libpynq

(release 5EWC0-2023 version 0.2.5 of 2023-10-08 10:51)

Generated on Sun Oct 8 2023 10:51:02 for libpynq by Doxygen 1.9.7

Sun Oct 8 2023 10:51:02

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

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

Module 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  )
```

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  )
```

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 )
```

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 )
```

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  )
```

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 )
```

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 )
```

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_
  = 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, 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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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  )
```

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 )
```

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.

```c
#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

```c
#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  )
```

Unitialize the buttons.

Definition at line 50 of file buttons.c.

4.4.4.2 buttons_init()

```
void buttons_init (
             void  )
```

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 )
```

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 )
```

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 )
```

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 )
```

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   )
```

Unitialize the buttons.

Definition at line 65 of file buttons.c.

4.4.4.8 switches_init()

```
void switches_init (
             void   )
```

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   )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

Stop using the display.

Parameters

display	Handle to display.

4.5.4.2 display_init()

```
void display_init (
            display_t * display )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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 )
```

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 )
```

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 )
```

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.

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 42 of file iic.h.

4.8.3 Function Documentation

4.8.3.1 iic_destroy()

```
void iic_destroy (
            const iic_index_t iic )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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  )
```

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 )
```

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 )
```

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  )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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  )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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  )
```

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  )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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 )
```

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  )
```

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  )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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 )
```

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  )
```

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 )
```

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  )
```

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  )
```

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

Go to the documentation of this file.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <adc.h>
#include <arm_shared_memory_system.h>
#include <errno.h>
#include <log.h>
#include <platform.h>
#include <stdio.h>
#include <stdlib.h>

static struct arm_shared_t adc_handle;
static volatile uint32_t *adc = NULL;

static const uint32_t twopow16 = 0b10000000000000000;

```

```
bool invalid_channel_adc(const adc_channel_t channel) {
  if (channel == ADC0) {
    return false;
  }
  if (channel == ADC1) {
    return false;
  }
  if (channel == ADC2) {
    return false;
  }
  if (channel == ADC3) {
    return false;
  }
  if (channel == ADC4) {
    return false;
  }
  if (channel == ADC5) {
    return false;
  }
  return true;
}

bool initialized_adc(void) {
  if (adc == NULL) {
    return false;
  }
  return true;
}

bool check_initialized_adc(void) {
  if (!initialized_adc()) {
    pynq_error("The ADC has not been initialized\n");
  }
  return true;
}

bool check_channel_adc(const adc_channel_t channel) {
  if (invalid_channel_adc(channel)) {
    pynq_error("Invalid ADC channel %d\n", channel);
  }
  return true;
}

void adc_init(void) { adc = arm_shared_init(&adc_handle, xadc_wiz_0, 4096); }

void adc_destroy(void) {
  if (adc != NULL) {
    (void)arm_shared_close(&adc_handle);
    adc = NULL;
  }
}

double adc_read_channel(const adc_channel_t channel) {
  (void)check_channel_adc(channel);
  (void)check_initialized_adc();

  // TODO we need to calibrate this
  double value = adc[channel] * (3.23 / twopow16);

  return value;
}

uint32_t adc_read_channel_raw(adc_channel_t channel) {
  (void)check_channel_adc(channel);
  (void)check_initialized_adc();

  if (adc == NULL) {
    return UINT32_MAX;
  }
  uint32_t value = adc[channel];

  return value;
}
```

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.

```c
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef ADC_H
#define ADC_H

#include <stdbool.h>
#include <stdint.h>

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

extern bool initialized_adc(void);

extern void adc_init(void);

extern void adc_destroy(void);

extern double adc_read_channel(adc_channel_t channel);

extern uint32_t adc_read_channel_raw(adc_channel_t channel);

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#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>

void *arm_shared_init(arm_shared *handle, const uint32_t address,
                      const uint32_t length) {
  if (handle == NULL) {
    fprintf(stderr, "You need to pass a valid handle to %s\n", __FUNCTION__);
    exit(EXIT_FAILURE);
  }

  handle->address = address;
  handle->length = length;
  handle->file_descriptor = open("/dev/mem", O_RDWR | O_SYNC);
  if (handle->file_descriptor < 0) {
    fprintf(stderr,
            "FAILED open memory: %s, please run with sufficient permissions "
            "(sudo).\n",
            strerror(errno));
    exit(EXIT_FAILURE);
  }

  long page_size = sysconf(_SC_PAGE_SIZE);

  uint32_t start_address = handle->address;
  uint32_t page_offset = start_address % page_size;
  start_address -= page_offset;
  handle->length += page_offset;

  handle->mmaped_region =
      mmap(NULL, handle->length, PROT_READ | PROT_WRITE, MAP_SHARED,
           handle->file_descriptor, start_address);

  if (handle->mmaped_region == MAP_FAILED) {
    fprintf(stderr, "FAILED to memory map requested region: %s\n",
            strerror(errno));
    close(handle->file_descriptor);
    exit(EXIT_FAILURE);
  }
  return (void *)(((uint32_t)(handle->mmaped_region)) + page_offset);
}

void arm_shared_close(arm_shared *handle) {
  if (handle == NULL) {
    fprintf(stderr, "You need to pass a valid handle to %s\n", __FUNCTION__);
    exit(EXIT_FAILURE);
  }
  if (handle->mmaped_region != MAP_FAILED) {
    munmap(handle->mmaped_region, handle->length);
  }
  if (handle->file_descriptor >= 0) {
    close(handle->file_descriptor);
  }
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef __ARM_SHARED_MEMORY_SYSTEMH_
#define __ARM_SHARED_MEMORY_SYSTEMH_

#include <stdint.h>

struct arm_shared_t {
  int file_descriptor;
  uint32_t address;
  uint32_t length;
  void *mmaped_region;
};
typedef struct arm_shared_t arm_shared;

extern void *arm_shared_init(arm_shared *handle, const uint32_t address,
                             const uint32_t length);

extern void arm_shared_close(arm_shared *handle);

#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.

```
/******************************************************************************
 *  Copyright (c) 2016, Xilinx, Inc.
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are met:
 *
 *  1.  Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 *
 *  2.  Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 *  3.  Neither the name of the copyright holder nor the names of its
 *      contributors may be used to endorse or promote products derived from
 *      this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************/

/******************************************************************************
 * @file audio_adau1761.c
 *
 * Functions to control audio controller.
 *
 * <pre>
 * MODIFICATION HISTORY:
 *
 * Ver   Who          Date     Changes
 * ----- ------------ -------- -----------------------------------------------
 * 1.00  Yun Rock Qu  12/04/17 Support for audio codec ADAU1761
 * 1.01  Yun Rock Qu  01/02/18 Enable microphone for CTIA and OMTP standards
 *
 * </pre>
 *
 ******************************************************************************/
#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>

#define SAMPLE_RATE 44100

#undef LOG_DOMAIN
#define LOG_DOMAIN "audio"

void audio_init(void) {
  config_audio_pll();
  config_audio_codec();
}

void audio_select_input(int input) {
  if (input == MIC) {
    select_mic();
  } else if (input == LINE_IN) {
    select_line_in();
  } else {
    pynq_error("audio_select_input: invalid input %d, must be LINE_IN or MIC\n",
               input);
  }
}

// Original ADAU1761 code

void write_audio_reg(unsigned char u8RegAddr, unsigned char u8Data,
                     int iic_fd) {
  unsigned char u8TxData[3];
  u8TxData[0] = 0x40;
  u8TxData[1] = u8RegAddr;
  u8TxData[2] = u8Data;
  if (writeI2C_asFile(iic_fd, u8TxData, 3) < 0) {
    pynq_error("write_audio_reg: unable to write audio register, ensure sudo "
               "chmod 666 /dev/i2c-1 has been executed. \n");
  }
}

void config_audio_pll(void) {
  int iic_index = 1;
  unsigned char u8TxData[8], u8RxData[6];
  int iic_fd;
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("config_audio_pll: unable to set I2C %d\n", iic_index);
  }

  // Disable Core Clock
  write_audio_reg(R0_CLOCK_CONTROL, 0x0E, iic_fd);
  /*  MCLK = 10 MHz
   *  R = 0100 = 4, N = 0x064C = 1612, M = 0x0C35 = 3125
   *  PLL required output = 1024x44.1 KHz = 45.1584 MHz
   *  PLLout/MCLK         = 45.1584 MHz/10 MHz = 4.51584 MHz
   *                      = R + (N/M)
   *                      = 4 + (1612/3125)
   *  Fs = PLL/1024 = 44.1 KHz
   */

  // Register write address [15:8]
  u8TxData[0] = 0x40;
  // Register write address [7:0]
  u8TxData[1] = 0x02;
  // byte 6 - M[15:8]
  u8TxData[2] = 0x0C;
  // byte 5 - M[7:0]
  u8TxData[3] = 0x35;
  // byte 4 - N[15:8]
  u8TxData[4] = 0x06;
  // byte 3 - N[7:0]
  u8TxData[5] = 0x4C;
  // byte 2 - bits 6:3 = R[3:0], 2:1 = X[1:0], 0 = PLL operation mode
  u8TxData[6] = 0x21;
  // byte 1 - 1 = PLL Lock, 0 = Core clock enable
  u8TxData[7] = 0x03;
  // Write bytes to PLL control register R1 at 0x4002
  if (writeI2C_asFile(iic_fd, u8TxData, 8) < 0) {
    pynq_error("config_audio_pll: unable to write audio register, ensure sudo "
               "chmod 666 /dev/i2c-1 has been executed. \n");
  }

  // Poll PLL Lock bit
  u8TxData[0] = 0x40;
  u8TxData[1] = 0x02;
  do {
    if (writeI2C_asFile(iic_fd, u8TxData, 2) < 0) {
      pynq_error("writeI2C_asFile: unable to write audio register, ensure sudo "
                 "chmod 666 /dev/i2c-1 has been executed. \n");
    }
    if (readI2C_asFile(iic_fd, u8RxData, 6) < 0) {
      pynq_error("readI2C_asFile: unable to write audio register, ensure sudo "
                 "chmod 666 /dev/i2c-1 has been executed. \n");
    }
  } while ((u8RxData[5] & 0x02) == 0);

  /* Clock control register:  bit 3        CLKSRC = PLL Clock input
   *                          bit 2:1      INFREQ = 1024 x fs
   *                          bit 0        COREN = Core Clock enabled
   */
  write_audio_reg(R0_CLOCK_CONTROL, 0x0F, iic_fd);

  if (unsetI2C(iic_fd) < 0) {
    pynq_error("config_audio_pll: unable to set I2C %d\n", iic_fd);
  }
}

/******************************************************************************
 * Function to configure the audio codec.
 * @param   iic_index is the i2c index in /dev list.
 * @return  none.
 *****************************************************************************/
void config_audio_codec(void) {
  int iic_index = 1;
  int iic_fd;
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("config_audio_codec: unable to set I2C %d\n", iic_index);
  }

  /*
   * Input path control registers are configured
   * in select_mic and select_line_in
   */

  // Mute Mixer1 and Mixer2 here, enable when MIC and Line In used
  write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
  write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
  // Set LDVOL and RDVOL to 21 dB and Enable left and right differential
  write_audio_reg(R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);
  write_audio_reg(R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);
  // Enable MIC bias
  write_audio_reg(R10_RECORD_MICROPHONE_BIAS_CONTROL, 0x01, iic_fd);
  // Enable ALC control and noise gate
  write_audio_reg(R14_ALC_CONTROL_3, 0x20, iic_fd);
  // Put CODEC in Master mode
  write_audio_reg(R15_SERIAL_PORT_CONTROL_0, 0x01, iic_fd);
  // Enable ADC on both channels, normal polarity and ADC high-pass filter
  write_audio_reg(R19_ADC_CONTROL, 0x33, iic_fd);
  // Mute play back Mixer3 and Mixer4 and enable when output is required
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
  // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
  write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
  // Mute left and right channels output; enable them when output is needed
  write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, 0xE5, iic_fd);
  write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, 0xE5, iic_fd);
  // Enable play back right and left channels
  write_audio_reg(R35_PLAYBACK_POWER_MANAGEMENT, 0x03, iic_fd);
  // Enable DAC for both channels
  write_audio_reg(R36_DAC_CONTROL_0, 0x03, iic_fd);
  // Set SDATA_In to DAC
  write_audio_reg(R58_SERIAL_INPUT_ROUTE_CONTROL, 0x01, iic_fd);
  // Set SDATA_Out to ADC
  write_audio_reg(R59_SERIAL_OUTPUT_ROUTE_CONTROL, 0x01, iic_fd);
  // Enable DSP and DSP Run
  write_audio_reg(R61_DSP_ENABLE, 0x01, iic_fd);
  write_audio_reg(R62_DSP_RUN, 0x01, iic_fd);
  /*
   * Enable Digital Clock Generator 0 and 1.
   * Generator 0 generates sample rates for the ADCs, DACs, and DSP.
   * Generator 1 generates BCLK and LRCLK for the serial port.
   */
  write_audio_reg(R65_CLOCK_ENABLE_0, 0x7F, iic_fd);
  write_audio_reg(R66_CLOCK_ENABLE_1, 0x03, iic_fd);

  if (unsetI2C(iic_fd) < 0) {
    pynq_error("config_audio_codec: unable to unset I2C %d\n", iic_index);
  }
}

void select_line_in(void) {
  int iic_index = 1;
  int iic_fd;
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("select_line_in: unable to set I2C %d\n", iic_index);
  }

  // Mixer 1  (left channel)
  write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
  // Enable LAUX (MX1AUXG)
  write_audio_reg(R5_RECORD_MIXER_LEFT_CONTROL_1, 0x07, iic_fd);

  // Mixer 2
  write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
  // Enable RAUX (MX2AUXG)
  write_audio_reg(R7_RECORD_MIXER_RIGHT_CONTROL_1, 0x07, iic_fd);

  if (unsetI2C(iic_fd) < 0) {
    pynq_error("select_line_in: unable to unset I2C %d\n", iic_index);
  }
}

void select_mic(void) {
  int iic_index = 1;
  int iic_fd;
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("select_mic: unable to set I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }

