR_OFFSET 0x40
- #define XIIC_CR_REG_OFFSET 0x100
- #define XIIC_SR_REG_OFFSET 0x104
- #define XIIC_DTR_REG_OFFSET 0x108
- #define XIIC_DRR_REG_OFFSET 0x10C
- #define XIIC_ADR_REG_OFFSET 0x110
- #define XIIC_TFO_REG_OFFSET 0x114
- #define XIIC_RFO_REG_OFFSET 0x118
- #define XIIC_TBA_REG_OFFSET 0x11C
- #define XIIC_RFD_REG_OFFSET 0x120
- #define XIIC_GPO_REG_OFFSET 0x124

Device Global Interrupt Enable Register masks (CR) mask(s)

- #define XIIC_GINTR_ENABLE_MASK 0x80000000

IIC Device Interrupt Status/Enable (INTR) Register Masks

Interrupt Status Register (IISR)

This register holds the interrupt status flags for the Spi device.

Interrupt Enable Register (IIER)

This register is used to enable interrupt sources for the IIC device. Writing a '1' to a bit in this register enables the corresponding Interrupt. Writing a '0' to a bit in this register disables the corresponding Interrupt.

IISR/IIER registers have the same bit definitions and are only defined once.

- #define XIIC_INTR_ARB_LOST_MASK 0x00000001
- #define XIIC_INTR_TX_ERROR_MASK 0x00000002
- #define XIIC_INTR_TX_EMPTY_MASK 0x00000004
- #define XIIC_INTR_RX_FULL_MASK 0x00000008
- #define XIIC_INTR_BNB_MASK 0x00000010
- #define XIIC_INTR_AAS_MASK 0x00000020
- #define XIIC_INTR_NAAS_MASK 0x00000040
- #define XIIC_INTR_TX_HALF_MASK 0x00000080
- #define XIIC_TX_INTERRUPTS (XIIC_INTR_TX_ERROR_MASK │ XIIC_INTR_TX_EMPTY_MASK │ XIIC_INTR_TX_HALF_MASK)
- #define XIIC_TX_RX_INTERRUPTS (XIIC_INTR_RX_FULL_MASK │ XIIC_TX_INTERRUPTS)

Reset Register mask

- #define XIIC_RESET_MASK 0x0000000A

Control Register masks (CR) mask(s)

- #define XIIC_CR_ENABLE_DEVICE_MASK 0x00000001
- #define XIIC_CR_TX_FIFO_RESET_MASK 0x00000002
- #define XIIC_CR_MSMS_MASK 0x00000004
- #define XIIC_CR_DIR_IS_TX_MASK 0x00000008
- #define XIIC_CR_NO_ACK_MASK 0x00000010
- #define XIIC_CR_REPEATED_START_MASK 0x00000020
- #define XIIC_CR_GENERAL_CALL_MASK 0x00000040

Status Register masks (SR) mask(s)

- #define XIIC_SR_GEN_CALL_MASK 0x00000001
- #define XIIC_SR_ADDR_AS_SLAVE_MASK 0x00000002
- #define XIIC_SR_BUS_BUSY_MASK 0x00000004
- #define XIIC_SR_MSTR_RDING_SLAVE_MASK 0x00000008
- #define XIIC_SR_TX_FIFO_FULL_MASK 0x00000010
- #define XIIC_SR_RX_FIFO_FULL_MASK 0x00000020
- #define XIIC_SR_RX_FIFO_EMPTY_MASK 0x00000040
- #define XIIC_SR_TX_FIFO_EMPTY_MASK 0x00000080

Data Tx Register (DTR) mask(s)

- #define XIIC_TX_DYN_START_MASK 0x00000100
- #define XIIC_TX_DYN_STOP_MASK 0x00000200
- #define IIC_TX_FIFO_DEPTH 16

Data Rx Register (DRR) mask(s)

- #define IIC_RX_FIFO_DEPTH 16
- #define XIIC_TX_ADDR_SENT 0x00
- #define XIIC_TX_ADDR_MSTR_RECV_MASK 0x02
- #define XIIC_READ_OPERATION 1
- #define XIIC_WRITE_OPERATION 0
- #define XIIC_MASTER_ROLE 1
- #define XIIC_SLAVE_ROLE 0
- #define XIIC_STOP 0x00
- #define XIIC_REPEATED_START 0x01
- #define XIic_In32 Xil_In32
- #define XIic_Out32 Xil_Out32
- #define XIic_ReadReg(BaseAddress, RegOffset) XIic_In32((BaseAddress) + (RegOffset))
- #define XIic_WriteReg(BaseAddress, RegOffset, RegisterValue) XIic_Out32((BaseAddress) + (RegOffset), (RegisterValue))
- #define XIic_IntrGlobalDisable(BaseAddress) XIic_WriteReg((BaseAddress), XIIC_DGIER_OFFSET, 0)
- #define XIic_IntrGlobalEnable(BaseAddress) XIic_WriteReg((BaseAddress), XIIC_DGIER_OFFSET, XIIC_GINTR_ENABLE_MASK)
- #define XIic_IsIntrGlobalEnabled(BaseAddress) (XIic_ReadReg((BaseAddress), XIIC_DGIER_OFFSET) == XIIC_GINTR_ENABLE_MASK)
- #define XIic_WriteIisr(BaseAddress, Status) XIic_WriteReg((BaseAddress), XIIC_IISR_OFFSET, (Status))
- #define XIic_ReadIisr(BaseAddress) XIic_ReadReg((BaseAddress), XIIC_IISR_OFFSET)
- #define XIic_WriteIier(BaseAddress, Enable) XIic_WriteReg((BaseAddress), XIIC_IIER_OFFSET, (Enable))
- #define XIic_ReadIier(BaseAddress) XIic_ReadReg((BaseAddress), XIIC_IIER_OFFSET)
- #define XIic_ClearIisr(BaseAddress, InterruptMask) XIic_WriteIisr((BaseAddress), XIic_ReadIisr(Base↩ Address) & (InterruptMask))
- #define XIic_Send7BitAddress(BaseAddress, SlaveAddress, Operation)
- #define XIic_DynSend7BitAddress(BaseAddress, SlaveAddress, Operation)

- #define XIic_DynSendStartStopAddress(BaseAddress, SlaveAddress, Operation)
- #define XIic_DynSendStop(BaseAddress, ByteCount)
- unsigned XIic_Recv (UINTPTR BaseAddress, u8 Address, u8 ∗BufferPtr, unsigned ByteCount, u8 Option)
- unsigned XIic_Send (UINTPTR BaseAddress, u8 Address, u8 ∗BufferPtr, unsigned ByteCount, u8 Option)
- unsigned XIic_DynRecv (UINTPTR BaseAddress, u8 Address, u8 ∗BufferPtr, u8 ByteCount)
- unsigned XIic_DynSend (UINTPTR BaseAddress, u16 Address, u8 ∗BufferPtr, u8 ByteCount, u8 Option)
- int XIic_DynInit (UINTPTR BaseAddress)
- u32 XIic_CheckIsBusBusy (UINTPTR BaseAddress)
- u32 XIic_WaitBusFree (UINTPTR BaseAddress)

6.85.1 Macro Definition Documentation

6.85.1.1 IIC_RX_FIFO_DEPTH

```
#define IIC_RX_FIFO_DEPTH 16
```

Rx fifo capacity

Definition at line 191 of file xiic_l.h.

6.85.1.2 IIC_TX_FIFO_DEPTH

```
#define IIC_TX_FIFO_DEPTH 16
```

Tx fifo capacity

Definition at line 184 of file xiic_l.h.

6.85.1.3 XIIC_ADR_REG_OFFSET

```
#define XIIC_ADR_REG_OFFSET 0x110
```

Address Register

Definition at line 86 of file xiic_l.h.