  // Mixer 1 (left channel)
  write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
  // LDBOOST, set to 20 dB
  write_audio_reg(R5_RECORD_MIXER_LEFT_CONTROL_1, 0x10, iic_fd);
  // LDVOL, set to 21 dB
  write_audio_reg(R8_LEFT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);

  // Mixer 2 (right channel)
  write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
  // RDBOOST, set to 20 dB
  write_audio_reg(R7_RECORD_MIXER_RIGHT_CONTROL_1, 0x10, iic_fd);
  // RDVOL, set to 21 dB
  write_audio_reg(R9_RIGHT_DIFFERENTIAL_INPUT_VOLUME_CONTROL, 0xB3, iic_fd);

  if (unsetI2C(iic_fd) < 0) {
    pynq_error("select_mic: unable to unset I2C %d\n", iic_index);
  }
}

void deselect(void) {
  int iic_index = 1;
  int iic_fd;
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("deselect: unable to set I2C %d\n", iic_index);
  }

  // mute mixer 1 in left channel
  write_audio_reg(R4_RECORD_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
  // mute mixer 2 in right channel
  write_audio_reg(R6_RECORD_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);

  if (unsetI2C(iic_fd) < 0) {
    pynq_error("deselect: unable to unset I2C %d\n", iic_index);
  }
}

void audio_bypass(unsigned int audio_mmap_size, unsigned int nsamples,
                  unsigned int volume, int uio_index) {
  if (uio_index > 2) {
    pynq_error("audio_bypass: uio_index outside of range. is %d, should be "
               "below 3. \n",
               uio_index);
  }
  if (volume > 100) {
    pynq_error("audio_bypass: volume outside allowed range. Is %d, should be "
               "below 100 \n",
               volume);
  }

  int iic_index = 1;
  int status;
  void *uio_ptr;
  int DataL, DataR;
  int iic_fd;

  uio_ptr = setUIO(uio_index, audio_mmap_size);
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("audio_bypass: unable to set I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }

  // Mute mixer1 and mixer2 input
  write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
  // Enable Mixer3 and Mixer4
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);

  unsigned char vol_register = (unsigned char)volume « 2 | 0x3;
  // Enable Left/Right Headphone out
  write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
                  iic_fd);
  write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
                  iic_fd);

  for (unsigned int i = 0; i < nsamples; i++) {
    // wait for RX data to become available
    do {
      status = *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
    } while (status == 0);
    *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
        0x00000001;

    // Read the sample from the input
    DataL = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_L_REG));
    DataR = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_R_REG));

    // Write the sample to output
    *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
    *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
  }

  write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x00, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x00, iic_fd);
  write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, 0xE5, iic_fd);
  write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, 0xE5, iic_fd);

  if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
    pynq_error("audio_bypass: unable to free UIO %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               uio_index);
  }
  if (unsetI2C(iic_fd) < 0) {
    pynq_error("audio_bypass: unable to unset I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }
}

void audio_record(unsigned int audio_mmap_size, unsigned int *BufAddr,
                  unsigned int nsamples, int uio_index) {
  if (uio_index > 2) {
    pynq_error("audio_record: uio_index outside of range. is %d, should be "
               "below 3. \n",
               uio_index);
  }
  int iic_index = 1;
  unsigned int i, status;
  void *uio_ptr;
  int DataL, DataR;
  int iic_fd;

  uio_ptr = setUIO(uio_index, audio_mmap_size);
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("audio_record: unable to set I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }

  for (i = 0; i < nsamples; i++) {
    do {
      status = *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
    } while (status == 0);
    *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
        0x00000001;

    // Read the sample from the input
    DataL = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_L_REG));
    DataR = *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_RX_R_REG));

    // Write the sample into memory
    *(BufAddr + 2 * i) = DataL;
    *(BufAddr + 2 * i + 1) = DataR;
  }

  if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
    pynq_error("audio_record: unable to free UIO %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               uio_index);
  }
  if (unsetI2C(iic_fd) < 0) {
    pynq_error("audio_record: unable to unset I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }
}

void audio_play(unsigned int audio_mmap_size, unsigned int *BufAddr,
                unsigned int nsamples, unsigned int volume, int uio_index) {
  if (uio_index > 2) {
    pynq_error(
        "audio_play: uio_index outside of range. is %d, should be below 3. \n",
        uio_index);
  }
  if (volume > 100) {
    pynq_error("audio_play: volume outside allowed range. Is %d, should be "
               "below 100 \n",
               volume);
  }
  int iic_index = 1;
  unsigned int i, status;
  void *uio_ptr;
  int DataL, DataR;
  int iic_fd;

  uio_ptr = setUIO(uio_index, audio_mmap_size);
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("audio_play: unable to set I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }

  // Unmute left and right DAC, enable Mixer3 and Mixer4
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);

  unsigned char vol_register = (unsigned char)volume « 2 | 0x3;
  // Enable Left/Right Headphone out
  write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
                  iic_fd);
  write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
                  iic_fd);

  for (i = 0; i < nsamples; i++) {
    do {
      status = *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
    } while (status == 0);
    *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
        0x00000001;

    // Read the sample from memory
    DataL = *(BufAddr + 2 * i);
    DataR = *(BufAddr + 2 * i + 1);

    // Write the sample to output
    *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
    *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
  }

  // Mute left and right DAC
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
  // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
  write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);

  if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
    pynq_error("audio_play: unable to free UIO %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               uio_index);
  }
  if (unsetI2C(iic_fd) < 0) {
    pynq_error("audio_play: unable to unset I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }
}

void audio_repeat_play(unsigned int audio_mmap_size, unsigned int *BufAddr,
                       unsigned int nsamples, unsigned int volume,
                       unsigned int repetitions) {
  if (volume > 100) {
    pynq_error("audio_repeat_play: volume outside allowed range. Is %d, should "
               "be below 100 \n",
               volume);
  }
  int iic_index = 1;
  unsigned int i, status;
  void *uio_ptr;
  int DataL, DataR;
  int iic_fd;

  uio_ptr = setUIO(0, audio_mmap_size);
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("audio_repeat_play: unable to set I2C %d, ensure sudo chmod 666 "
               "/dev/i2c-1 has been executed\n",
               iic_index);
  }

  // Unmute left and right DAC, enable Mixer3 and Mixer4
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);

  unsigned char vol_register = (unsigned char)volume « 2 | 0x3;
  // Enable Left/Right Headphone out
  write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
                  iic_fd);
  write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
                  iic_fd);

  for (unsigned int repeat = 0; repeat < repetitions; repeat++) {
    for (i = 0; i < nsamples; i++) {
      do {
        status =
            *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
      } while (status == 0);
      *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
          0x00000001;

      // Read the sample from memory
      DataL = *(BufAddr + 2 * i);
      DataR = *(BufAddr + 2 * i + 1);

      // Write the sample to output
      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
    }
  }
  // Mute left and right DAC
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
  // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
  write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);

  if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
    pynq_error("audio_repeat_play: unable to free UIO %d\n", 0);
  }
  if (unsetI2C(iic_fd) < 0) {
    pynq_error("audio_repeat_play: unable to unset I2C %d, ensure sudo chmod "
               "666 /dev/i2c-1 has been executed\n",
               iic_index);
  }
}

void audio_generate_tone(unsigned int frequency, uint32_t time_ms,
                         unsigned int volume) {

  if (frequency < 10) {
    pynq_error("audio_generate_tone: frequency should be 10 or higher, "
               "frequency is: %d\n",
               frequency);
  }
  if (volume > 100) {
    pynq_error("audio_generate_tone: volume outside allowed range. Is %d, "
               "should be below 100 \n",
               volume);
  }
  double period = 1 / ((double)(frequency));
  unsigned int samplesPerPeriod = (int)(SAMPLE_RATE * period);
  double time_s = ((double)(time_ms)) / 1000;
  int totalPeriods = (int)(time_s / period); // Number of times one period must
                                             // be played to play for time_ms

  uint32_t audioBuffer[16 * 1024 + 1] = {0};
  unsigned int i, status;

  for (i = 0; i < samplesPerPeriod; i++) {
    double t = (double)i / SAMPLE_RATE;
    double value = sin(6.28318531 * frequency * t); // 6.28... = 2pi
    value = value + 1;
    value = value * 16000;
    audioBuffer[2 * i] = (uint32_t)value;
    audioBuffer[2 * i + 1] = (uint32_t)value;
  }

  unsigned int audio_mmap_size = 64 * 1024;
  unsigned int *BufAddr = audioBuffer;
  int iic_index = 1;
  void *uio_ptr;
  int DataL, DataR;
  int iic_fd;

  uio_ptr = setUIO(0, audio_mmap_size);
  iic_fd = setI2C(iic_index, IIC_SLAVE_ADDR);
  if (iic_fd < 0) {
    pynq_error("audio_generate_tone: unable to set I2C %d, ensure sudo chmod "
               "666 /dev/i2c-1 has been executed\n",
               iic_index);
  }

  // Unmute left and right DAC, enable Mixer3 and Mixer4
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x21, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x41, iic_fd);

  unsigned char vol_register = (unsigned char)volume « 2 | 0x3;
  // Enable Left/Right Headphone out
  write_audio_reg(R29_PLAYBACK_HEADPHONE_LEFT_VOLUME_CONTROL, vol_register,
                  iic_fd);
  write_audio_reg(R30_PLAYBACK_HEADPHONE_RIGHT_VOLUME_CONTROL, vol_register,
                  iic_fd);

  for (int period = 0; period < totalPeriods; period++) {
    for (i = 0; i < samplesPerPeriod; i++) {
      do {
        status =
            *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG));
      } while (status == 0);
      *((volatile unsigned *)(((uint8_t *)uio_ptr) + I2S_STATUS_REG)) =
          0x00000001;

      // Read the sample from memory
      DataL = *(BufAddr + 2 * i);
      DataR = *(BufAddr + 2 * i + 1);

      // Write the sample to output
      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_L_REG)) = DataL;
      *((volatile int *)(((uint8_t *)uio_ptr) + I2S_DATA_TX_R_REG)) = DataR;
    }
  }
  // Mute left and right DAC
  write_audio_reg(R22_PLAYBACK_MIXER_LEFT_CONTROL_0, 0x01, iic_fd);
  write_audio_reg(R24_PLAYBACK_MIXER_RIGHT_CONTROL_0, 0x01, iic_fd);
  // Mute left input to mixer3 (R23) and right input to mixer4 (R25)
  write_audio_reg(R23_PLAYBACK_MIXER_LEFT_CONTROL_1, 0x00, iic_fd);
  write_audio_reg(R25_PLAYBACK_MIXER_RIGHT_CONTROL_1, 0x00, iic_fd);

  if (unsetUIO(uio_ptr, audio_mmap_size) < 0) {
    pynq_error("audio_generate_tone: unable to free UIO %d, ensure sudo chmod "
               "666 /dev/i2c-1 has been executed\n",
               0);
  }
  if (unsetI2C(iic_fd) < 0) {
    pynq_error("audio_generate_tone: unable to unset I2C %d, ensure has been "
               "executed\n",
               iic_index);
  }
}
```

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, unsigned 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.
```
#ifndef AUDIO_H
#define AUDIO_H
#include <stdint.h>

#define LINE_IN 0
#define MIC 1

// Slave address for the ADAU audio controller 8
#define IIC_SLAVE_ADDR 0x3B

// I2C Serial Clock frequency in Hertz
#define IIC_SCLK_RATE 400000

```

```c
// I2S Register
#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

// Audio registers
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
};

extern void audio_init(void);

extern void audio_select_input(int input);

// Original ADAU1761 code

extern void write_audio_reg(unsigned char u8RegAddr, unsigned char u8Data,
                            int iic_fd);

extern void config_audio_pll(void);

extern void config_audio_codec(void);

extern void select_line_in(void);

extern void select_mic(void);

extern void deselect(void);

extern void audio_bypass(unsigned int audio_mmap_size, unsigned int nsamples,
                         unsigned int volume, int uio_index);

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

```

/*
 * @brief Function to support audio playing 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.
 * Consecutive indexes are played synchronisly on left and right output.
 *
 * @param   audio_mmap_size is the address range of the audio codec.
 * @param   BufAddr is the buffer address.
 * @param   nsamples is the number of samples.
 * @param   uio_index is the uio index in /dev list.
 * @param   volume is the volume of the output.
 */
extern void audio_play(unsigned int audio_mmap_size, unsigned int *BufAddr,
                       unsigned int nsamples, unsigned int volume,
                       int uio_index);

extern void audio_repeat_play(unsigned int audio_mmap_size,
                              unsigned int *BufAddr, unsigned int nsamples,
                              unsigned int volume, unsigned int repetitions);

/*
 * @brief Function to generate a specific tone on the audio output.
 * @param   frequency is the frequency in Hz to be played.
 * @param   time_ms is the time the frequency should be played in ms.
 * @param   volume is the volume of the output.
 */
extern void audio_generate_tone(unsigned int frequency, uint32_t time_ms,
                                unsigned int volume);

#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) {
    pynq_error("switches_destroy: switches weren't initialized\n");
  }
}

int get_button_state(const int button) {
  if (buttons_initialized == false) {
    pynq_error("get_button_state: buttons weren't initialized\n");
  }
  if (button < 0 || button >= NUM_BUTTONS) {
    pynq_error("get_button_state: invalid button=%d, must be 0..%d-1\n",
               NUM_BUTTONS);
  }
  return (gpio_get_level(IO_BTN0 + button) == GPIO_LEVEL_LOW ? BUTTON_NOT_PUSHED
                                                             : BUTTON_PUSHED);
}

int wait_until_button_state(const int button, const int state) {
  if (buttons_initialized == false) {
    pynq_error("wait_until_button_state: buttons weren't initialized\n");
  }
  if (button < 0 || button >= NUM_BUTTONS) {
    pynq_error("get_button_state: invalid button=%d, must be 0..%d-1\n", button,
               NUM_BUTTONS);
  }
  const io_t btn = IO_BTN0 + button;
  if (gpio_get_direction(btn) != GPIO_DIR_INPUT) {
    pynq_error("get_button_state: button %d has not been set as input\n",
               button);
  }
  struct timeval call, close;
  int dTime;
  gettimeofday(&call, NULL);
  const unsigned int check =
      (state == BUTTON_NOT_PUSHED ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH);
  while (gpio_get_level(btn) != check) {
  }
  gettimeofday(&close, NULL);
  dTime = (close.tv_sec - call.tv_sec) * 1000.0;     // # of ms
  dTime += (close.tv_usec - call.tv_usec) / 1000.0; // # of usec in ms
  return dTime;
}

int sleep_msec_button_pushed(const int button, const int ms) {
  if (buttons_initialized == false) {
    pynq_error("sleep_msec_button: buttons weren't initialized\n");
  }
  if (button < 0 || button >= NUM_BUTTONS) {
    pynq_error("sleep_msec_button_pushed: invalid button=%d, must be 0..%d-1\n",
               button, NUM_BUTTONS);
  }
  const io_t btn = IO_BTN0 + button;
  if (gpio_get_direction(btn) != GPIO_DIR_INPUT) {
    pynq_error(
        "sleep_msec_button_pushed: button %d has not been set as input\n",
        button);
  }
  int status;
  struct timeval call, close;
  double dTime;
  // mapping call time to call struct
  gettimeofday(&call, NULL);
  do {
    // update level and latch if is pushed
    if (status != GPIO_LEVEL_HIGH) {
      status = gpio_get_level(btn);
    }
    (void)gettimeofday(&close, NULL);
    dTime = (close.tv_sec - call.tv_sec) * 1000.0;     // # of ms
    dTime += (close.tv_usec - call.tv_usec) / 1000.0; // # of usec in ms
  } while (dTime < ms);
  return (status == GPIO_LEVEL_LOW ? BUTTON_NOT_PUSHED : BUTTON_PUSHED);
}

void sleep_msec_buttons_pushed(int button_states[], const int ms) {
  if (buttons_initialized == false) {
    pynq_error("sleep_msec_buttons_pushed: buttons weren't initialized\n");
  }
  if (button_states == NULL) {
    pynq_error("sleep_msec_buttons_pushed: button_states is NULL\n");
  }
  struct timeval call, close;
  int dTime;
  const io_t buttons[NUM_BUTTONS] = {IO_BTN0, IO_BTN1, IO_BTN2, IO_BTN3};
  // mapping call time to call struct
  (void)gettimeofday(&call, NULL);
  do {
    for (int i = 0; i < NUM_BUTTONS; i++) {
      if (button_states[i] != BUTTON_PUSHED) {
        button_states[i] =
            (gpio_get_level(buttons[i]) == GPIO_LEVEL_HIGH ? BUTTON_PUSHED
                                                          : BUTTON_NOT_PUSHED);
      }
    }
    (void)gettimeofday(&close, NULL);
    dTime = (close.tv_sec - call.tv_sec) * 1000.0;     // # of ms
    dTime += (close.tv_usec - call.tv_usec) / 1000.0; // # of usec in ms
  } while (dTime < ms);
}

int wait_until_button_pushed(const int button) {
  // all checks are done in wait_until_button state
  return wait_until_button_state(button, BUTTON_PUSHED);
}

int wait_until_button_released(const int button) {
  // all checks are done in wait_until_button state
  return wait_until_button_state(button, BUTTON_NOT_PUSHED);
}

int wait_until_any_button_pushed(void) {
  const io_t buttons[NUM_BUTTONS] = {IO_BTN0, IO_BTN1, IO_BTN2, IO_BTN3};
  if (buttons_initialized == false) {
    pynq_error("wait_until_any_button_pushed: buttons weren't initialized\n");
  }
  for (int b = 0; b < NUM_BUTTONS; b++) {
    if (gpio_get_direction(b) != GPIO_DIR_INPUT) {
      pynq_error(
          "wait_until_any_button_pushed: button %d has not been set as input\n",
          b);
    }
  }
  do {
    for (int b = 0; b < NUM_BUTTONS; b++) {
      if (gpio_get_level(buttons[b]) == GPIO_LEVEL_HIGH) {
        return b; // we return the index, i.e. 0..NUM_BUTTONS-1
      }
    }
  } while (true);
}

int wait_until_any_button_released(void) {
  const io_t buttons[NUM_BUTTONS] = {IO_BTN0, IO_BTN1, IO_BTN2, IO_BTN3};
  if (buttons_initialized == false) {
    pynq_error("wait_until_any_button_released: buttons weren't initialized\n");
  }
  for (int b = 0; b < NUM_BUTTONS; b++) {
    if (gpio_get_direction(b) != GPIO_DIR_INPUT) {
      pynq_error("wait_until_any_button_released: button %d has not been set "
                 "as input\n",
                 b);
    }
  }
  do {
    for (int b = 0; b < NUM_BUTTONS; b++) {
      if (gpio_get_level(buttons[b]) == GPIO_LEVEL_LOW)
        return b; // we return the index, i.e. 0..NUM_BUTTONS-1
    }
  } while (true);
}

int get_switch_state(const int switch_num) {
  if (switches_initialized == false) {
    pynq_error("get_switch_state: switches weren't initialized\n");
  }
  if (switch_num != SWITCH0 && switch_num != SWITCH1) {
    pynq_error("get_switch_state: invalid switch_num=%d, must be 0..%i-1\n",
               switch_num, NUM_SWITCHES);
  }
  return (gpio_get_level(IO_SW0 + switch_num) == GPIO_LEVEL_LOW ? SWITCH_ON
                                                                : SWITCH_OFF);
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef BUTTONS_H
#define BUTTONS_H

#include <gpio.h>

#define BUTTON_NOT_PUSHED 0
#define BUTTON_PUSHED 1
#define SWITCH_OFF 0
```

```
#define SWITCH_ON 1

typedef enum { BUTTON0, BUTTON1, BUTTON2, BUTTON3, NUM_BUTTONS } button_index_t;

typedef enum { SWITCH0, SWITCH1, NUM_SWITCHES } switches_index_t;

extern void switches_init(void);

extern void switches_destroy(void);

extern void buttons_init(void);

extern void buttons_destroy(void);

extern int get_button_state(const int button);

extern int wait_until_button_state(const int button, const int state);

extern int sleep_msec_button_pushed(const int button, const int msec);

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

extern int wait_until_button_pushed(const int button);

extern int wait_until_button_released(const int button);

extern int wait_until_any_button_pushed(void);

extern int wait_until_any_button_released(void);

extern int get_switch_state(const int switch_num);

#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

Go to the documentation of this file.

```
/*
MIT License

Copyright (c) 2020

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Modified by Eindhoven University of Technology 2023.
*/
#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>

#undef LOG_DOMAIN
#define LOG_DOMAIN "display"

#define TAG "ST7789"
#define _DEBUG_ 0

#define M_PI 3.14159265358979323846

static arm_shared spi0_handle;
static volatile uint32_t *spi0 = NULL;

// states that are set for usage of the DC pin in SPI
typedef enum { SPI_Data_Mode = 1, SPI_Command_Mode = 0 } spi_mode_t;

#define GPIO_MODE_OUTPUT 1

gpio_level_t spi_to_gpio(spi_mode_t mode) {
  switch (mode) {
  case SPI_Data_Mode:
    return GPIO_LEVEL_HIGH;
  case SPI_Command_Mode:
    return GPIO_LEVEL_LOW;
  default:
    return GPIO_LEVEL_LOW;
  }
}

bool spi_master_write_command(display_t *display, uint8_t cmd) {
  gpio_set_level(display->_dc, spi_to_gpio(SPI_Command_Mode));
  spi0[0x68 / 4] = cmd;
  while (((spi0[0x64 / 4]) & 4) == 0) {
  }
  usleep(1);
  return true;
}

bool spi_master_write_data_byte(display_t *display, uint8_t data) {
  gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));

  spi0[0x68 / 4] = data;
  while (((spi0[0x64 / 4]) & 4) == 0) {
  }
  return true;
}

```

```c
bool spi_master_write_data_word(display_t *display, uint16_t data) {
  static uint8_t Byte[2];
  Byte[0] = (data » 8) & 0xFF;
  Byte[1] = data & 0xFF;
  gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
  spi0[0x68 / 4] = Byte[0];
  spi0[0x68 / 4] = Byte[1];

  while (((spi0[0x64 / 4]) & 4) == 0) {
  }
  return true;
}

bool spi_master_write_addr(display_t *display, uint16_t addr1, uint16_t addr2) {
  static uint8_t Byte[4];
  Byte[0] = (addr1 » 8) & 0xFF;
  Byte[1] = addr1 & 0xFF;
  Byte[2] = (addr2 » 8) & 0xFF;
  Byte[3] = addr2 & 0xFF;
  gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));

  // check ordering
  spi0[0x68 / 4] = Byte[0];
  spi0[0x68 / 4] = Byte[1];
  spi0[0x68 / 4] = Byte[2];
  spi0[0x68 / 4] = Byte[3];

  while (((spi0[0x64 / 4]) & 4) == 0) {
  }
  return true;
}

bool spi_master_write_color(display_t *display, uint16_t color, uint16_t size) {
  gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
  for (int i = 0; i < size; i++) {
    while (((spi0[0x64 / 4]) & 8) == 8) {
    }
    spi0[0x68 / 4] = (color » 8) & 0xFF;
    while (((spi0[0x64 / 4]) & 8) == 8) {
    }
    spi0[0x68 / 4] = (color)&0xFF;
  }
  while (((spi0[0x64 / 4]) & 4) == 0) {
  }
  return -1;
}

bool spi_master_write_colors(display_t *display, uint16_t *colors,
                             uint16_t size) {
  gpio_set_level(display->_dc, spi_to_gpio(SPI_Data_Mode));
  for (int i = 0; i < size; i++) {
    while (((spi0[0x64 / 4]) & 8) == 8) {
    }
    spi0[0x68 / 4] = (colors[i] » 8) & 0xFF;
    while (((spi0[0x64 / 4]) & 8) == 8) {
    }
    spi0[0x68 / 4] = (colors[i]) & 0xFF;
  }
  // wait till empty, then add a small extra buffer
  // because last byte we don’t exactly know when send.
  while (((spi0[0x64 / 4]) & 4) == 0) {
  }
  return true;
}

void spi_master_init(display_t *display) {
  // linking given pins in the switchbox
  switchbox_set_pin(LCD_MOSI, SWB_SPI1_MOSI);
  switchbox_set_pin(LCD_SCLK, SWB_SPI1_CLK);
  switchbox_set_pin(LCD_CS, SWB_SPI1_SS);
  switchbox_set_pin(LCD_DC, SWB_GPIO);
  switchbox_set_pin(LCD_RESET, SWB_GPIO);
  switchbox_set_pin(LCD_BL, SWB_GPIO);

  // setting the appropriate direction of each protocol pin
  gpio_set_direction(LCD_DC, GPIO_DIR_OUTPUT);
  gpio_set_direction(LCD_RESET, GPIO_DIR_OUTPUT);
  gpio_set_direction(LCD_BL, GPIO_DIR_OUTPUT);
  gpio_set_level(LCD_DC, GPIO_LEVEL_LOW);
  gpio_set_level(LCD_RESET, GPIO_LEVEL_LOW);
  gpio_set_level(LCD_BL, GPIO_LEVEL_LOW);

  // creating a shared memory instance for communicating the hardware addresses
  // of the linked pins
  spi0 = arm_shared_init(&spi0_handle, axi_quad_spi_1, 4096);
  if (_DEBUG_)
    printf("spi reset: %08X\n", spi0[0x40 / 4]);
  spi0[0x40 / 4] = 0x0000000a;
  if (_DEBUG_)
    printf("spi control: %08X\n", spi0[0x60 / 4]);
  spi0[0x60 / 4] = (1 « 4) | (1 « 3) | (1 « 2) | (1 « 1);
  if (_DEBUG_)
    printf("spi control: %08X\n", spi0[0x60 / 4]);
  if (_DEBUG_)
    printf("spi status: %08X\n", spi0[0x64 / 4]);