6.85.1.4 XIic_ClearIisr

```
#define XIic_ClearIisr(
            BaseAddress,
            InterruptMask )  XIic_WriteIisr((BaseAddress), XIic_ReadIisr(BaseAddress) &
(InterruptMask))
```

This macro clears the specified interrupt in the Interrupt status register. It is non-destructive in that the register is read and only the interrupt specified is cleared. Clearing an interrupt acknowledges it.

Parameters

BaseAddress	is the base address of the IIC device.
InterruptMask	is the bit mask of the interrupts to be cleared.

Returns

None.

Note

C-Style signature: void Xlic_ClearIisr(u32 BaseAddress, u32 InterruptMask);

Definition at line 432 of file xiic_l.h.

6.85.1.5 XIIC_CR_DIR_IS_TX_MASK

```
#define XIIC_CR_DIR_IS_TX_MASK 0x00000008
```

Dir of Tx. Txing=1

Definition at line 152 of file xiic_l.h.

6.85.1.6 XIIC_CR_ENABLE_DEVICE_MASK

```
#define XIIC_CR_ENABLE_DEVICE_MASK 0x00000001
```

Device enable = 1

Definition at line 149 of file xiic_l.h.

6.85.1.7 XIIC_CR_GENERAL_CALL_MASK

```
#define XIIC_CR_GENERAL_CALL_MASK 0x00000040
```

Gen Call enabled = 1

Definition at line 155 of file xiic_l.h.

6.85.1.8 XIIC_CR_MSMS_MASK

```
#define XIIC_CR_MSMS_MASK 0x00000004
```

Master starts Txing=1

Definition at line 151 of file xiic_l.h.

6.85.1.9 XIIC_CR_NO_ACK_MASK

```
#define XIIC_CR_NO_ACK_MASK 0x00000010
```

Tx Ack. NO ack = 1

Definition at line 153 of file xiic_l.h.

6.85.1.10 XIIC_CR_REG_OFFSET

```
#define XIIC_CR_REG_OFFSET 0x100
```

Control Register

Definition at line 82 of file xiic_l.h.

6.85.1.11 XIIC_CR_REPEATED_START_MASK

```
#define XIIC_CR_REPEATED_START_MASK 0x00000020
```

Repeated start = 1

Definition at line 154 of file xiic_l.h.

6.85.1.12 XIIC_CR_TX_FIFO_RESET_MASK

```
#define XIIC_CR_TX_FIFO_RESET_MASK 0x00000002
```

Transmit FIFO reset=1

Definition at line 150 of file xiic_l.h.

6.85.1.13 XIIC_DGIER_OFFSET

```
#define XIIC_DGIER_OFFSET 0x1C
```

Global Interrupt Enable Register

Definition at line 78 of file xiic_l.h.

6.85.1.14 XIIC_DRR_REG_OFFSET

```
#define XIIC_DRR_REG_OFFSET 0x10C
```

Data Rx Register

Definition at line 85 of file xiic_l.h.

6.85.1.15 XIIC_DTR_REG_OFFSET

```
#define XIIC_DTR_REG_OFFSET 0x108
```

Data Tx Register

Definition at line 84 of file xiic_l.h.

6.85.1.16 XIic_DynSend7BitAddress

```
#define XIic_DynSend7BitAddress(
            BaseAddress,
            SlaveAddress,
            Operation )
```

Value:
```
{                                                               \
  u8 LocalAddr = (u8)(SlaveAddress « 1);                        \
  LocalAddr = (LocalAddr & 0xFE) | (Operation);                \
  XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,              \
            XIIC_TX_DYN_START_MASK | LocalAddr);               \
}
```

This macro sends the address for a 7 bit address during both read and write operations. It takes care of the details to format the address correctly. This macro is designed to be called internally to the drivers for Dynamic controller functionality.

Parameters

BaseAddress	is the base address of the IIC Device.
SlaveAddress	is the address of the slave to send to.
Operation	indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION.

Returns

> None.

Note

> C-Style signature: void XIic_DynSend7BitAddress(u32 BaseAddress, u8 SlaveAddress, u8 Operation);

Definition at line 479 of file xiic_l.h.

6.85.1.17 XIic_DynSendStartStopAddress

```
#define XIic_DynSendStartStopAddress(
            BaseAddress,
            SlaveAddress,
            Operation )
```

Value:
```
{                                                                      \
  u8 LocalAddr = (u8)(SlaveAddress « 1);                               \
  LocalAddr = (LocalAddr & 0xFE) | (Operation);                        \
  XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                      \
            XIIC_TX_DYN_START_MASK | XIIC_TX_DYN_STOP_MASK | LocalAddr); \
}
```

This macro sends the address, start and stop for a 7 bit address during both write operations. It takes care of the details to format the address correctly. This macro is designed to be called internally to the drivers.

Parameters

BaseAddress	is the base address of the IIC Device.
SlaveAddress	is the address of the slave to send to.
Operation	indicates XIIC_WRITE_OPERATION.

Returns

> None.

Note

> C-Style signature: void XIic_DynSendStartStopAddress(u32 BaseAddress, u8 SlaveAddress, u8 Operation);

Definition at line 506 of file xiic_l.h.

6.85.1.18 XIic_DynSendStop

```
#define XIic_DynSendStop(
            BaseAddress,
            ByteCount )
```

Value:
```
{                                                                      \
  XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                       \
           XIIC_TX_DYN_STOP_MASK | ByteCount);                         \
}
```

This macro sends a stop condition on IIC bus for Dynamic logic.

Parameters

BaseAddress	is the base address of the IIC Device.
ByteCount	is the number of Rx bytes received before the master. doesn't respond with ACK.

Returns

> None.

Note

> C-Style signature: void XIic_DynSendStop(u32 BaseAddress, u32 ByteCount);

Definition at line 529 of file xiic_l.h.

6.85.1.19 XIIC_GINTR_ENABLE_MASK

```
#define XIIC_GINTR_ENABLE_MASK 0x80000000
```

Global Interrupt Enable Mask

Definition at line 98 of file xiic_l.h.

6.85.1.20 XIIC_GPO_REG_OFFSET

```
#define XIIC_GPO_REG_OFFSET 0x124
```

Output Register

Definition at line 91 of file xiic_l.h.

6.85.1.21 XIIC_IIER_OFFSET

```
#define XIIC_IIER_OFFSET 0x28
```

Interrupt Enable Register

Definition at line 80 of file xiic_l.h.

6.85.1.22   XIIC_IISR_OFFSET

```
#define XIIC_IISR_OFFSET 0x20
```

Interrupt Status Register

Definition at line 79 of file xiic_l.h.

6.85.1.23   XIic_In32

```
#define XIic_In32 Xil_In32
```

Definition at line 225 of file xiic_l.h.

6.85.1.24   XIIC_INTR_AAS_MASK

```
#define XIIC_INTR_AAS_MASK 0x00000020
```

1 = When addr as slave

Definition at line 121 of file xiic_l.h.

6.85.1.25   XIIC_INTR_ARB_LOST_MASK

```
#define XIIC_INTR_ARB_LOST_MASK 0x00000001
```

1 = Arbitration lost

Definition at line 116 of file xiic_l.h.

6.85.1.26   XIIC_INTR_BNB_MASK

```
#define XIIC_INTR_BNB_MASK 0x00000010
```

1 = Bus not busy

Definition at line 120 of file xiic_l.h.

6.85.1.27   XIIC_INTR_NAAS_MASK

```
#define XIIC_INTR_NAAS_MASK 0x00000040
```

1 = Not addr as slave

Definition at line 122 of file xiic_l.h.

6.85.1.28 XIIC_INTR_RX_FULL_MASK

```
#define XIIC_INTR_RX_FULL_MASK 0x00000008
```

1 = Rx FIFO/reg=OCY level

Definition at line 119 of file xiic_l.h.

6.85.1.29 XIIC_INTR_TX_EMPTY_MASK

```
#define XIIC_INTR_TX_EMPTY_MASK 0x00000004
```

1 = Tx FIFO/reg empty

Definition at line 118 of file xiic_l.h.