  // select slave 1
  spi0[0x70 / 4] = 0;
  if (_DEBUG_)
    printf("spi control: %08X\n", spi0[0x60 / 4]);
  if (_DEBUG_)
    printf("testing DISPLAY\n");
  if (_DEBUG_)
    printf("LCD_CS=%d\n", LCD_CS);
  if (LCD_CS >= 0) {
    gpio_reset_pin(LCD_CS);
    gpio_set_direction(LCD_CS, GPIO_MODE_OUTPUT);
    gpio_set_level(LCD_CS, 0);
  }

  if (_DEBUG_)
    printf("LCD_DC=%d", LCD_DC);
  gpio_reset_pin(LCD_DC);
  gpio_set_direction(LCD_DC, GPIO_MODE_OUTPUT);
  gpio_set_level(LCD_DC, 0);
  if (_DEBUG_)
    printf("LCD_RESET=%d", LCD_RESET);

  if (LCD_RESET >= 0) {
    gpio_reset_pin(LCD_RESET);
    gpio_set_direction(LCD_RESET, GPIO_MODE_OUTPUT);
    gpio_set_level(LCD_RESET, 1);
    sleep_msec(100);
    gpio_set_level(LCD_RESET, 0);
    sleep_msec(500);
    gpio_set_level(LCD_RESET, 1);
    sleep_msec(300);
  }

  if (_DEBUG_)
    printf("LCD_BL=%d", LCD_BL);
  if (LCD_BL >= 0) {
    gpio_reset_pin(LCD_BL);
    gpio_set_direction(LCD_BL, GPIO_MODE_OUTPUT);
    gpio_set_level(LCD_BL, 0);
  }

  if (_DEBUG_)
    printf("LCD_MOSI=%d", LCD_MOSI);
  if (_DEBUG_)
    printf("LCD_SCLK=%d\n", LCD_SCLK);

  display->_dc = LCD_DC;
  display->_bl = LCD_BL;
}

void displayInit(display_t *display, int width, int height, int offsetx,
                 int offsety) {
  spi_master_init(display);
  display->_width = width;
  display->_height = height;
  display->_offsetx = offsetx;
  display->_offsety = offsety;
  display->_font_direction = TEXT_DIRECTION0;
  display->_font_fill = false;
  display->_font_underline = false;

  spi_master_write_command(display, 0x01); // software Reset
  sleep_msec(150);

  spi_master_write_command(display, 0x11); // sleep Out
  sleep_msec(255);

  spi_master_write_command(display, 0x3A); // Interface Pixel Format
  spi_master_write_data_byte(display, 0x55);
  sleep_msec(10);

  spi_master_write_command(display, 0x36); // Memory Data Access Control
  spi_master_write_data_byte(display, 0x00);

  spi_master_write_command(display, 0x2A); // Column Address Set
  spi_master_write_data_byte(display, 0x00);
  spi_master_write_data_byte(display, 0x00);
  spi_master_write_data_byte(display, 0x00);
  spi_master_write_data_byte(display, 0xF0);

  spi_master_write_command(display, 0x2B); // Row Address Set
  spi_master_write_data_byte(display, 0x00);
  spi_master_write_data_byte(display, 0x00);
  spi_master_write_data_byte(display, 0x00);
  spi_master_write_data_byte(display, 0xF0);

  spi_master_write_command(display, 0x21); // Display Inversion On
  sleep_msec(10);

  spi_master_write_command(display, 0x13); // Normal Display Mode On
  sleep_msec(10);

  spi_master_write_command(display, 0x29); // Display ON
  sleep_msec(255);

  if (display->_bl >= 0) {
    gpio_set_level(display->_bl, 1);
  }
}

void display_set_flip(display_t *display, bool xflip, bool yflip) {
  if (display == NULL) {
    pynq_error("display_destroy: display has not been initialized\n");
  }
  if (display->_width != DISPLAY_WIDTH || display->_height != DISPLAY_HEIGHT) {
    pynq_error("display_destroy: internal error (wrong display hardware)\n");
  }
  spi_master_write_command(display, 0x36); // Memory Data Access Control
  uint8_t set = (yflip « 7) | (xflip « 6);
  spi_master_write_data_byte(display, set);
  if (yflip) {
    display->_offsety = 320 - display->_height;
  } else {
    display->_offsety = 0;
  }
  if (xflip) {
    display->_offsetx = 240 - display->_width;
  } else {
    display->_offsetx = 0;
  }
}

void display_init(display_t *display) {
  displayInit(display, DISPLAY_WIDTH, DISPLAY_HEIGHT, 0, 0);
  display_set_flip(display, true, true);
}

void display_destroy(display_t *display __attribute__((unused))) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("display_destroy: display has not been initialized\n");
  }
  // if channel is open
  if (spi0 != NULL) {
    (void)arm_shared_close(&spi0_handle);
    spi0 = NULL;
  }
}

void displayDrawPixel(display_t *display, uint16_t x, uint16_t y,
                      uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawPixel: display has not been initialized\n");
  }
  if (x >= display->_width || y >= display->_height) {
    pynq_error("displayDrawPixel: x=%d y=%d outside screen boundaries\n", x, y);
  }
  uint16_t _x = x + display->_offsetx;
  uint16_t _y = y + display->_offsety;

  spi_master_write_command(display, 0x2A); // set column(x) address
  spi_master_write_addr(display, _x, _x);
  spi_master_write_command(display, 0x2B); // set Page(y) address
  spi_master_write_addr(display, _y, _y);
  spi_master_write_command(display, 0x2C); // memory write
  spi_master_write_data_word(display, color);
}

void displayDrawMultiPixels(display_t *display, uint16_t x, uint16_t y,
                            uint16_t size, uint16_t *colors) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawMultiPixels: display has not been initialized\n");
  }
  if (x > display->_width || x + size > display->_width ||
      y >= display->_height) {
    pynq_error(
        "displayDrawMultiPixels: x=%d y=%d size=%d outside screen boundaries\n",
        x, y, size);
  }

  uint16_t _x1 = x + display->_offsetx;
  uint16_t _x2 = _x1 + size;
  uint16_t _y1 = y + display->_offsety;
  uint16_t _y2 = _y1;

  spi_master_write_command(display, 0x2A); // set column(x) address
  spi_master_write_addr(display, _x1, _x2);
  spi_master_write_command(display, 0x2B); // set Page(y) address
  spi_master_write_addr(display, _y1, _y2);
  spi_master_write_command(display, 0x2C); // memory write
  spi_master_write_colors(display, colors, size);
}

void displayDrawFillRect(display_t *display, uint16_t x1, uint16_t y1,
                         uint16_t x2, uint16_t y2, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawPixel: display has not been initialized\n");
  }
  if (x1 >= display->_width || x2 >= display->_width ||
      y1 >= display->_height || y2 >= display->_height) {
    pynq_error("displayDrawFillRect: x1=%d y1=%d x2=%d y2=%d outside screen "
               "boundaries\n",
               x1, y1, x2, y2);
  }
  // swapping points so that it is always plotted from x1 y1 bottom left, x2 y2
  // top right
  uint16_t x1_temp = x1, x2_temp = x2;
  uint16_t y1_temp = y1, y2_temp = y2;
  if (x1 > x2) {
    x1 = x2_temp;
    x2 = x1_temp;
  }

  if (y1 > y2) {
    y1 = y2_temp;
    y2 = y1_temp;
  }

  // printf("offset(x)=%d offset(y)=%d",display->_offsetx,display->_offsety);
  uint16_t _x1 = x1 + display->_offsetx;
  uint16_t _x2 = x2 + display->_offsetx;
  uint16_t _y1 = y1 + display->_offsety;
  uint16_t _y2 = y2 + display->_offsety;

  spi_master_write_command(display, 0x2A); // set column(x) address
  spi_master_write_addr(display, _x1, _x2);
  spi_master_write_command(display, 0x2B); // set Page(y) address
  spi_master_write_addr(display, _y1, _y2);
  spi_master_write_command(display, 0x2C); // memory write
  for (int i = _x1; i <= _x2; i++) {
    uint16_t size = _y2 - _y1 + 1;
    spi_master_write_color(display, color, size);
  }
}

void displayDisplayOff(display_t *display) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDisplayOff: display has not been initialized\n");
  }
  spi_master_write_command(display, 0x28); // display off
}

void displayDisplayOn(display_t *display) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDisplayOn: display has not been initialized\n");
  }
  spi_master_write_command(display, 0x29); // display on
}

void displayFillScreen(display_t *display, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayFillScreen: display has not been initialized\n");
  }
  displayDrawFillRect(display, 0, 0, display->_width - 1, display->_height - 1,
                      color);
}

void displayDrawLine(display_t *display, uint16_t x1, uint16_t y1, uint16_t x2,
                     uint16_t y2, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawLine: display has not been initialized\n");
  }
  if (x1 >= display->_width || y1 >= display->_height) {
    pynq_error("displayDrawLine: x1=%d y1=%d outside screen boundaries\n", x1,
               y1);
  } else if (x2 >= display->_width || y2 >= display->_height) {
    pynq_error("displayDrawLine: x2=%d y2=%d outside screen boundaries\n", x2,
               y2);
  }
  int i;
  int dx, dy;
  int sx, sy;
  int E;

  /* distance between two points */
  dx = (x2 > x1) ? x2 - x1 : x1 - x2;
  dy = (y2 > y1) ? y2 - y1 : y1 - y2;

  /* direction of two point */
  sx = (x2 > x1) ? 1 : -1;
  sy = (y2 > y1) ? 1 : -1;

  /* inclination < 1 */
  if (dx > dy) {
    E = -dx;
    for (i = 0; i <= dx; i++) {
      displayDrawPixel(display, x1, y1, color);
      x1 += sx;
      E += 2 * dy;
      if (E >= 0) {
        y1 += sy;
        E -= 2 * dx;
      }
    }

    /* inclination >= 1 */
  } else {
    E = -dy;
    for (i = 0; i <= dy; i++) {
      displayDrawPixel(display, x1, y1, color);
      y1 += sy;
      E += 2 * dx;
      if (E >= 0) {
        x1 += sx;
        E -= 2 * dy;
      }
    }
  }
}

void displayDrawRect(display_t *display, uint16_t x1, uint16_t y1, uint16_t x2,
                     uint16_t y2, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawRect: display has not been initialized\n");
  }
  if (x1 >= display->_width || y1 >= display->_height) {
    pynq_error("displayDrawRect: x1=%d y1=%d outside screen boundaries\n", x1,
               y1);
  } else if (x2 >= display->_width || y2 >= display->_height) {
    pynq_error("displayDrawRect: x2=%d y2=%d outside screen boundaries\n", x2,
               y2);
  }
  displayDrawLine(display, x1, y1, x2, y1, color);
  displayDrawLine(display, x2, y1, x2, y2, color);
  displayDrawLine(display, x2, y2, x1, y2, color);
  displayDrawLine(display, x1, y2, x1, y1, color);
}

void displayDrawRectAngle(display_t *display, uint16_t xc, uint16_t yc,
                          uint16_t w, uint16_t h, uint16_t angle,
                          uint16_t color) {
  double xd, yd, rd;
  int x1, y1;
  int x2, y2;
  int x3, y3;
  int x4, y4;
  rd = -angle * M_PI / 180.0;
  xd = 0.0 - w / 2;
  yd = h / 2;
  x1 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y1 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  yd = 0.0 - yd;
  x2 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y2 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  xd = w / 2;
  yd = h / 2;
  x3 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y3 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  yd = 0.0 - yd;
  x4 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y4 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawRectAngle: display has not been initialized\n");
  }
  if (x1 >= display->_width || y1 >= display->_height) {
    pynq_error("displayDrawRectAngle: x1=%d y1=%d outside screen boundaries\n",
               x1, y1);
  } else if (x2 >= display->_width || y2 >= display->_height) {
    pynq_error("displayDrawRectAngle: x2=%d y2=%d outside screen boundaries\n",
               x2, y2);
  } else if (x3 >= display->_width || y3 >= display->_height) {
    pynq_error("displayDrawRectAngle: x3=%d y3=%d outside screen boundaries\n",
               x3, y3);
  } else if (x4 >= display->_width || y4 >= display->_height) {
    pynq_error("displayDrawRectAngle: x4=%d y4=%d outside screen boundaries\n",
               x4, y4);
  }

  displayDrawLine(display, x1, y1, x2, y2, color);
  displayDrawLine(display, x1, y1, x3, y3, color);
  displayDrawLine(display, x2, y2, x4, y4, color);
  displayDrawLine(display, x3, y3, x4, y4, color);
}

// x1: First X coordinate of triangle point
// y1: First Y coordinate of triangle point
// x2: Second X coordinate of triangle point
// y2: Second Y coordinate of triangle point
// x3: Third X coordinate of triangle point
// y3: Third Y coordinate of triangle point
// color: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) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawTriangle: display has not been initialized\n");
  }
  if (x1 >= display->_width || y1 >= display->_height) {
    pynq_error("displayDrawRectAngle: x1=%d y1=%d outside screen boundaries\n",
               x1, y1);
  } else if (x2 >= display->_width || y2 >= display->_height) {
    pynq_error("displayDrawRectAngle: x2=%d y2=%d outside screen boundaries\n",
               x2, y2);
  } else if (x3 >= display->_width || y3 >= display->_height) {
    pynq_error("displayDrawRectAngle: x3=%d y3=%d outside screen boundaries\n",
               x3, y3);
  }

  // draw the lines for the basic triangle
  displayDrawLine(display, x1, y1, x2, y2, color);
  displayDrawLine(display, x2, y2, x3, y3, color);
  displayDrawLine(display, x3, y3, x1, y1, color);
}

// when the origin is (0, 0), the point (x1, y1) after rotating the point (x, y)
// by the angle is obtained by the following calculation.
//  x1 = x * cos(angle) - y * sin(angle)
//  y1 = x * sin(angle) + y * cos(angle)
void displayDrawTriangleCenter(display_t *display, uint16_t xc, uint16_t yc,
                               uint16_t w, uint16_t h, uint16_t angle,
                               uint16_t color) {
  double xd, yd, rd;
  int x1, y1;
  int x2, y2;
  int x3, y3;
  rd = -angle * M_PI / 180.0;
  xd = 0.0;
  yd = h / 2;
  x1 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y1 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  xd = w / 2;
  yd = 0.0 - yd;
  x2 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y2 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  xd = 0.0 - w / 2;
  x3 = (int)(xd * cos(rd) - yd * sin(rd) + xc);
  y3 = (int)(xd * sin(rd) + yd * cos(rd) + yc);

  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawTriangleCenter: display has not been initialized\n");
  }
```

```c
  if (x1 >= display->_width || y1 >= display->_height) {
    pynq_error("displayDrawRectAngle: x1=%d y1=%d outside screen boundaries\n",
               x1, y1);
  } else if (x2 >= display->_width || y2 >= display->_height) {
    pynq_error("displayDrawRectAngle: x2=%d y2=%d outside screen boundaries\n",
               x2, y2);
  } else if (x3 >= display->_width || y3 >= display->_height) {
    pynq_error("displayDrawRectAngle: x3=%d y3=%d outside screen boundaries\n",
               x3, y3);
  }

  displayDrawLine(display, x1, y1, x2, y2, color);
  displayDrawLine(display, x1, y1, x3, y3, color);
  displayDrawLine(display, x2, y2, x3, y3, color);
}

void displayDrawCircle(display_t *display, uint16_t x_center, uint16_t y_center,
                       uint16_t r, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawCircle: display has not been initialized\n");
  }
  if (r == 0) {
    pynq_error(
        "displayDrawCircle: x_center=%d y_center=%d r=%d r cannot be 0\n",
        x_center, y_center, r);
  }

  int x_max = x_center + r, x_min = x_center - r, y_max = y_center + r,
      y_min = y_center - r;

  if (x_max >= display->_width || x_min < 0 || y_max >= display->_height ||
      y_min < 0) {
    pynq_error("displayDrawCircle: x_center=%d y_center=%d r=%d outside screen "
               "boundaries\n",
               x_center, y_center, r);
  }

  int x;
  int y;
  int err;
  int old_err;

  x = 0;
  y = -r;
  err = 2 - 2 * r;
  do {
    displayDrawPixel(display, x_center - x, y_center + y, color);
    displayDrawPixel(display, x_center - y, y_center - x, color);
    displayDrawPixel(display, x_center + x, y_center - y, color);
    displayDrawPixel(display, x_center + y, y_center + x, color);
    if ((old_err = err) <= x)
      err += ++x * 2 + 1;
    if (old_err > y || err > x)
      err += ++y * 2 + 1;
  } while (y < 0);
}

void displayDrawFillCircle(display_t *display, uint16_t x_center,
                           uint16_t y_center, uint16_t r, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawFillCircle: display has not been initialized\n");
  }
  if (r == 0) {
    pynq_error(
        "displayDrawFillCircle: x_center=%d y_center=%d r=%d r cannot be 0\n",
        x_center, y_center, r);
  }

  int x_max = x_center + r, x_min = x_center - r, y_max = y_center + r,
      y_min = y_center - r;

  if (x_max >= display->_width || x_min < 0 || y_max >= display->_height ||
      y_min < 0) {
    pynq_error("displayDrawFillCircle: x_center=%d y_center=%d r=%d outside "
               "screen boundaries\n",
               x_center, y_center, r);
  }

  int x;
  int y;
  int err;
  int old_err;
  int ChangeX;

  x = 0;
  y = -r;
  err = 2 - 2 * r;
  ChangeX = 1;
  do {
    if (ChangeX) {
      displayDrawLine(display, x_center - x, y_center - y, x_center - x,
                      y_center + y, color);
      displayDrawLine(display, x_center + x, y_center - y, x_center + x,
                      y_center + y, color);
    } // endif
    ChangeX = (old_err = err) <= x;
    if (ChangeX)
      err += ++x * 2 + 1;
    if (old_err > y || err > x)
      err += ++y * 2 + 1;
  } while (y <= 0);
}

void displayDrawRoundRect(display_t *display, uint16_t x1, uint16_t y1,
                          uint16_t x2, uint16_t y2, uint16_t r,
                          uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawRoundRect: display has not been initialized\n");
  }
  if (r == 0) {
    pynq_error("displayDrawRoundRect: x_center=%d x1=%d y1=%d r cannot be 0\n",
               x1, y1, r);
  } else if (x1 >= display->_width || y1 >= display->_height) {
    pynq_error("displayDrawRoundRect: x1=%d y1=%d outside screen boundaries\n",
               x1, y1);
  } else if (x2 >= display->_width || y2 >= display->_height) {
    pynq_error("displayDrawRoundRect: x2=%d y2=%d outside screen boundaries\n",
               x2, y2);
  }
  int x;
  int y;
  int err;
  int old_err;
  unsigned char temp;

  if (x1 > x2) {
    temp = x1;
    x1 = x2;
    x2 = temp;
  }

  if (y1 > y2) {
    temp = y1;
    y1 = y2;
    y2 = temp;
  }

  if (_DEBUG_)
    printf("x1=%d x2=%d delta=%d r=%d", x1, x2, x2 - x1, r);
  if (_DEBUG_)
    printf("y1=%d y2=%d delta=%d r=%d", y1, y2, y2 - y1, r);
  if (x2 - x1 < r)
    return; // TODO add 20190517?
  if (y2 - y1 < r)
    return; // TODO add 20190517?

  x = 0;
  y = -r;
  err = 2 - 2 * r;

  do {
    if (x) {
      displayDrawPixel(display, x1 + r - x, y1 + r + y, color);
      displayDrawPixel(display, x2 - r + x, y1 + r + y, color);
      displayDrawPixel(display, x1 + r - x, y2 - r - y, color);
      displayDrawPixel(display, x2 - r + x, y2 - r - y, color);
    }
    if ((old_err = err) <= x)
      err += ++x * 2 + 1;
    if (old_err > y || err > x)
      err += ++y * 2 + 1;
  } while (y < 0);

  if (_DEBUG_)
    printf("x1+r=%d x2-r=%d", x1 + r, x2 - r);
  displayDrawLine(display, x1 + r, y1, x2 - r, y1, color);
  displayDrawLine(display, x1 + r, y2, x2 - r, y2, color);
  if (_DEBUG_)
    printf("y1+r=%d y2-r=%d", y1 + r, y2 - r);
  displayDrawLine(display, x1, y1 + r, x1, y2 - r, color);
  displayDrawLine(display, x2, y1 + r, x2, y2 - r, color);
}

uint16_t rgb_conv(uint16_t r, uint16_t g, uint16_t b) {
  return (((r & 0xF8) « 8) | ((g & 0xFC) « 3) | (b » 3));
}

int displayDrawChar(display_t *display, FontxFile *fxs, uint16_t x, uint16_t y,
                    uint8_t ascii, uint16_t color) {
  uint16_t xx, yy, bit, ofs;
  unsigned char fonts[128]; // font pattern
  unsigned char pw, ph;
  int h, w;
  uint16_t mask;
  bool rc = GetFontx(fxs, ascii, fonts, &pw, &ph);

  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawChar: display has not been initialized\n");
  }
  if (_DEBUG_) {
    printf("_font_direction=%d\n", display->_font_direction);
    printf("GetFontx rc=%d pw=%d ph=%d\n", rc, pw, ph);
  }

  if (!rc) {
    pynq_error("displayDrawChar: cannot get font from font file\n");
  }

  switch (display->_font_direction) {
  case TEXT_DIRECTION0:
    if (x + pw >= display->_width || y + ph >= display->_height) {
      pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
                 "direction=%d outside screen boundaries\n",
                 x, y, ph, pw, display->_font_direction);
    }
    break;
  case TEXT_DIRECTION90:
    if (x + ph >= display->_height || y + pw >= display->_width) {
      pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
                 "direction=%d outside screen boundaries\n",
                 x, y, ph, pw, display->_font_direction);
    }
    break;
  case TEXT_DIRECTION180:
    if (x - pw <= 0 || y - ph <= 0) {
      pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
                 "direction=%d outside screen boundaries\n",
                 x, y, ph, pw, display->_font_direction);
    }
    break;
  case TEXT_DIRECTION270:
    if (x - ph <= 0 || y - pw <= 0) {
      pynq_error("displayDrawChar: x=%d y=%d for font height=%d width=%d and "
                 "direction=%d outside screen boundaries\n",
                 x, y, ph, pw, display->_font_direction);
    }
    break;
  }

  int16_t xd1 = 0, yd1 = 0, xd2 = 0, yd2 = 0;
  uint16_t xss = 0, yss = 0;
  int16_t xsd = 0, ysd = 0, next = 0;
  uint16_t x0 = 0, x1 = 0, y0 = 0, y1 = 0;
  if (display->_font_direction == 0) {
    xd1 = +1;
    yd1 = +1; //-1;
    xd2 = 0;
    yd2 = 0;
    xss = x;
    yss = y - (ph - 1);
    xsd = 1;
    ysd = 0;
    next = x + pw;

    x0 = x;
    y0 = y - (ph - 1);
    x1 = x + (pw - 1);
    y1 = y;
  } else if (display->_font_direction == 2) {
    xd1 = -1;
    yd1 = -1; //+1;
    xd2 = 0;
    yd2 = 0;
    xss = x;
    yss = y + ph + 1;
    xsd = 1;
    ysd = 0;
    next = x - pw;

    x0 = x - (pw - 1);
    y0 = y;
    x1 = x;
    y1 = y + (ph - 1);
  } else if (display->_font_direction == 1) {
    xd1 = 0;
    yd1 = 0;
    xd2 = -1;
    yd2 = +1; //-1;
    xss = x + ph;
    yss = y;
    xsd = 0;
    ysd = 1;
    next = y + pw; // y - pw;

    x0 = x;
    y0 = y;
    x1 = x + (ph - 1);
    y1 = y + (pw - 1);
  } else if (display->_font_direction == 3) {
    xd1 = 0;
    yd1 = 0;
    xd2 = +1;
    yd2 = -1; //+1;
    xss = x - (ph - 1);
    yss = y;
    xsd = 0;
    ysd = 1;
    next = y - pw; // y + pw;

    x0 = x - (ph - 1);
    y0 = y - (pw - 1);
    x1 = x;
    y1 = y;
  }

  // TODO: fix the problem of underflow properly some time
  if (display->_font_fill && x0 < DISPLAY_WIDTH && y0 < DISPLAY_HEIGHT &&
      x1 < DISPLAY_WIDTH && y1 < DISPLAY_HEIGHT) {
    displayDrawFillRect(display, x0, y0, x1, y1, display->_font_fill_color);
  }

  int bits;
  if (_DEBUG_)
    printf("xss=%d yss=%d\n", xss, yss);
  ofs = 0;
  yy = yss;
  xx = xss;
  for (h = 0; h < ph; h++) {
    if (xsd)
      xx = xss;
    if (ysd)
      yy = yss;
    bits = pw;
    for (w = 0; w < ((pw + 4) / 8); w++) {
      mask = 0x80;
      for (bit = 0; bit < 8; bit++) {
        bits--;
        if (bits < 0)
          continue;
        // TODO: fix the problem of underflow properly some time
        if (fonts[ofs] & mask && xx < DISPLAY_WIDTH && yy < DISPLAY_HEIGHT) {
          displayDrawPixel(display, xx, yy, color);
        }
        // TODO: fix the problem of underflow properly some time
        if (h == (ph - 2) && display->_font_underline && xx < DISPLAY_WIDTH &&
            yy < DISPLAY_HEIGHT)
          displayDrawPixel(display, xx, yy, display->_font_underline_color);
        // TODO: fix the problem of underflow properly some time
        if (h == (ph - 1) && display->_font_underline && xx < DISPLAY_WIDTH &&
            yy < DISPLAY_HEIGHT)
          displayDrawPixel(display, xx, yy, display->_font_underline_color);
        xx = xx + xd1;
        yy = yy + yd2;
        mask = mask » 1;
      }
      ofs++;
    }
    yy = yy + yd1;
    xx = xx + xd2;
  }

  if (next < 0)
    next = 0;
  return next;
}

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

```
  int length = strlen((char *)ascii);
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayDrawString: display has not been initialized\n");
  }
  if (_DEBUG_)
    printf("displayDrawString length=%d\n", length);
  for (int i = 0; i < length; i++) {
    if (_DEBUG_)
      printf("ascii[%d]=%x x=%d y=%d\n", i, ascii[i], x, y);
    if (display->_font_direction == 0)
      x = displayDrawChar(display, fx, x, y, ascii[i], color);
    if (display->_font_direction == 1)
      y = displayDrawChar(display, fx, x, y, ascii[i], color);
    if (display->_font_direction == 2)
      x = displayDrawChar(display, fx, x, y, ascii[i], color);
    if (display->_font_direction == 3)
      y = displayDrawChar(display, fx, x, y, ascii[i], color);
  }
  if (display->_font_direction == 0)
    return x;
  if (display->_font_direction == 2)
    return x;
  if (display->_font_direction == 1)
    return y;
  if (display->_font_direction == 3)
    return y;
  return 0;
}

void displaySetFontDirection(display_t *display, uint16_t dir) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displaySetFontDirection: display has not been initialized\n");
  }
  display->_font_direction = dir;
}

void displaySetFontFill(display_t *display, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displaySetFontFill: display has not been initialized\n");
  }
  display->_font_fill = true;
  display->_font_fill_color = color;
}

void displayUnsetFontFill(display_t *display) { display->_font_fill = false; }

void displaySetFontUnderLine(display_t *display, uint16_t color) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displaySetFontUnderLine: display has not been initialized\n");
  }
  display->_font_underline = true;
  display->_font_underline_color = color;
}

void displayUnsetFontUnderLine(display_t *display) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayUnsetFontUnderLine: display has not been initialized\n");
  }
  display->_font_underline = false;
}

void displayBacklightOff(display_t *display) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayBacklightOff: display has not been initialized\n");
  }
  if (display->_bl >= 0) {
    gpio_set_level(display->_bl, 0);
  }
}

void displayBacklightOn(display_t *display) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayBacklightOn: display has not been initialized\n");
  }
  if (display->_bl >= 0) {
    gpio_set_level(display->_bl, 1);
  }
}

void displayInversionOff(display_t *display) {
  if (display == NULL || display->_width != DISPLAY_WIDTH) {
    pynq_error("displayInversionOff: display has not been initialized\n");
  }
  spi_master_write_command(display, 0x21); // display Inversion Off
}

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

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.