6.85.1.30 XIIC_INTR_TX_ERROR_MASK

```
#define XIIC_INTR_TX_ERROR_MASK 0x00000002
```

1 = Tx error/msg complete

Definition at line 117 of file xiic_l.h.

6.85.1.31 XIIC_INTR_TX_HALF_MASK

```
#define XIIC_INTR_TX_HALF_MASK 0x00000080
```

1 = Tx FIFO half empty

Definition at line 123 of file xiic_l.h.

6.85.1.32 XIic_IntrGlobalDisable

```
#define XIic_IntrGlobalDisable(
            BaseAddress )  XIic_WriteReg((BaseAddress), XIIC_DGIER_OFFSET, 0)
```

This macro disables all interrupts for the device by writing to the Global interrupt enable register.

Parameters

BaseAddress	is the base address of the IIC device.

Returns

None.

Note

C-Style signature: void XIic_IntrGlobalDisable(u32 BaseAddress);

Definition at line 287 of file xiic_l.h.

6.85.1.33 XIic_IntrGlobalEnable

```
#define XIic_IntrGlobalEnable(
            BaseAddress )  XIic_WriteReg((BaseAddress), XIIC_DGIER_OFFSET, XIIC_GINTR_ENABLE_MASK)
```

This macro writes to the global interrupt enable register to enable interrupts from the device. This function does not enable individual interrupts as the Interrupt Enable Register must be set appropriately.

Parameters

BaseAddress	is the base address of the IIC device.

Returns

None.

Note

C-Style signature: void XIic_IntrGlobalEnable(u32 BaseAddress);

Definition at line 305 of file xiic_l.h.

6.85.1.34 XIic_IsIntrGlobalEnabled

```
#define XIic_IsIntrGlobalEnabled(
            BaseAddress )  (XIic_ReadReg((BaseAddress), XIIC_DGIER_OFFSET) == XIIC_GINTR_ENABLE_MASK)
```

This function determines if interrupts are enabled at the global level by reading the global interrupt register.

Parameters

BaseAddress	is the base address of the IIC device.

Returns

- TRUE if the global interrupt is enabled.
- FALSE if global interrupt is disabled.

Note

C-Style signature: int XIic_IsIntrGlobalEnabled(u32 BaseAddress);

Definition at line 324 of file xiic_l.h.

6.85.1.35 XIIC_L_H

```
#define XIIC_L_H /* by using protection macros */
```

This header file contains identifiers and driver functions (or macros) that can be used to access the device in normal and dynamic controller mode. High-level driver functions are defined in xiic.h.

```
MODIFICATION HISTORY:

Ver   Who  Date      Changes
----- ---- --------  ----------------------------------------------
1.00b jhl  05/07/02  First release
1.01c ecm  12/05/02  new rev
1.01d jhl  10/08/03  Added general purpose output feature
1.02a mta  03/09/06  Implemented Repeated Start in the Low Level Driver.
1.03a mta  04/04/06  Implemented Dynamic IIC core routines.
1.03a rpm  09/08/06  Added include of xstatus.h for completeness
1.13a wgr  03/22/07  Converted to new coding style.
1.16a ktn  07/18/09  Updated the notes in XIIC_RESET macro to clearly indicate
                     that only the Interrupt Registers are reset.
1.16a ktn  10/16/09  Updated the notes in the XIIC_RESET macro to mention
                     that the complete IIC core is Reset on giving a software
                     reset to the IIC core. Some previous versions of the
                     core only reset the Interrupt Logic/Registers, please
                     refer to the HW specification for further details.
2.00a sdm  10/22/09  Converted all register accesses to 32 bit access,
           the register offsets are defined to be on 32 bit boundary.
           Removed the macro XIIC_RESET, XIic_Reset API should be
           used in its place.
           Some of the macros have been renamed to be consistent -
           XIIC_GINTR_DISABLE is renamed as XIic_IntrGlobalDisable,
           XIIC_GINTR_ENABLE is renamed as XIic_IntrGlobalEnable,
           XIIC_IS_GINTR_ENABLED is renamed as
           XIic_IsIntrGlobalEnabled,
           XIIC_WRITE_IISR is renamed as XIic_WriteIisr,
           XIIC_READ_IISR is renamed as XIic_ReadIisr,
           XIIC_WRITE_IIER is renamed as XIic_WriteIier
           The _m prefix in the name of the macros has been removed -
           XIic_mClearIisr is now XIic_ClearIisr,
           XIic_mSend7BitAddress is now XIic_Send7BitAddress,
           XIic_mDynSend7BitAddress is now XIic_DynSend7BitAddress,
           XIic_mDynSendStartStopAddress is now
           XIic_DynSendStartStopAddress,
           XIic_mDynSendStop is now XIic_DynSendStop.
3.2   sk   11/10/15  Used UINTPTR instead of u32 for Baseaddress CR# 867425.
                     Changed the prototypes of XIic_Recv, XIic_Send,
                     XIic_DynRecv, XIic_DynSend and XIic_DynInit APIs.
3.3   als  06/27/16  Added Low-level XIic_CheckIsBusBusy API.
3.3   als  06/27/16  Added low-level XIic_WaitBusFree API.
```

Definition at line 61 of file xiic_l.h.

6.85.1.36 XIIC_MASTER_ROLE

```
#define XIIC_MASTER_ROLE 1
```

The following constants are used with the transmit FIFO fill function to specify the role which the IIC device is acting as, a master or a slave. Master on the IIC bus

Definition at line 208 of file xiic_l.h.

6.85.1.37 XIic_Out32

```
#define XIic_Out32 Xil_Out32
```

Definition at line 226 of file xiic_l.h.

6.85.1.38 XIIC_READ_OPERATION

```
#define XIIC_READ_OPERATION 1
```

The following constants are used to specify whether to do Read or a Write operation on IIC bus. Read operation on the IIC bus

Definition at line 201 of file xiic_l.h.

6.85.1.39 XIic_ReadIier

```
#define XIic_ReadIier(
              BaseAddress ) XIic_ReadReg((BaseAddress), XIIC_IIER_OFFSET)
```

This function gets the Interrupt Enable Register contents.

Parameters

BaseAddress	is the base address of the IIC device.

Returns

The contents read from the Interrupt Enable Register. Bit positions of 1 indicate that the corresponding interrupt is enabled. Bit positions of 0 indicate that the corresponding interrupt is disabled.

Note

C-Style signature: u32 XIic_ReadIier(u32 BaseAddress)

Definition at line 414 of file xiic_l.h.

6.85.1.40 XIic_ReadIisr

```
#define XIic_ReadIisr(
              BaseAddress ) XIic_ReadReg((BaseAddress), XIIC_IISR_OFFSET)
```

This function gets the contents of the Interrupt Status Register. This register indicates the status of interrupt sources for the device. The status is independent of whether interrupts are enabled such that the status register may also be polled when interrupts are not enabled.

Parameters

BaseAddress	is the base address of the IIC device.

Returns

The value read from the Interrupt Status Register.

Note

C-Style signature: u32 XIic_ReadIisr(u32 BaseAddress);

Definition at line 371 of file xiic_l.h.

6.85.1.41 XIic_ReadReg

```
#define XIic_ReadReg(
            BaseAddress,
            RegOffset )  XIic_In32((BaseAddress) + (RegOffset))
```

Read from the specified IIC device register.

Parameters

BaseAddress	is the base address of the device.
RegOffset	is the offset from the 1st register of the device to select the specific register.

Returns

The value read from the register.

Note

C-Style signature: u32 XIic_ReadReg(u32 BaseAddress, u32 RegOffset);

```
This macro does not do any checking to ensure that the
```

register exists if the register may be excluded due to parameterization, such as the GPO Register.

Definition at line 247 of file xiic_l.h.

6.85.1.42 XIIC_REPEATED_START

```
#define XIIC_REPEATED_START  0x01
```

Donot Send a stop on the IIC bus after \ the current data transfer

Definition at line 221 of file xiic_l.h.

6.85.1.43 XIIC_RESET_MASK

```
#define XIIC_RESET_MASK 0x0000000A
```

RESET Mask

Definition at line 142 of file xiic_l.h.

6.85.1.44 XIIC_RESETR_OFFSET

`#define XIIC_RESETR_OFFSET 0x40`

Reset Register

Definition at line 81 of file xiic_l.h.

6.85.1.45 XIIC_RFD_REG_OFFSET

`#define XIIC_RFD_REG_OFFSET 0x120`

Rx FIFO Depth reg

Definition at line 90 of file xiic_l.h.

6.85.1.46 XIIC_RFO_REG_OFFSET

`#define XIIC_RFO_REG_OFFSET 0x118`

Rx FIFO Occupancy

Definition at line 88 of file xiic_l.h.

6.85.1.47 XIic_Send7BitAddress

```
#define XIic_Send7BitAddress(
            BaseAddress,
            SlaveAddress,
            Operation )
```

Value:
```
{                                                                    \
    u8 LocalAddr = (u8)(SlaveAddress << 1);                          \
    LocalAddr = (LocalAddr & 0xFE) | (Operation);                    \
    XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, LocalAddr);      \
}
```

This macro sends the address for a 7 bit address during both read and write operations. It takes care of the details to format the address correctly. This macro is designed to be called internally to the drivers.

Parameters

BaseAddress	is the base address of the IIC Device.
SlaveAddress	is the address of the slave to send to.
Operation	indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION

Returns

None.

Note

> C-Style signature: void XIic_Send7BitAddress(u32 BaseAddress, u8 SlaveAddress, u8 Operation);

Definition at line 453 of file xiic_l.h.

6.85.1.48 XIIC_SLAVE_ROLE

```
#define XIIC_SLAVE_ROLE 0
```

Slave on the IIC bus

Definition at line 209 of file xiic_l.h.

6.85.1.49 XIIC_SR_ADDR_AS_SLAVE_MASK

```
#define XIIC_SR_ADDR_AS_SLAVE_MASK  0x00000002
```

1 = When addressed as \ slave

Definition at line 167 of file xiic_l.h.

6.85.1.50 XIIC_SR_BUS_BUSY_MASK

```
#define XIIC_SR_BUS_BUSY_MASK 0x00000004
```

1 = Bus is busy

Definition at line 168 of file xiic_l.h.

6.85.1.51 XIIC_SR_GEN_CALL_MASK

```
#define XIIC_SR_GEN_CALL_MASK  0x00000001
```

1 = A Master issued \ a GC

Definition at line 164 of file xiic_l.h.

6.85.1.52 XIIC_SR_MSTR_RDING_SLAVE_MASK

```
#define XIIC_SR_MSTR_RDING_SLAVE_MASK  0x00000008
```

1 = Dir: Master <− \ slave

Definition at line 171 of file xiic_l.h.

6.85.1.53 XIIC_SR_REG_OFFSET

```
#define XIIC_SR_REG_OFFSET 0x104
```

Status Register

Definition at line 83 of file xiic_l.h.

6.85.1.54 XIIC_SR_RX_FIFO_EMPTY_MASK

```
#define XIIC_SR_RX_FIFO_EMPTY_MASK 0x00000040
```

1 = Rx FIFO empty

Definition at line 174 of file xiic_l.h.

6.85.1.55 XIIC_SR_RX_FIFO_FULL_MASK

```
#define XIIC_SR_RX_FIFO_FULL_MASK 0x00000020
```

1 = Rx FIFO full

Definition at line 173 of file xiic_l.h.

6.85.1.56 XIIC_SR_TX_FIFO_EMPTY_MASK

```
#define XIIC_SR_TX_FIFO_EMPTY_MASK 0x00000080
```

1 = Tx FIFO empty

Definition at line 175 of file xiic_l.h.

6.85.1.57 XIIC_SR_TX_FIFO_FULL_MASK

```
#define XIIC_SR_TX_FIFO_FULL_MASK 0x00000010
```

1 = Tx FIFO full

Definition at line 172 of file xiic_l.h.

6.85.1.58 XIIC_STOP

```
#define XIIC_STOP  0x00
```

The following constants are used with Transmit Function (XIic_Send) to specify whether to STOP after the current transfer of data or own the bus with a Repeated start. Send a stop on the IIC bus after \ the current data transfer

Definition at line 218 of file xiic_l.h.

6.85.1.59 XIIC_TBA_REG_OFFSET

```
#define XIIC_TBA_REG_OFFSET 0x11C
```

10 Bit Address reg

Definition at line 89 of file xiic_l.h.

6.85.1.60 XIIC_TFO_REG_OFFSET

```
#define XIIC_TFO_REG_OFFSET 0x114
```

Tx FIFO Occupancy

Definition at line 87 of file xiic_l.h.

6.85.1.61 XIIC_TX_ADDR_MSTR_RECV_MASK

```
#define XIIC_TX_ADDR_MSTR_RECV_MASK 0x02
```

Definition at line 195 of file xiic_l.h.

6.85.1.62 XIIC_TX_ADDR_SENT

```
#define XIIC_TX_ADDR_SENT 0x00
```

Definition at line 194 of file xiic_l.h.

6.85.1.63 XIIC_TX_DYN_START_MASK

```
#define XIIC_TX_DYN_START_MASK 0x00000100
```

1 = Set dynamic start

Definition at line 182 of file xiic_l.h.

6.85.1.64 XIIC_TX_DYN_STOP_MASK

```
#define XIIC_TX_DYN_STOP_MASK 0x00000200
```

1 = Set dynamic stop

Definition at line 183 of file xiic_l.h.

6.85.1.65 XIIC_TX_INTERRUPTS

```
#define XIIC_TX_INTERRUPTS  (XIIC_INTR_TX_ERROR_MASK | XIIC_INTR_TX_EMPTY_MASK | XIIC_INTR_TX_HALF_MASK)
```

All Tx interrupts commonly used.

Definition at line 128 of file xiic_l.h.

6.85.1.66 XIIC_TX_RX_INTERRUPTS

```
#define XIIC_TX_RX_INTERRUPTS (XIIC_INTR_RX_FULL_MASK | XIIC_TX_INTERRUPTS)
```

All interrupts commonly used

Definition at line 134 of file xiic_l.h.

6.85.1.67 XIIC_WRITE_OPERATION

```
#define XIIC_WRITE_OPERATION 0
```

Write operation on the IIC bus

Definition at line 202 of file xiic_l.h.

6.85.1.68 XIic_WriteIier

```
#define XIic_WriteIier(
             BaseAddress,
             Enable )  XIic_WriteReg((BaseAddress), XIIC_IIER_OFFSET, (Enable))
```

This function sets the contents of the Interrupt Enable Register.

This function writes only the specified value to the register such that some interrupt sources may be enabled and others disabled. It is the caller's responsibility to get the value of the interrupt enable register prior to setting the value to prevent a destructive behavior.

Parameters

BaseAddress	is the base address of the IIC device.
Enable	is the value to be written to the Interrupt Enable Register. Bit positions of 1 will be enabled. Bit positions of 0 will be disabled.

Returns

> None

Note

> C-Style signature: void XIic_WriteIier(u32 BaseAddress, u32 Enable);

Definition at line 394 of file xiic_l.h.

6.85.1.69 XIic_WriteIisr

```
#define XIic_WriteIisr(
            BaseAddress,
            Status )  XIic_WriteReg((BaseAddress), XIIC_IISR_OFFSET, (Status))
```

This function sets the Interrupt status register to the specified value.

This register implements a toggle on write functionality. The interrupt is cleared by writing to this register with the bits to be cleared set to a one and all others to zero. Setting a bit which is zero within this register causes an interrupt to be generated.

This function writes only the specified value to the register such that some status bits may be set and others cleared. It is the caller's responsibility to get the value of the register prior to setting the value to prevent an destructive behavior.

Parameters

BaseAddress	is the base address of the IIC device.
Status	is the value to be written to the Interrupt status register.