```
/*
MIT License

Copyright (c) 2020

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Modified by Eindhoven University of Technology 2023.
*/
#ifndef SCREEN_H
#define SCREEN_H

#include <fontx.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>

#define DISPLAY_HEIGHT 240
#define DISPLAY_WIDTH 240

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
```

```c
};

enum directions {
  TEXT_DIRECTION0 = 0,
  TEXT_DIRECTION90 = 1,
  TEXT_DIRECTION180 = 2,
  TEXT_DIRECTION270 = 3,
  NUM_TEXT_DIRECTIONS
};

typedef struct {
  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;
} display_t;

extern void display_init(display_t *display);

extern void display_destroy(display_t *display);

extern void displayDrawPixel(display_t *display, uint16_t x, uint16_t y,
                             uint16_t color);

extern void displayDrawFillRect(display_t *display, uint16_t x1, uint16_t y1,
                                uint16_t x2, uint16_t y2, uint16_t color);

extern void displayFillScreen(display_t *display, uint16_t color);

extern void displayDrawLine(display_t *display, uint16_t x1, uint16_t y1,
                            uint16_t x2, uint16_t y2, uint16_t color);

extern void displayDrawRect(display_t *display, uint16_t x1, uint16_t y1,
                            uint16_t x2, uint16_t y2, uint16_t color);

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

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

extern void displayDrawCircle(display_t *display, uint16_t x_center,
                              uint16_t y_center, uint16_t r, uint16_t color);

extern void displayDrawFillCircle(display_t *display, uint16_t x_center,
                                  uint16_t y_center, uint16_t r,
                                  uint16_t color);

extern void displayDrawRoundRect(display_t *display, uint16_t x1, uint16_t y1,
                                 uint16_t x2, uint16_t y2, uint16_t r,
                                 uint16_t color);

extern uint16_t rgb_conv(uint16_t r, uint16_t g, uint16_t b);

extern int displayDrawChar(display_t *display, FontxFile *fx, uint16_t x,
                           uint16_t y, uint8_t ascii, uint16_t color);

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

extern void displaySetFontDirection(display_t *display, uint16_t dir);

extern void displaySetFontFill(display_t *display, uint16_t color);

extern void displayUnsetFontFill(display_t *display);

extern void displaySetFontUnderLine(display_t *display, uint16_t color);

extern void displayUnsetFontUnderLine(display_t *display);

extern void displayDisplayOff(display_t *display);

extern void displayDisplayOn(display_t *display);

extern void displayBacklightOff(display_t *display);

extern void displayBacklightOn(display_t *display);

extern void displayInversionOff(display_t *display);

extern void displayInversionOn(display_t *display);

extern 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);
#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;
```

```
    }
    fx->opened = true;
    fx->file = f;
    char buf[18];
    if (fread(buf, 1, sizeof(buf), fx->file) != sizeof(buf)) {
      fx->valid = false;
      printf("Fontx:%s not FONTX format.\n", fx->path);
      fclose(fx->file);
      return fx->valid;
    }

    if (FontxDebug) {
      for (uint32_t i = 0; i < strlen(buf); i++) {
        printf("buf[%d]=0x%x\n", i, buf[i]);
      }
    }
    memcpy(fx->fxname, &buf[6], 8);
    fx->w = buf[14];
    fx->h = buf[15];
    fx->is_ank = (buf[16] == 0);
    fx->bc = buf[17];
    fx->fsz = (fx->w + 7) / 8 * fx->h;
    if (fx->fsz > FontxGlyphBufSize) {
      printf("Fontx:%s is too big font size.\n", fx->path);
      fx->valid = false;
      fclose(fx->file);
      return fx->valid;
    }
    fx->valid = true;
  }
  return fx->valid;
}

void CloseFontx(FontxFile *fx) {
  if (fx->opened) {
    fclose(fx->file);
    fx->opened = false;
  }
}

void DumpFontx(FontxFile *fxs) {
  for (int i = 0; i < 2; i++) {
    printf("fxs[%d]->path=%s\n", i, fxs[i].path);
    printf("fxs[%d]->opened=%d\n", i, fxs[i].opened);
    printf("fxs[%d]->fxname=%s\n", i, fxs[i].fxname);
    printf("fxs[%d]->valid=%d\n", i, fxs[i].valid);
    printf("fxs[%d]->is_ank=%d\n", i, fxs[i].is_ank);
    printf("fxs[%d]->w=%d\n", i, fxs[i].w);
    printf("fxs[%d]->h=%d\n", i, fxs[i].h);
    printf("fxs[%d]->fsz=%d\n", i, fxs[i].fsz);
    printf("fxs[%d]->bc=%d\n", i, fxs[i].bc);
  }
}

uint8_t getFortWidth(FontxFile *fx) {
  printf("fx->w=%d\n", fx->w);
  return (fx->w);
}

uint8_t getFortHeight(FontxFile *fx) {
  printf("fx->h=%d\n", fx->h);
  return (fx->h);
}

bool GetFontx(FontxFile *fxs, uint8_t ascii, uint8_t *pGlyph, uint8_t *pw,
              uint8_t *ph) {
  int i;
  uint32_t offset;

  if (FontxDebug)
    printf("[GetFontx]ascii=0x%x\n", ascii);
  for (i = 0; i < 2; i++) {
    if (!OpenFontx(&fxs[i]))
      continue;
    if (FontxDebug)
      printf("[GetFontx]openFontxFile[%d] ok\n", i);

    if (fxs[i].is_ank) {
      if (FontxDebug)
        printf("[GetFontx]fxs.is_ank fxs.fsz=%d\n", fxs[i].fsz);
      offset = 17 + ascii * fxs[i].fsz;
      if (FontxDebug)
        printf("[GetFontx]offset=%d\n", offset);
      if (fseek(fxs[i].file, offset, SEEK_SET)) {
        printf("Fontx:seek(%u) failed.\n", offset);
        return false;
      }
      if (fread(pGlyph, 1, fxs[i].fsz, fxs[i].file) != fxs[i].fsz) {
        printf("Fontx:fread failed.\n");
        return false;
      }
      if (pw)
        *pw = fxs[i].w;
      if (ph)
        *ph = fxs[i].h;
      return true;
    }
  }
  return false;
}

void Font2Bitmap(uint8_t *fonts, uint8_t *line, uint8_t w, uint8_t h,
                 uint8_t inverse) {
  int x, y;
  for (y = 0; y < (h / 8); y++) {
    for (x = 0; x < w; x++) {
      line[y * 32 + x] = 0;
    }
  }

  int mask = 7;
  int fontp;
  fontp = 0;
  for (y = 0; y < h; y++) {
    for (x = 0; x < w; x++) {
      uint8_t d = fonts[fontp + x / 8];
      uint8_t linep = (y / 8) * 32 + x;
      if (d & (0x80 » (x % 8)))
        line[linep] = line[linep] + (1 « mask);
    }
    mask--;
    if (mask < 0)
      mask = 7;
    fontp += (w + 7) / 8;
  }

  if (inverse) {
    for (y = 0; y < (h / 8); y++) {
      for (x = 0; x < w; x++) {
        line[y * 32 + x] = RotateByte(line[y * 32 + x]);
      }
    }
  }
}

void UnderlineBitmap(uint8_t *line, uint8_t w, uint8_t h) {
  int x, y;
  uint8_t wk;
  for (y = 0; y < (h / 8); y++) {
    for (x = 0; x < w; x++) {
      wk = line[y * 32 + x];
      if ((y + 1) == (h / 8))
        line[y * 32 + x] = wk + 0x80;
    }
  }
}

void ReversBitmap(uint8_t *line, uint8_t w, uint8_t h) {
  int x, y;
  uint8_t wk;
  for (y = 0; y < (h / 8); y++) {
    for (x = 0; x < w; x++) {
      wk = line[y * 32 + x];
      line[y * 32 + x] = ~wk;
    }
  }
}

void ShowFont(uint8_t *fonts, uint8_t pw, uint8_t ph) {
  int x, y, fpos;
  printf("[ShowFont pw=%d ph=%d]\n", pw, ph);
  fpos = 0;
  for (y = 0; y < ph; y++) {
    printf("%02d", y);
    for (x = 0; x < pw; x++) {
      if (fonts[fpos + x / 8] & (0x80 » (x % 8))) {
        printf("*");
      } else {
        printf(".");
      }
    }
    printf("\n");
    fpos = fpos + (pw + 7) / 8;
  }
  printf("\n");
}

void ShowBitmap(uint8_t *bitmap, uint8_t pw, uint8_t ph) {
  int x, y, fpos;
  printf("[ShowBitmap pw=%d ph=%d]\n", pw, ph);

  fpos = 0;
  for (y = 0; y < ph; y++) {
    printf("%02d", y);
    for (x = 0; x < pw; x++) {

      if (bitmap[x + (y / 8) * 32] & (0x80 » fpos)) {
        printf("*");
      } else {
        printf(".");
      }
    }
    printf("\n");
    fpos++;
    if (fpos > 7)
      fpos = 0;
  }
  printf("\n");
}

uint8_t RotateByte(uint8_t ch1) {
  uint8_t ch2 = 0;
  int j;
  for (j = 0; j < 8; j++) {
    ch2 = (ch2 « 1) + (ch1 & 0x01);
    ch1 = ch1 » 1;
  }
  return ch2;
}
```

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);
```

```

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);

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#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>

static arm_shared gpio_handle, intc0_handle;
volatile uint32_t *gpio = NULL;
volatile uint32_t *intc0 = NULL;

bool gpio_is_initialized(void) {
  /* if gpio == NULL we know we are not inialized */
  return (gpio != NULL) ? true : false;
}

void gpio_init(void) {
  pynq_info("Initialize");
  check_version();
  gpio = arm_shared_init(&gpio_handle, axi_gpio_0, 4096);
  intc0 = arm_shared_init(&intc0_handle, axi_intc_0, 4096);
}

void gpio_destroy(void) {
  pynq_info("Destroy");
  arm_shared_close(&gpio_handle);
  arm_shared_close(&intc0_handle);
  gpio = NULL;
  intc0 = NULL;
}

void gpio_reset_pin(const io_t pin) {
  PIN_CHECK(pin);
  pynq_info("Reset pin: %d", pin);
  gpio_set_direction(pin, GPIO_DIR_INPUT);
  gpio_set_level(pin, GPIO_LEVEL_LOW);
}

void gpio_reset(void) {
  pynq_info("Reset all pins");
  // set all pins as input
  gpio[1] = 0xFFFFFFFF;
  // re-set all outputs to 0
  gpio[0] = 0x0;

  // set all pins as input
  gpio[3] = 0xFFFFFFFF;
  // re-set all outputs to 0
  gpio[2] = 0x0;
  // disable all interrupts
  intc0[0] = 0;
  intc0[1] = 0;
  // remove all pending interrupts
  intc0[2] = 0;
  intc0[3] = 0;
}

void gpio_set_direction(const io_t pin, const gpio_direction_t dir) {
  PIN_CHECK(pin);
  if (!(dir == GPIO_DIR_INPUT || dir == GPIO_DIR_OUTPUT)) {
    pynq_error("gpio_set_direction: invalid direction %d", dir);
  }
  int pin_bank = pin % 32;
  int bank = pin < 32 ? 1 : 3;
  if (dir == GPIO_DIR_INPUT) {
    gpio[bank] = gpio[bank] | (1 « pin_bank);
  } else {
    gpio[bank] = gpio[bank] & ~(1 « pin_bank);
  }
}

gpio_direction_t gpio_get_direction(const io_t pin) {
  PIN_CHECK(pin);
  int pin_bank = pin % 32;
  int bank = pin < 32 ? 1 : 3;
  int dir =
      ((gpio[bank] & (1 « pin_bank)) != 0) ? GPIO_DIR_INPUT : GPIO_DIR_OUTPUT;
  return dir;
}

void gpio_set_level(const io_t pin, const gpio_level_t level) {
  PIN_CHECK(pin);
  if (!(level == GPIO_LEVEL_HIGH || level == GPIO_LEVEL_LOW)) {
    pynq_error("gpio_set_level: level %d is invalid", level);
  }
  int pin_bank = pin % 32;
  int bank = pin < 32 ? 0 : 2;
  if (level == GPIO_LEVEL_HIGH) {
    gpio[bank] = gpio[bank] | (1 « pin_bank);
  } else {
    gpio[bank] = gpio[bank] & ~(1 « pin_bank);
  }
}

gpio_level_t gpio_get_level(const io_t pin) {
  PIN_CHECK(pin);
  int pin_bank = pin % 32;
  int bank = pin < 32 ? 0 : 2;
  return (gpio[bank] & (1 « pin_bank)) != 0 ? GPIO_LEVEL_HIGH : GPIO_LEVEL_LOW;
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef GPIO_H
#define GPIO_H
#include <pinmap.h>
#include <stdbool.h>
#include <stdint.h>

typedef enum {
  GPIO_DIR_INPUT = 0,
  GPIO_DIR_OUTPUT = 1
} gpio_direction_t;

typedef enum {
  GPIO_LEVEL_LOW = 0,
  GPIO_LEVEL_HIGH = 1
} gpio_level_t;

extern void gpio_init(void);
extern void gpio_destroy(void);

extern void gpio_reset_pin(const io_t pin);

extern void gpio_set_direction(const io_t pin,
                               const gpio_direction_t direction);

extern gpio_direction_t gpio_get_direction(const io_t pin);

extern void gpio_set_level(const io_t pin, const gpio_level_t level);

extern gpio_level_t gpio_get_level(const io_t pin);

extern void gpio_reset(void);

extern bool gpio_is_initialized(void);
#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.

```
/******************************************************************************
 *  Copyright (c) 2016, Xilinx, Inc.
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are met:
 *
 *  1.  Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 *
 *  2.  Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 *  3.  Neither the name of the copyright holder nor the names of its
 *      contributors may be used to endorse or promote products derived from
 *      this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
 *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *****************************************************************************/
/******************************************************************************
 *
 *
 * @file i2cps.c
 *
 * Functions to interact with linux I2C. No safe checks here, so users must
 * know what they are doing.
 *
 * <pre>
 * MODIFICATION HISTORY:
 *
 * Ver   Who  Date     Changes
 * ----- --- -------- -----------------------------------------------
 * 1.00a gn  02/03/16 release
 * 1.00b yrq 08/31/16 add license header
 *
 * </pre>
 *
 *****************************************************************************/

#include "i2cps.h"
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>

int setI2C(unsigned int index, long slave_addr) {
  int i2c_fd;
  char buf[50];
  sprintf(buf, "/dev/i2c-%d", index);
  // printf("buf = %s \n",buf);
  if ((i2c_fd = open(buf, O_RDWR)) < 0) {
    return -1;
  }
  if (ioctl(i2c_fd, I2C_SLAVE, slave_addr) < 0) {
    return -1;
  }
  return i2c_fd;
}

int unsetI2C(int i2c_fd) {
  close(i2c_fd);
  return 0;
}

int writeI2C_asFile(int i2c_fd, unsigned char writebuffer[],
                    unsigned char bytes) {
  unsigned char bytesWritten = write(i2c_fd, writebuffer, bytes);
  if (bytes != bytesWritten) {
    return -1;
  }
  return 0;
}

int readI2C_asFile(int i2c_fd, unsigned char readbuffer[],
                   unsigned char bytes) {
  unsigned char bytesRead = read(i2c_fd, readbuffer, bytes);
  if (bytes != bytesRead)
    return -1;
  return 0;
}
```

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.