Returns

> None.

Note

> C-Style signature: void XIic_WriteIisr(u32 BaseAddress, u32 Status);

Definition at line 352 of file xiic_l.h.

6.85.1.70 XIic_WriteReg

```
#define XIic_WriteReg(
            BaseAddress,
            RegOffset,
            RegisterValue )  XIic_Out32((BaseAddress) + (RegOffset), (RegisterValue))
```

Write to the specified IIC device register.

Parameters

BaseAddress	is the base address of the device.
RegOffset	is the offset from the 1st register of the device to select the specific register.
RegisterValue	is the value to be written to the register.

Returns

> None.

Note

> C-Style signature: void Xlic_WriteReg(u32 BaseAddress, u32 RegOffset, u32 RegisterValue); This macro does not do any checking to ensure that the register exists if the register may be excluded due to parameterization, such as the GPO Register.

Definition at line 270 of file xiic_l.h.

6.85.2 Function Documentation

6.85.2.1 XIic_CheckIsBusBusy()

```
u32 XIic_CheckIsBusBusy (
            UINTPTR BaseAddress )
```

Definition at line 614 of file xiic_l.c.

Here is the caller graph for this function:

6.85.2.2 XIic_DynInit()

```
int XIic_DynInit (
            UINTPTR BaseAddress )
```

6.85.2.3 XIic_DynRecv()

```
unsigned XIic_DynRecv (
            UINTPTR BaseAddress,
            u8 Address,
            u8 * BufferPtr,
            u8 ByteCount )
```

6.85.2.4 XIic_DynSend()

```
unsigned XIic_DynSend (
            UINTPTR BaseAddress,
            u16 Address,
            u8 * BufferPtr,
            u8 ByteCount,
            u8 Option )
```

6.85.2.5 XIic_Recv()

```
unsigned XIic_Recv (
            UINTPTR BaseAddress,
            u8 Address,
            u8 * BufferPtr,
            unsigned ByteCount,
            u8 Option )
```

Receive data as a master on the IIC bus. This function receives the data using polled I/O and blocks until the data has been received. It only supports 7 bit addressing mode of operation. This function returns zero if bus is busy.

Parameters

BaseAddress	contains the base address of the IIC device.
Address	contains the 7 bit IIC address of the device to send the specified data to.
BufferPtr	points to the data to be sent.
ByteCount	is the number of bytes to be sent.
Option	indicates whether to hold or free the bus after reception of data, XIIC_STOP = end with STOP condition, XIIC_REPEATED_START = don't end with STOP condition.

Returns

The number of bytes received.

Note

None.

Definition at line 117 of file xiic_l.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.85.2.6 XIic_Send()

```
unsigned XIic_Send (
            UINTPTR BaseAddress,
            u8 Address,
            u8 * BufferPtr,
            unsigned ByteCount,
            u8 Option )
```

Send data as a master on the IIC bus. This function sends the data using polled I/O and blocks until the data has been sent. It only supports 7 bit addressing mode of operation. This function returns zero if bus is busy.

Parameters

BaseAddress	contains the base address of the IIC device.
Address	contains the 7 bit IIC address of the device to send the specified data to.
BufferPtr	points to the data to be sent.
ByteCount	is the number of bytes to be sent.
Option	indicates whether to hold or free the bus after transmitting the data.

Returns

The number of bytes sent.

Note

None.

Definition at line 373 of file xiic_l.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.85.2.7 XIic_WaitBusFree()

```
u32 XIic_WaitBusFree (
            UINTPTR BaseAddress )
```

This function will wait until the I2C bus is free or timeout.

Parameters

BaseAddress	contains the base address of the I2C device.

Returns

- XST_SUCCESS if the I2C bus was freed before the timeout.
- XST_FAILURE otherwise.

Note

None.

Definition at line 638 of file xiic_l.c.

Here is the call graph for this function: Here is the caller graph for this function:

6.86   xiic_l.h

Go to the documentation of this file.
```
00001 /******************************************************************************
00002  * Copyright (C) 2002 - 2021 Xilinx, Inc.  All rights reserved.
00003  * SPDX-License-Identifier: MIT
00004  ******************************************************************************/
00005
00006 /******************************************************************************/
00060 #ifndef XIIC_L_H /* prevent circular inclusions */
00061 #define XIIC_L_H /* by using protection macros */
00062
00063 #ifdef __cplusplus
00064 extern "C" {
00065 #endif
00066
00067 /*********************** Include Files ****************************/
00068
00069 #include "xil_io.h"
00070 #include "xil_types.h"
00071
00072 /********************** Constant Definitions **************************/
00073
00078 #define XIIC_DGIER_OFFSET 0x1C
00079 #define XIIC_IISR_OFFSET 0x20
00080 #define XIIC_IIER_OFFSET 0x28
00081 #define XIIC_RESETR_OFFSET 0x40
00082 #define XIIC_CR_REG_OFFSET 0x100
00083 #define XIIC_SR_REG_OFFSET 0x104
00084 #define XIIC_DTR_REG_OFFSET 0x108
00085 #define XIIC_DRR_REG_OFFSET 0x10C
00086 #define XIIC_ADR_REG_OFFSET 0x110
00087 #define XIIC_TFO_REG_OFFSET 0x114
00088 #define XIIC_RFO_REG_OFFSET 0x118
00089 #define XIIC_TBA_REG_OFFSET 0x11C
00090 #define XIIC_RFD_REG_OFFSET 0x120
00091 #define XIIC_GPO_REG_OFFSET 0x124
00092 /* @} */
00093
00098 #define XIIC_GINTR_ENABLE_MASK 0x80000000
00099 /* @} */
```

```
00100
00116 #define XIIC_INTR_ARB_LOST_MASK 0x00000001
00117 #define XIIC_INTR_TX_ERROR_MASK 0x00000002
00118 #define XIIC_INTR_TX_EMPTY_MASK 0x00000004
00119 #define XIIC_INTR_RX_FULL_MASK 0x00000008
00120 #define XIIC_INTR_BNB_MASK 0x00000010
00121 #define XIIC_INTR_AAS_MASK 0x00000020
00122 #define XIIC_INTR_NAAS_MASK 0x00000040
00123 #define XIIC_INTR_TX_HALF_MASK 0x00000080
00128 #define XIIC_TX_INTERRUPTS                                                \
00129   (XIIC_INTR_TX_ERROR_MASK | XIIC_INTR_TX_EMPTY_MASK | XIIC_INTR_TX_HALF_MASK)
00130
00134 #define XIIC_TX_RX_INTERRUPTS (XIIC_INTR_RX_FULL_MASK | XIIC_TX_INTERRUPTS)
00135
00136 /* @} */
00137
00142 #define XIIC_RESET_MASK 0x0000000A
00143 /* @} */
00144
00149 #define XIIC_CR_ENABLE_DEVICE_MASK 0x00000001
00150 #define XIIC_CR_TX_FIFO_RESET_MASK 0x00000002
00151 #define XIIC_CR_MSMS_MASK 0x00000004
00152 #define XIIC_CR_DIR_IS_TX_MASK 0x00000008
00153 #define XIIC_CR_NO_ACK_MASK 0x00000010
00154 #define XIIC_CR_REPEATED_START_MASK 0x00000020
00155 #define XIIC_CR_GENERAL_CALL_MASK 0x00000040
00156 /* @} */
00157
00162 #define XIIC_SR_GEN_CALL_MASK                                             \
00163   0x00000001
00165 #define XIIC_SR_ADDR_AS_SLAVE_MASK                                        \
00166   0x00000002
00168 #define XIIC_SR_BUS_BUSY_MASK 0x00000004
00169 #define XIIC_SR_MSTR_RDING_SLAVE_MASK                                     \
00170   0x00000008
00172 #define XIIC_SR_TX_FIFO_FULL_MASK 0x00000010
00173 #define XIIC_SR_RX_FIFO_FULL_MASK 0x00000020
00174 #define XIIC_SR_RX_FIFO_EMPTY_MASK 0x00000040
00175 #define XIIC_SR_TX_FIFO_EMPTY_MASK 0x00000080
00176 /* @} */
00177
00182 #define XIIC_TX_DYN_START_MASK 0x00000100
00183 #define XIIC_TX_DYN_STOP_MASK 0x00000200
00184 #define IIC_TX_FIFO_DEPTH 16
00185 /* @} */
00186
00191 #define IIC_RX_FIFO_DEPTH 16
00192 /* @} */
00193
00194 #define XIIC_TX_ADDR_SENT 0x00
00195 #define XIIC_TX_ADDR_MSTR_RECV_MASK 0x02
00196
00201 #define XIIC_READ_OPERATION 1
00202 #define XIIC_WRITE_OPERATION 0
00208 #define XIIC_MASTER_ROLE 1
00209 #define XIIC_SLAVE_ROLE 0
00216 #define XIIC_STOP                                                         \
00217   0x00
00219 #define XIIC_REPEATED_START                                               \
00220   0x01
00223 /***************** Macros (Inline Functions) Definitions *********************/
00224
00225 #define XIic_In32 Xil_In32
00226 #define XIic_Out32 Xil_Out32
00227
00228 /*****************************************************************************/
00247 #define XIic_ReadReg(BaseAddress, RegOffset)                              \
00248   XIic_In32((BaseAddress) + (RegOffset))
00249
00250 /*****************************************************************************/
00270 #define XIic_WriteReg(BaseAddress, RegOffset, RegisterValue)              \
00271   XIic_Out32((BaseAddress) + (RegOffset), (RegisterValue))
00272
00273 /*****************************************************************************/
00287 #define XIic_IntrGlobalDisable(BaseAddress)                               \
00288   XIic_WriteReg((BaseAddress), XIIC_DGIER_OFFSET, 0)
00289
00290 /*****************************************************************************/
00305 #define XIic_IntrGlobalEnable(BaseAddress)                                \
00306   XIic_WriteReg((BaseAddress), XIIC_DGIER_OFFSET, XIIC_GINTR_ENABLE_MASK)
00307
00308 /*****************************************************************************/
00324 #define XIic_IsIntrGlobalEnabled(BaseAddress)                             \
00325   (XIic_ReadReg((BaseAddress), XIIC_DGIER_OFFSET) == XIIC_GINTR_ENABLE_MASK)
00326
00327 /*****************************************************************************/
00352 #define XIic_WriteIisr(BaseAddress, Status)                               \
```

```
00353   XIic_WriteReg((BaseAddress), XIIC_IISR_OFFSET, (Status))
00354
00355 /********************************************************************/
00371 #define XIic_ReadIisr(BaseAddress) XIic_ReadReg((BaseAddress), XIIC_IISR_OFFSET)
00372
00373 /********************************************************************/
00394 #define XIic_WriteIier(BaseAddress, Enable)                              \
00395   XIic_WriteReg((BaseAddress), XIIC_IIER_OFFSET, (Enable))
00396
00397 /********************************************************************/
00414 #define XIic_ReadIier(BaseAddress) XIic_ReadReg((BaseAddress), XIIC_IIER_OFFSET)
00415
00416 /********************************************************************/
00432 #define XIic_ClearIisr(BaseAddress, InterruptMask)                       \
00433   XIic_WriteIisr((BaseAddress), XIic_ReadIisr(BaseAddress) & (InterruptMask))
00434
00435 /********************************************************************/
00453 #define XIic_Send7BitAddress(BaseAddress, SlaveAddress, Operation)       \
00454   {                                                                      \
00455     u8 LocalAddr = (u8)(SlaveAddress « 1);                               \
00456     LocalAddr = (LocalAddr & 0xFE) | (Operation);                        \
00457     XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, LocalAddr);           \
00458   }
00459
00460 /********************************************************************/
00479 #define XIic_DynSend7BitAddress(BaseAddress, SlaveAddress, Operation)    \
00480   {                                                                      \
00481     u8 LocalAddr = (u8)(SlaveAddress « 1);                               \
00482     LocalAddr = (LocalAddr & 0xFE) | (Operation);                        \
00483     XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                       \
00484                 XIIC_TX_DYN_START_MASK | LocalAddr);                     \
00485   }
00486
00487 /********************************************************************/
00506 #define XIic_DynSendStartStopAddress(BaseAddress, SlaveAddress, Operation) \
00507   {                                                                      \
00508     u8 LocalAddr = (u8)(SlaveAddress « 1);                               \
00509     LocalAddr = (LocalAddr & 0xFE) | (Operation);                        \
00510     XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                       \
00511                 XIIC_TX_DYN_START_MASK | XIIC_TX_DYN_STOP_MASK | LocalAddr); \
00512   }
00513
00514 /********************************************************************/
00529 #define XIic_DynSendStop(BaseAddress, ByteCount)                         \
00530   {                                                                      \
00531     XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                       \
00532                 XIIC_TX_DYN_STOP_MASK | ByteCount);                      \
00533   }
00534
00535 /************************* Function Prototypes ***************************/
00536
00537 unsigned XIic_Recv(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
00538                 unsigned ByteCount, u8 Option);
00539
00540 unsigned XIic_Send(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
00541                 unsigned ByteCount, u8 Option);
00542
00543 unsigned XIic_DynRecv(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
00544                     u8 ByteCount);
00545
00546 unsigned XIic_DynSend(UINTPTR BaseAddress, u16 Address, u8 *BufferPtr,
00547                     u8 ByteCount, u8 Option);
00548
00549 int XIic_DynInit(UINTPTR BaseAddress);
00550
00551 u32 XIic_CheckIsBusBusy(UINTPTR BaseAddress);
00552
00553 u32 XIic_WaitBusFree(UINTPTR BaseAddress);
00554
00555 #ifdef __cplusplus
00556 }
00557 #endif
00558
00559 #endif /* end of protection macro */
```

6.87   library/xil_io.h File Reference

```
#include "xil_types.h"
```
Include dependency graph for xil_io.h: This graph shows which files directly or indirectly include this file:

Macros

- #define XIL_IO_H /∗ by using protection macros ∗/
- #define SYNCHRONIZE_IO
- #define INST_SYNC
- #define DATA_SYNC
- #define INST_SYNC
- #define DATA_SYNC
- #define INLINE __inline
- #define Xil_In16LE Xil_In16
- #define Xil_In32LE Xil_In32
- #define Xil_Out16LE Xil_Out16
- #define Xil_Out32LE Xil_Out32
- #define Xil_Htons Xil_EndianSwap16
- #define Xil_Htonl Xil_EndianSwap32
- #define Xil_Ntohs Xil_EndianSwap16
- #define Xil_Ntohl Xil_EndianSwap32

6.87.1 Macro Definition Documentation

6.87.1.1 DATA_SYNC [1/2]

```
#define DATA_SYNC
```

Definition at line 64 of file xil_io.h.

6.87.1.2 DATA_SYNC [2/2]

```
#define DATA_SYNC
```

Definition at line 64 of file xil_io.h.

6.87.1.3 INLINE

```
#define INLINE __inline
```

Definition at line 72 of file xil_io.h.

6.87.1.4 INST_SYNC [1/2]

```
#define INST_SYNC
```

Definition at line 63 of file xil_io.h.

6.87.1.5 INST_SYNC [2/2]

```
#define INST_SYNC
```

Definition at line 63 of file xil_io.h.

6.87.1.6 SYNCHRONIZE_IO

`#define SYNCHRONIZE_IO`

Definition at line 62 of file xil_io.h.

6.87.1.7 Xil_Htonl

`#define Xil_Htonl Xil_EndianSwap32`

Definition at line 315 of file xil_io.h.

6.87.1.8 Xil_Htons

`#define Xil_Htons Xil_EndianSwap16`

Definition at line 314 of file xil_io.h.

6.87.1.9 Xil_In16LE

`#define Xil_In16LE Xil_In16`

Definition at line 310 of file xil_io.h.