```
/******************************************************************************
 *  Copyright (c) 2016, Xilinx, Inc.
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are met:
 *
 *  1.  Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 *
 *  2.  Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 *  3.  Neither the name of the copyright holder nor the names of its
 *      contributors may be used to endorse or promote products derived from
 *      this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
 *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *****************************************************************************/

/******************************************************************************/
#ifndef __I2CPS_H__
#define __I2CPS_H__

#include <linux/i2c-dev.h>

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);

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

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

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
```

```
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#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>

#define IIC_TIMEOUT 5
typedef enum {
  IIC_IDLE = 0,
  IIC_ADDRESS = 1,
  IIC_READ = 2,
  IIC_WRITE = 3

} IICState;

typedef struct IICHandle {
  arm_shared mem_handle;
  volatile uint32_t *ptr;

  // Register interface for slave mode.
  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;
} IICHandle;

static IICHandle iic_handles[NUM_IICS] = {
    {.ptr = NULL,

     .saddr = 0,
     .register_map = NULL,
     .register_map_length = 0,
     .selected_register = 0,
     .state = IIC_IDLE,
     .addressed = 0},
};

#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

void iic_init(const iic_index_t iic) {
  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d\n", iic, NUM_IICS);
  }
  if (iic == IIC0) {
    iic_handles[iic].ptr =
        arm_shared_init(&((iic_handles[iic].mem_handle)), axi_iic_0, 4096);
  } else if (iic == IIC1) {
    iic_handles[iic].ptr =
        arm_shared_init(&((iic_handles[iic].mem_handle)), axi_iic_1, 4096);
  }
  // Reset
  (iic_handles[iic].ptr[IIC_REG_SOFT_RESET / 4]) = 0xA;
  usleep(1000);
}

void iic_destroy(const iic_index_t iic) {
  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
  }
  if (iic_handles[iic].ptr == NULL) {
    pynq_error("IIC%d has not been initialized.\n", iic);
  }
  arm_shared_close(&((iic_handles[iic].mem_handle)));
  iic_handles[iic].ptr = NULL;
}

bool iic_set_slave_mode(const iic_index_t iic, const uint8_t addr,
                        uint32_t *register_map, const uint32_t rm_length) {
  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
  }
  if (iic_handles[iic].ptr == NULL) {
    pynq_error("IIC%d has not been initialized.\n", iic);
  }
  (iic_handles[iic].saddr) = addr;
  (iic_handles[iic].ptr[IIC_ADR_REG_OFFSET / 4]) = addr « 1;
  uint32_t ctr_reg = (iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4]);
  // Clear the master bit.
  ctr_reg &= ~(IIC_CR_MSMS_MASK);
  // Enable IIC
  ctr_reg |= IIC_CR_ENABLE_DEVICE_MASK;

  (iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4]) = ctr_reg;
  (iic_handles[iic].ptr[IIC_RFD_REG_OFFSET / 4]) = 0x0;

  iic_handles[iic].register_map = register_map;
  iic_handles[iic].register_map_length = rm_length;

  return true;
}

static inline void iic_clear_isr_mask(const iic_index_t iic, uint32_t mask) {

  (iic_handles[iic].ptr[IIC_IISR_OFFSET / 4]) =
      (iic_handles[iic].ptr[IIC_IISR_OFFSET / 4]) & mask;
}

static void iic_flush_tx_fifo(const iic_index_t iic) {
  IICHandle *handle = &(iic_handles[iic]);
  uint32_t reg = handle->ptr[IIC_CR_REG_OFFSET / 4];
  handle->ptr[IIC_CR_REG_OFFSET / 4] = reg | IIC_CR_TX_FIFO_RESET_MASK;
  handle->ptr[IIC_CR_REG_OFFSET / 4] = reg;
}

static void iic_tx_error_handler(const iic_index_t iic) {
  IICHandle *handle = &(iic_handles[iic]);
  iic_flush_tx_fifo(iic);
  iic_clear_isr_mask(iic, IIC_INTR_RX_FULL_MASK | IIC_INTR_TX_HALF_MASK |
                              IIC_INTR_TX_ERROR_MASK | IIC_INTR_TX_EMPTY_MASK);

  uint32_t reg = handle->ptr[IIC_CR_REG_OFFSET / 4];
  handle->ptr[IIC_CR_REG_OFFSET / 4] = reg & ~IIC_CR_MSMS_MASK;
}
static void iic_slave_master_write(const iic_index_t iic, const uint32_t c) {
  IICHandle *handle = &(iic_handles[iic]);
  uint32_t v = (c « (handle->recv_cnt) * 8);
  handle->new_val |= v;
  handle->recv_cnt++;
  // If we have one full word, write it back to register.
  if (handle->recv_cnt == 4) {
    handle->register_map[handle->selected_register %
                         handle->register_map_length] = handle->new_val;
    // go to idle mode.
    handle->state = IIC_IDLE;
  }
}

static void iic_slave_master_read(const iic_index_t iic) {
  IICHandle *handle = &(iic_handles[iic]);
  if (handle->state == IIC_ADDRESS) {
    handle->state = IIC_WRITE;
  }
  if (handle->state == IIC_WRITE) {
    uint32_t r = (handle->register_map[handle->selected_register %
                                       handle->register_map_length]);
    uint8_t c = (r » ((handle->recv_cnt) * 8)) & 0xFF;
    (iic_handles[iic].ptr[IIC_DTR_REG_OFFSET / 4]) = c;
    handle->recv_cnt++;
    if (handle->recv_cnt == 4) {
      // printf("1\n");
      handle->state = IIC_IDLE;
    }
    // modulo 4;
    handle->recv_cnt &= 0x03;
  }
};
static void iic_interrupt_handle(const iic_index_t iic) {
  time_t start = time(NULL);
  IICHandle *handle = &(iic_handles[iic]);
  int loop = 1;
  uint32_t sr_reg = (handle->ptr[IIC_SR_REG_OFFSET / 4]);
  do {
    time_t now = time(NULL);
    uint32_t nisr = (handle->ptr[IIC_IISR_OFFSET / 4]);
    uint32_t clear = 0;
    uint32_t isr = 0;
    isr = nisr;
    if (isr & IIC_INTR_ARB_LOST_MASK) {

      clear = IIC_INTR_ARB_LOST_MASK;
    } else if (isr & IIC_INTR_TX_ERROR_MASK) {
      iic_tx_error_handler(iic);
      handle->state = IIC_IDLE;
      clear = IIC_INTR_TX_ERROR_MASK;
    } else if (isr & IIC_INTR_RX_FULL_MASK) {
      // if there is data in outgoing fifo, flush this.
      uint8_t d = handle->ptr[IIC_DRR_REG_OFFSET / 4];

      uint32_t reg = handle->ptr[IIC_CR_REG_OFFSET / 4];
      reg &= ~IIC_CR_NO_ACK_MASK;
      handle->ptr[IIC_CR_REG_OFFSET / 4] = reg;
      switch (handle->state) {
      case IIC_IDLE:
        handle->recv_cnt = 0;
        handle->new_val = 0;
        handle->selected_register = d;
        handle->state = IIC_ADDRESS;
        break;
      case IIC_ADDRESS:
        handle->state = IIC_WRITE;
        // FALLTHROUGH
      case IIC_WRITE:
        iic_slave_master_write(iic, d);
        start = now;
        break;
      default:
        pynq_warning("unhandled");
        break;
      }

      clear = IIC_INTR_RX_FULL_MASK;
    } else if (handle->addressed && (isr & IIC_INTR_NAAS_MASK)) {
      handle->addressed = 0;

      clear = IIC_INTR_NAAS_MASK;
    } else if (!handle->addressed && (isr & IIC_INTR_AAS_MASK)) {
      handle->addressed = 1;
      clear = IIC_INTR_AAS_MASK;
    } else if (isr & IIC_INTR_BNB_MASK) {
      loop = 0;

      clear = IIC_INTR_BNB_MASK;
    } else if (isr & (IIC_INTR_TX_EMPTY_MASK | IIC_INTR_TX_HALF_MASK)) {

      if (handle->state == IIC_ADDRESS || handle->state == IIC_WRITE) {
        if (sr_reg & IIC_SR_MSTR_RDING_SLAVE_MASK) {
          iic_slave_master_read(iic);
          start = now;
        }
      }
      clear = isr & (IIC_INTR_TX_EMPTY_MASK | IIC_INTR_TX_HALF_MASK);
    }

    if ((now - start) > IIC_TIMEOUT) {
      pynq_warning("IIC timeout, resetting bus.");
      iic_reset(iic);
      iic_clear_isr_mask(iic, 0xFF);
      uint32_t ctr_reg = (handle->ptr[IIC_CR_REG_OFFSET / 4]);
      (iic_handles[iic].ptr[IIC_ADR_REG_OFFSET / 4]) = handle->saddr « 1;
      // Clear the master bit.
      ctr_reg &= ~(IIC_CR_MSMS_MASK);
      // Enable IIC
      ctr_reg |= IIC_CR_ENABLE_DEVICE_MASK;

      (handle->ptr[IIC_CR_REG_OFFSET / 4]) = ctr_reg;
      loop = 0;
    }
    //(iic_handles[iic].ptr[IIC_IISR_OFFSET / 4]) = nisr;
    iic_clear_isr_mask(iic, clear);
    sr_reg = (handle->ptr[IIC_SR_REG_OFFSET / 4]);
  } while (loop && (sr_reg & IIC_SR_BUS_BUSY_MASK));
  // iic_clear_isr_mask(iic, 0xFF);
}
void iic_slave_mode_handler(const iic_index_t iic) {

  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
  }
  if (iic_handles[iic].ptr == NULL) {
    pynq_error("IIC%d has not been initialized.\n", iic);
  }
  iic_interrupt_handle(iic);
  return;
}

void iic_reset(const iic_index_t iic) {
  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
  }
  if (iic_handles[iic].ptr == NULL) {
    pynq_error("IIC%d has not been initialized.\n", iic);
  }
  iic_handles[iic].ptr[IIC_REG_SOFT_RESET / 4] = 0x0A;
  uint32_t reg = iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4];
  iic_handles[iic].ptr[IIC_CR_REG_OFFSET / 4] =
      reg & ~IIC_CR_REPEATED_START_MASK;
}

bool iic_read_register(const iic_index_t iic, const uint8_t addr,
                       const uint8_t reg, uint8_t *data, uint16_t data_length) {
  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
  }
  if (iic_handles[iic].ptr == NULL) {
    pynq_error("IIC%d has not been initialized.\n", iic);
  }
  if (XIic_Send((UINTPTR)iic_handles[iic].ptr, addr, (u8 *)&reg, 1,
                XIIC_REPEATED_START) != 1) {
    return 1;
  }
  uint8_t ByteCount = XIic_Recv((UINTPTR)iic_handles[iic].ptr, addr, data,
                                data_length, XIIC_STOP);
  return (ByteCount == data_length) ? 0 : 1;
}

bool iic_write_register(const iic_index_t iic, const uint8_t addr,
                        const uint8_t reg, uint8_t *data,
                        uint16_t data_length) {
  if (!(iic >= IIC0 && iic < NUM_IICS)) {
    pynq_error("invalid IIC %d, must be 0..%d-1\n", iic, NUM_IICS);
  }
  if (iic_handles[iic].ptr == NULL) {
    pynq_error("IIC%d has not been initialized.\n", iic);
  }
  uint8_t buffer[1 + data_length];
  buffer[0] = reg;
  memcpy(&(buffer[1]), data, data_length);
  uint8_t ByteCount = XIic_Send((UINTPTR)iic_handles[iic].ptr, addr,
                                &(buffer[0]), 1 + data_length, XIIC_STOP);
  return (ByteCount == (data_length + 1)) ? 0 : 1;
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef IIC_H
#define IIC_H
#include <stdbool.h>
#include <stdint.h>

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

extern void iic_init(const iic_index_t iic);

extern void iic_destroy(const iic_index_t iic);

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

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

```

extern bool iic_set_slave_mode(const iic_index_t iic, const uint8_t addr,
                               uint32_t *register_map,
                               const uint32_t rm_length);

extern void iic_slave_mode_handler(const iic_index_t iic);

extern void iic_reset(const iic_index_t iic);
#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.