6.87.1.10 Xil_In32LE

`#define Xil_In32LE Xil_In32`

Definition at line 311 of file xil_io.h.

6.87.1.11 XIL_IO_H

`#define XIL_IO_H /* by using protection macros */`

The xil_io.h file contains the interface for the general I/O component, which encapsulates the Input/Output functions for the processors that do not require any special I/O handling.

```
  MODIFICATION HISTORY:

  Ver    Who       Date      Changes
  -----  --------  --------  ----------------------------------------------
  5.00   pkp       05/29/14  First release
  6.00   mus       08/19/16  Remove checking of __LITTLE_ENDIAN__ flag for
                             ARM processors
  7.20   har       01/03/20  Added Xil_SecureOut32 for avoiding blindwrite for
                             CR-1049218
  7.30   kpt       09/21/20  Moved Xil_EndianSwap16 and Xil_EndianSwap32 to
                             xil_io.h and made them as static inline
         am        10/13/20  Changed the return type of Xil_SecureOut32 function
                             from u32 to int
  7.50   dp        02/12/21  Fix compilation error in Xil_EndianSwap32() that
*occur when -Werror=conversion compiler flag is enabled 7.5   mus     05/17/21
*Update the functions with comments. It fixes CR#1067739.
```

Definition at line 36 of file xil_io.h.

6.87.1.12 Xil_Ntohl

```
#define Xil_Ntohl Xil_EndianSwap32
```

Definition at line 317 of file xil_io.h.

6.87.1.13 Xil_Ntohs

```
#define Xil_Ntohs Xil_EndianSwap16
```

Definition at line 316 of file xil_io.h.

6.87.1.14 Xil_Out16LE

```
#define Xil_Out16LE Xil_Out16
```

Definition at line 312 of file xil_io.h.

6.87.1.15 Xil_Out32LE

```
#define Xil_Out32LE Xil_Out32
```

Definition at line 313 of file xil_io.h.

6.88 xil_io.h

Go to the documentation of this file.
```
00001 /******************************************************************************
00002  * Copyright (c) 2014 - 2021 Xilinx, Inc.  All rights reserved.
00003  * SPDX-License-Identifier: MIT
00004  ******************************************************************************/
00005
00006 /******************************************************************************/
00035 #ifndef XIL_IO_H /* prevent circular inclusions */
00036 #define XIL_IO_H /* by using protection macros */
00037
00038 #ifdef __cplusplus
00039 extern "C" {
00040 #endif
00041
00042 /*********************** Include Files *********************************/
00043
00044 #include "xil_types.h"
00045
00046 /*********************** Function Prototypes *****************************/
00047 #ifdef ENABLE_SAFETY
00048 extern u32 XStl_RegUpdate(u32 RegAddr, u32 RegVal);
00049 #endif
00050
00051 /***************** Macros (Inline Functions) Definitions *****************/
00052 #if defined __GNUC__
00053 #if defined(__MICROBLAZE__)
00054 #define INST_SYNC mbar(0)
00055 #define DATA_SYNC mbar(1)
00056 #else
00057 #define SYNCHRONIZE_IO dmb()
00058 #define INST_SYNC isb()
00059 #define DATA_SYNC dsb()
00060 #endif
00061 #else
00062 #define SYNCHRONIZE_IO
00063 #define INST_SYNC
```

```
00064 #define DATA_SYNC
00065 #define INST_SYNC
00066 #define DATA_SYNC
00067 #endif
00068
00069 #if defined(__GNUC__) || defined(__ICCARM__) || defined(__MICROBLAZE__)
00070 #define INLINE inline
00071 #else
00072 #define INLINE __inline
00073 #endif
00074
00075 /******************************************************************************/
00088 static INLINE u8 Xil_In8(UINTPTR Addr) { return *(volatile u8 *)Addr; }
00089
00090 /******************************************************************************/
00102 static INLINE u16 Xil_In16(UINTPTR Addr) { return *(volatile u16 *)Addr; }
00103
00104 /******************************************************************************/
00116 static INLINE u32 Xil_In32(UINTPTR Addr) { return *(volatile u32 *)Addr; }
00117
00118 /******************************************************************************/
00130 static INLINE u64 Xil_In64(UINTPTR Addr) { return *(volatile u64 *)Addr; }
00131
00132 /******************************************************************************/
00145 static INLINE void Xil_Out8(UINTPTR Addr, u8 Value) {
00146   /* write 8 bit value to specified address */
00147   volatile u8 *LocalAddr = (volatile u8 *)Addr;
00148   *LocalAddr = Value;
00149 }
00150
00151 /******************************************************************************/
00163 static INLINE void Xil_Out16(UINTPTR Addr, u16 Value) {
00164   /* write 16 bit value to specified address */
00165   volatile u16 *LocalAddr = (volatile u16 *)Addr;
00166   *LocalAddr = Value;
00167 }
00168
00169 /******************************************************************************/
00182 static INLINE void Xil_Out32(UINTPTR Addr, u32 Value) {
00183   /* write 32 bit value to specified address */
00184 #ifndef ENABLE_SAFETY
00185   volatile u32 *LocalAddr = (volatile u32 *)Addr;
00186   *LocalAddr = Value;
00187 #else
00188   XStl_RegUpdate(Addr, Value);
00189 #endif
00190 }
00191
00192 /******************************************************************************/
00205 static INLINE void Xil_Out64(UINTPTR Addr, u64 Value) {
00206   /* write 64 bit value to specified address */
00207   volatile u64 *LocalAddr = (volatile u64 *)Addr;
00208   *LocalAddr = Value;
00209 }
00210
00211 /******************************************************************************/
00227 static INLINE int Xil_SecureOut32(UINTPTR Addr, u32 Value) {
00228   int Status = XST_FAILURE;
00229   u32 ReadReg;
00230   u32 ReadRegTemp;
00231
00232   /* writing 32 bit value to specified address */
00233   Xil_Out32(Addr, Value);
00234
00235   /* verify value written to specified address with multiple reads */
00236   ReadReg = Xil_In32(Addr);
00237   ReadRegTemp = Xil_In32(Addr);
00238
00239   if ((ReadReg == Value) && (ReadRegTemp == Value)) {
00240     Status = XST_SUCCESS;
00241   }
00242
00243   return Status;
00244 }
00245
00246 /******************************************************************************/
00256 static INLINE __attribute__((always_inline)) u16 Xil_EndianSwap16(u16 Data) {
00257   return (u16)(((Data & 0xFF00U) >> 8U) | ((Data & 0x00FFU) << 8U));
00258 }
00259
00260 /******************************************************************************/
00270 static INLINE __attribute__((always_inline)) u32 Xil_EndianSwap32(u32 Data) {
00271   u16 LoWord;
00272   u16 HiWord;
00273
00274   /* get each of the half words from the 32 bit word */
00275
```

```
00276    LoWord = (u16)(Data & 0x0000FFFFU);
00277    HiWord = (u16)((Data & 0xFFFF0000U) >> 16U);
00278
00279    /* byte swap each of the 16 bit half words */
00280
00281    LoWord = (u16)(((LoWord & 0xFF00U) >> 8U) | ((LoWord & 0x00FFU) << 8U));
00282    HiWord = (u16)(((HiWord & 0xFF00U) >> 8U) | ((HiWord & 0x00FFU) << 8U));
00283
00284    /* swap the half words before returning the value */
00285
00286    return ((((u32)LoWord) << (u32)16U) | (u32)HiWord);
00287 }
00288
00289 #if defined(__MICROBLAZE__)
00290 #ifdef __LITTLE_ENDIAN__
00291 #define Xil_In16LE Xil_In16
00292 #define Xil_In32LE Xil_In32
00293 #define Xil_Out16LE Xil_Out16
00294 #define Xil_Out32LE Xil_Out32
00295 #define Xil_Htons Xil_EndianSwap16
00296 #define Xil_Htonl Xil_EndianSwap32
00297 #define Xil_Ntohs Xil_EndianSwap16
00298 #define Xil_Ntohl Xil_EndianSwap32
00299 #else
00300 #define Xil_In16BE Xil_In16
00301 #define Xil_In32BE Xil_In32
00302 #define Xil_Out16BE Xil_Out16
00303 #define Xil_Out32BE Xil_Out32
00304 #define Xil_Htons(Data) (Data)
00305 #define Xil_Htonl(Data) (Data)
00306 #define Xil_Ntohs(Data) (Data)
00307 #define Xil_Ntohl(Data) (Data)
00308 #endif
00309 #else
00310 #define Xil_In16LE Xil_In16
00311 #define Xil_In32LE Xil_In32
00312 #define Xil_Out16LE Xil_Out16
00313 #define Xil_Out32LE Xil_Out32
00314 #define Xil_Htons Xil_EndianSwap16
00315 #define Xil_Htonl Xil_EndianSwap32
00316 #define Xil_Ntohs Xil_EndianSwap16
00317 #define Xil_Ntohl Xil_EndianSwap32
00318 #endif
00319
00320 #if defined(__MICROBLAZE__)
00321 #ifdef __LITTLE_ENDIAN__
00322 static INLINE u16 Xil_In16BE(UINTPTR Addr)
00323 #else
00324 static INLINE u16 Xil_In16LE(UINTPTR Addr)
00325 #endif
00326 #else
00327 static INLINE u16 Xil_In16BE(UINTPTR Addr)
00328 #endif
00329 {
00330    u16 value = Xil_In16(Addr);
00331    return Xil_EndianSwap16(value);
00332 }
00333
00334 #if defined(__MICROBLAZE__)
00335 #ifdef __LITTLE_ENDIAN__
00336 static INLINE u32 Xil_In32BE(UINTPTR Addr)
00337 #else
00338 static INLINE u32 Xil_In32LE(UINTPTR Addr)
00339 #endif
00340 #else
00341 static INLINE u32 Xil_In32BE(UINTPTR Addr)
00342 #endif
00343 {
00344    u32 value = Xil_In32(Addr);
00345    return Xil_EndianSwap32(value);
00346 }
00347
00348 #if defined(__MICROBLAZE__)
00349 #ifdef __LITTLE_ENDIAN__
00350 static INLINE void Xil_Out16BE(UINTPTR Addr, u16 Value)
00351 #else
00352 static INLINE void Xil_Out16LE(UINTPTR Addr, u16 Value)
00353 #endif
00354 #else
00355 static INLINE void Xil_Out16BE(UINTPTR Addr, u16 Value)
00356 #endif
00357 {
00358    Value = Xil_EndianSwap16(Value);
00359    Xil_Out16(Addr, Value);
00360 }
00361
00362 #if defined(__MICROBLAZE__)
```

```
00363 #ifdef __LITTLE_ENDIAN__
00364 static INLINE void Xil_Out32BE(UINTPTR Addr, u32 Value)
00365 #else
00366 static INLINE void Xil_Out32LE(UINTPTR Addr, u32 Value)
00367 #endif
00368 #else
00369 static INLINE void Xil_Out32BE(UINTPTR Addr, u32 Value)
00370 #endif
00371 {
00372   Value = Xil_EndianSwap32(Value);
00373   Xil_Out32(Addr, Value);
00374 }
00375
00376 #ifdef __cplusplus
00377 }
00378 #endif
00379
00380 #endif /* end of protection macro */
```

6.89   library/xil_types.h File Reference

This graph shows which files directly or indirectly include this file:

6.90   xil_types.h

Go to the documentation of this file.
```
00001 /******************************************************************************
00002  * Copyright (c) 2010 - 2021 Xilinx, Inc.  All rights reserved.
00003  * Copyright (c) 2022 Advanced Micro Devices, Inc. All Rights Reserved.
00004  * SPDX-License-Identifier: MIT
00005  ******************************************************************************/
00006
00007 /******************************************************************************
00034 #ifndef XIL_TYPES_H /* prevent circular inclusions */
00035 #define XIL_TYPES_H /* by using protection macros */
00036
00037 #ifdef __cplusplus
00038 extern "C" {
00039 #endif
00040
00041 #include <stddef.h>
00042 #include <stdint.h>
00043
00044 /************************** Constant Definitions *****************************/
00045
00046 #define XST_SUCCESS 0L
00047 #define XST_FAILURE 1L
00048 #ifndef TRUE
00049 #define TRUE 1U
00050 #endif
00051
00052 #ifndef FALSE
00053 #define FALSE 0U
00054 #endif
00055
00056 #ifndef NULL
00057 #define NULL 0U
00058 #endif
00059
00060 #define XIL_COMPONENT_IS_READY                                        \
00061   0x11111111U
00066 #define XIL_COMPONENT_IS_STARTED                                      \
00067   0x22222222U
00072 /* @name New types
00073  * New simple types.
00074  * @{
00075  */
00076 #ifndef __KERNEL__
00077 #ifndef XBASIC_TYPES_H
00078 /*
00079  * guarded against xbasic_types.h.
00080  */
00081 typedef uint8_t u8;
00082 typedef uint16_t u16;
00083 typedef uint32_t u32;
```

```
00085 #define __XUINT64__
00086 typedef struct {
00087    u32 Upper;
00088    u32 Lower;
00089 } Xuint64;
00090
00091 /*****************************************************************************/
00100 #define XUINT64_MSW(x) ((x).Upper)
00101
00102 /*****************************************************************************/
00111 #define XUINT64_LSW(x) ((x).Lower)
00112
00113 #endif /* XBASIC_TYPES_H */
00114
00115 /*
00116  * xbasic_types.h does not typedef s* or u64
00117  */
00119 typedef char char8;
00120 typedef int8_t s8;
00121 typedef int16_t s16;
00122 typedef int32_t s32;
00123 typedef int64_t s64;
00124 typedef uint64_t u64;
00125 typedef int sint32;
00126
00127 #if defined(__MICROBLAZE__) && !defined(__arch64__) &&                      \
00128     (XPAR_MICROBLAZE_ADDR_SIZE > 32)
00129 typedef uint64_t UINTPTR;
00130 typedef int64_t INTPTR;
00131 #else
00132 typedef uintptr_t UINTPTR;
00133 typedef intptr_t INTPTR;
00134 #endif
00135
00136 typedef ptrdiff_t PTRDIFF;
00138 #if !defined(LONG) || !defined(ULONG)
00139 typedef long LONG;
00140 typedef unsigned long ULONG;
00141 #endif
00142
00143 #define ULONG64_HI_MASK 0xFFFFFFFF00000000U
00144 #define ULONG64_LO_MASK ~ULONG64_HI_MASK
00145
00146 #else
00147 #include <linux/types.h>
00148 #endif
00149
00155 typedef void (*XInterruptHandler)(void *InstancePtr);
00156
00161 typedef void (*XExceptionHandler)(void *InstancePtr);
00162
00172 #if defined(__aarch64__) || defined(__arch64__)
00173 #define UPPER_32_BITS(n) ((u32)(((n) >> 16) >> 16))
00174 #else
00175 #define UPPER_32_BITS(n) 0U
00176 #endif
00182 #define LOWER_32_BITS(n) ((u32)(n))
00183
00189 #if defined(__aarch64__) || defined(__arch64__)
00190 #define LEFT_SHIFT_BY_32_BITS(n) (u64)(((u64)n) << 32)
00191 #else
00192 #define LEFT_SHIFT_BY_32_BITS(n) 0U
00193 #endif
00194
00195 /************************* Constant Definitions ****************************/
00196
00197 #ifndef TRUE
00198 #define TRUE 1U
00199 #endif
00200
00201 #ifndef FALSE
00202 #define FALSE 0U
00203 #endif
00204
00205 #ifndef NULL
00206 #define NULL 0U
00207 #endif
00208
00209 #ifdef __cplusplus
00210 }
00211 #endif
00212
00213 #endif /* end of protection macro */
```

Index