```c
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#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>

#define DOMAIN "Interrupt"

extern uint32_t *gpio;
extern uint32_t *intc0;

static bool gpio_initialized = false;

void check_initialization(void) {
  if (gpio_initialized == false) {
    pynq_error("Interrupts have not been initialized. Call "
               "gpio_interupt_init() first.\n");
  }
}

int gpio_interrupt_init(void) {
  int fd = open("/dev/uio1", O_RDWR, O_CLOEXEC);
  if (fd < 0) {
    pynq_error("failed to open interrupts\n");
  }
  int32_t m = 1;
  write(fd, &m, 4);
  gpio_initialized = true;
  return fd;
}

void gpio_enable_interrupt(const io_t pin) {
  check_initialization();
  int pin_bank = pin % 32;
  int bank = pin < 32 ? 0 : 1;
  if (bank == 0) {
    printf("interrupt set 0: %08X %08X\r\n", pin, pin_bank);
    intc0[0] |= (1 « pin_bank);
  } else {
    printf("interrupt set 1: %08X %08X\r\n", pin, pin_bank);
    intc0[1] |= (1 « (pin_bank));
  }
}

void gpio_disable_interrupt(const io_t pin) {
  check_initialization();
  intc0[0] &= ~(1 « pin);
}

void gpio_disable_all_interrupts(void) {
  check_initialization();
  intc0[0] = 0;
  intc0[1] = 0;
}

```

```
uint64_t gpio_get_interrupt(void) {
  check_initialization();
  uint64_t retv = intc0[3];
  retv «= 32;
  retv |= intc0[2];
  return retv;
}

void gpio_ack_interrupt(void) {
  check_initialization();
  intc0[2] = 1;
}

void verify_interrupt_request(const io_t pin) {
  // TODO check if interrupts are initialized when using other interrupt
  // functions
  uint64_t retv = intc0[1];
  retv «= 32;
  retv |= intc0[0];
  if (pin < 64) {
    uint64_t bitMask = 1ULL « pin;
    if (!(bitMask & retv)) {
      pynq_error("Pin %d is not enabled. Enable by using "
                 "gpio_enable_interrupt(pin). \n",
                 pin);
    }
  } else {
    if (retv == 0) {
      pynq_error("No interrupts enabled. Enable by using "
                 "gpio_enable_interrupt(pin). \n");
    }
  }
}

void gpio_print_interrupt(void) {
  check_initialization();
  //  printf("11c: %08X\r\n", gpio[0x11c / 4]);
  //  printf("128: %08X\r\n", gpio[0x128 / 4]);
  //  printf("120: %08X\r\n", gpio[0x120 / 4]);
  printf("interrupt 0: %08X %08X\r\n", intc0[0], intc0[2]);
  printf("interrupt 1: %08X %08X\r\n", intc0[1], intc0[3]);
}

void findSetBitPositions(uint64_t word, uint8_t *positions) {
  int index = 0;
  int count = 0;
  while (word) {
    if (word & 1) {
      positions[count++] = index;
    }
    word »= 1;
    index++;
  }
}

void gpio_wait_for_interrupt(const io_t pin) {
  check_initialization();
  verify_interrupt_request(pin);
  if (pin > 63) {
    while (1) {
      uint64_t interrupt = gpio_get_interrupt();
      if (interrupt != 0) {
        break;
      }
    }
  } else {
    while (1) {
      uint64_t interrupt = gpio_get_interrupt();
      uint64_t bitMask = 1ULL « pin;
      if (bitMask & interrupt) {
        break;
      }
      sleep_msec(100);
    }
  }
}

uint8_t *gpio_get_interrupt_pins(uint8_t *positions) {
  check_initialization();
  verify_interrupt_request(64); // check if any interupt pin is enabled
  // uint8_t *positions = (uint8_t *)malloc(64 * sizeof(uint8_t));
  uint64_t pin = (uint64_t)((uint64_t)(intc0[3]) « 32 | intc0[2]);
  findSetBitPositions(pin, positions);
  // printf("Interrupted pin(s): ");
  bool empty = true;
  for (int i = 0; i < 64; i++) {
    if (positions[i] != 0) {
      empty = false;
      // printf("%d ", positions[i]);
      break;
    }
  }
  if (empty) {
    printf("WARNING: gpio_get_interrupt_pins: No pins interrupted. ");
  }
  printf("\n");
  return (positions);
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef INTERRUPT_H
#define INTERRUPT_H

#include <gpio.h>

extern int gpio_interrupt_init(void);

extern void gpio_ack_interrupt(void);

extern void verify_interrupt_request(const io_t pin);

extern void gpio_print_interrupt(void);

extern void gpio_enable_interrupt(const io_t pin);

extern void gpio_disable_interrupt(const io_t pin);

extern void gpio_disable_all_interrupts(void);

extern uint64_t gpio_get_interrupt(void);

extern uint8_t *gpio_get_interrupt_pins(uint8_t *positions);

extern void gpio_wait_for_interrupt(const io_t pin);

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <gpio.h>
#include <leds.h>
```

```
#include <log.h>
#include <pinmap.h>
#include <pwm.h>
#include <stdio.h>
#include <stdlib.h>

#undef LOG_DOMAIN
#define LOG_DOMAIN "leds"

typedef enum _led_mode { uninitialized, binary, pwm_green, pwm_color } led_mode;
static led_mode mode = uninitialized;

// LEDs are either on or off
void leds_init_onoff(void) {
  if (mode == binary)
    return;
  if (mode != uninitialized) {
    pynq_error("leds_init_onoff: mode=%d should be uninitialized\n", mode);
  }
  gpio_set_direction(IO_LD0, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD1, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD2, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD3, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD4B, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD4G, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD4R, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD5B, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD5G, GPIO_DIR_OUTPUT);
  gpio_set_direction(IO_LD5R, GPIO_DIR_OUTPUT);
  mode = binary;
}

// can change the intensity of LEDs, the onoff parameters are then in the range
// 0..255
void green_leds_init_pwm(void) {
  if (mode == pwm_green)
    return;
  if (mode != uninitialized) {
    pynq_error("green_leds_init_pwm: mode=%d should be uninitialized\n", mode);
  }
  // initialize switchbox and routing PWM to LEDs
  switchbox_set_pin(IO_LD0, SWB_PWM0);
  switchbox_set_pin(IO_LD1, SWB_PWM1);
  switchbox_set_pin(IO_LD2, SWB_PWM2);
  switchbox_set_pin(IO_LD3, SWB_PWM3);
  // initialize the PWM channels
  pwm_init(PWM0, 256);
  pwm_init(PWM1, 256);
  pwm_init(PWM2, 256);
  pwm_init(PWM3, 256);
  mode = pwm_green;
}

// can change the intensity of LEDs, the onoff parameters are then in the range
// 0..255
void color_leds_init_pwm(void) {
  if (mode == pwm_color)
    return;
  if (mode != uninitialized) {
    pynq_error("color_leds_init_pwm: mode=%d should be uninitialized\n", mode);
  }
  // initialize switchbox and routing PWM to LEDs
  switchbox_set_pin(IO_LD4R, SWB_PWM0);
  switchbox_set_pin(IO_LD4G, SWB_PWM1);
  switchbox_set_pin(IO_LD4B, SWB_PWM2);
  // initialize the PWM channels
  pwm_init(PWM0, 256);
  pwm_init(PWM1, 256);
  pwm_init(PWM2, 256);
  mode = pwm_color;
}

void leds_destroy(void) {
  // note that pynq_destroy will also reset all GPIO and switch off all LEDs
  if (mode == binary) {
    for (int i = 0; i < NUM_GREEN_LEDS; i++)
      green_led_off(i);
  }
  if (mode == pwm_green || mode == pwm_color) {
    green_led_off(0);
    green_led_off(1);
    green_led_off(2);
    pwm_destroy(PWM0);
    pwm_destroy(PWM1);
    pwm_destroy(PWM2);
  }
  if (mode == pwm_green) {
    green_led_off(3);
    pwm_destroy(PWM3);
  }
  mode = uninitialized;
}

void green_led_onoff(const int led, const int onoff) {
  if (led < 0 || led >= NUM_GREEN_LEDS) {
    pynq_error("green_led_onoff: invalid led=%d, must be 0..%d-1\n",
               NUM_GREEN_LEDS);
  }
  int oo = onoff;
  switch (mode) {
  case binary:
    gpio_set_level(IO_LD0 + led,
                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
    break;
  case pwm_green:
  case pwm_color:
    if (onoff < 0) {
      oo = 0;
    } else {
      if (onoff > 255) {
        oo = 255;
      }
    }
    pwm_set_duty_cycle(PWM0 + led, oo);
    break;
  default:
    pynq_error("green_led_onoff: LEDs have not been initialized with "
               "green_leds_init_pwm\n");
    break;
  }
}

void green_led_on(const int led) { green_led_onoff(led, LED_ON); }
void green_led_off(const int led) { green_led_onoff(led, LED_OFF); }
void color_led_red_onoff(const int onoff) {
  int oo = onoff;
  switch (mode) {
  case binary:
    gpio_set_level(IO_LD4R,
                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
    break;
  case pwm_green:
  case pwm_color:
    if (onoff < 0) {
      oo = 0;
    } else {
      if (onoff > 255) {
        oo = 255;
      }
    }
    pwm_set_duty_cycle(PWM0, oo);
    break;
  default:
    pynq_error("color_led_red_onoff: LEDs have not been initialized with "
               "color_leds_init_pwm\n");
  }
}

void color_led_green_onoff(const int onoff) {
  int oo = onoff;
  switch (mode) {
  case binary:
    gpio_set_level(IO_LD4G,
                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
    break;
  case pwm_color:
    if (onoff < 0) {
      oo = 0;
    } else {
      if (onoff > 255) {
        oo = 255;
      }
    }
    pwm_set_duty_cycle(PWM1, oo);
    break;
  default:
    pynq_error("color_led_green_onoff: LEDs have not been initialized with "
               "color_leds_init_pwm\n");
  }
}

void color_led_blue_onoff(const int onoff) {
  int oo = onoff;
  switch (mode) {
```

```
  case binary:
    gpio_set_level(IO_LD4B,
                   (onoff == LED_OFF ? GPIO_LEVEL_LOW : GPIO_LEVEL_HIGH));
    break;
  case pwm_color:
    if (onoff < 0) {
      oo = 0;
    } else {
      if (onoff > 255) {
        oo = 255;
      }
    }
    pwm_set_duty_cycle(PWM2, oo);
    break;
  default:
    pynq_error("color_led_blue_onoff: LEDs have not been initialized with "
               "color_leds_init_pwm\n");
  }
}

void color_led_onoff(const int red_onoff, const int green_onoff,
                     const int blue_onoff) {
  color_led_red_onoff(red_onoff);
  color_led_green_onoff(green_onoff);
  color_led_blue_onoff(blue_onoff);
}

void color_led_on(void) { color_led_onoff(LED_ON, LED_ON, LED_ON); }
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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef LED_H
#define LED_H

#include <gpio.h>
#include <pinmap.h>

typedef enum {
  LED0,
  LED1,
  LED2,
  LED3,
  NUM_GREEN_LEDS,
} green_led_index_t;

typedef enum {
  COLOR_LED0,
  COLOR_LED1,
  NUM_COLOR_LEDS,
} color_led_index_t;

#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

extern void leds_init_onoff(void);

extern void green_leds_init_pwm(void);

extern void color_leds_init_pwm(void);

extern void leds_destroy(void);

extern void green_led_onoff(const int led, const int onoff);

extern void green_led_on(const int led);

extern void green_led_off(const int led);

extern void color_led_red_onoff(const int onoff);

extern void color_led_green_onoff(const int onoff);

extern void color_led_blue_onoff(const int onoff);

extern void color_led_onoff(const int red_onoff, const int green_onoff,
                            const int blue_onoff);

extern void color_led_on(void);

extern void color_led_off(void);

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
```

```
SOFTWARE.
*/
#include "libpynq.h"

void pynq_init(void) {
  gpio_init();
  gpio_reset();
  switchbox_init();
  switchbox_reset();

  // set line buffering on the output, should help with logging
  //  setlinebuf(stdout);
  //  setlinebuf(stderr);
}

void pynq_destroy(void) {
  gpio_reset();
  gpio_destroy();
  switchbox_reset();
  switchbox_destroy();
}
```

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  )
```

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  )
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef PYNQLIB_H
#define PYNQLIB_H

#ifdef __cplusplus
extern "C" {
#endif // all of your legacy C code here

// standard libraries
#include <stdbool.h>
#include <stdint.h>

// library > (...)
#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>

// platform > (...)
#include <lcdconfig.h>
#include <platform.h>

extern void pynq_init(void);

extern void pynq_destroy(void);

#ifdef __cplusplus
}
#endif

#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

Go to the documentation of this file.

```c
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "log.h"

#define DOMAIN "LOGGER"

static const char color_escape_calls[NUM_LOG_LEVELS][8] = {
    "\033[1;32m",
    "\033[1;33m",
    "\033[1;31m"};
static const char log_level_name[NUM_LOG_LEVELS][10] = {
    "INFO:    ", "WARNING: ", "ERROR:   "};
static const char color_escape_blue[] = "\033[1;34m";
static const char color_escape_reset[] = "\033[0m";

static bool pynq_log_init = false;
static LogLevel critical_level = LOG_LEVEL_ERROR;
static LogLevel min_log_level = LOG_LEVEL_WARNING;

void pynq_log(const LogLevel level, char const *domain, char const *location,
              unsigned int lineno, char const *fmt, ...) {
  va_list arg_list;

  // on first call, initialize based on input arguments
  if (!pynq_log_init) {
    // if DEBUG is set, we also print log level INFO
    char const *env = getenv("DEBUG");
    if (env != NULL) {
      min_log_level = LOG_LEVEL_INFO;
    }
    // make warnings fatal
    env = getenv("FATAL_WARNING");
    if (env != NULL) {
      critical_level = LOG_LEVEL_WARNING;
    }
    pynq_log_init = true;
  }
  // check if the log level is valid
  if (level < LOG_LEVEL_INFO || level > LOG_LEVEL_ERROR) {
    printf("pynq_log: invalid log level specified (%d)\r\n", level);
    return;
  }

  if (level < min_log_level) {
    return;
  }
  fputs(color_escape_calls[level], stderr);
  fputs(log_level_name[level], stderr);

  fputs(color_escape_blue, stderr);
  if (domain != NULL) {
    fprintf(stderr, "%s::", domain);
  }
  fprintf(stderr, "%s:%d ", location, lineno);
  fputs(color_escape_reset, stderr);

  va_start(arg_list, fmt);
  vfprintf(stderr, fmt, arg_list);
  va_end(arg_list);
  if (fmt[strlen(fmt) - 1] != '\n') {
    fputs("\n", stderr);
  }

  if (level >= critical_level) {
    abort();
  }
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef LOG_H
#define LOG_H

#define LOG_DOMAIN NULL
typedef enum LogLevel {
  LOG_LEVEL_INFO,
  LOG_LEVEL_WARNING,
  LOG_LEVEL_ERROR,
  NUM_LOG_LEVELS
} LogLevel;

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

#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(...)                                                        \
  do {                                                                         \
    pynq_log(LOG_LEVEL_ERROR, LOG_DOMAIN, __FUNCTION__, __LINE__,              \
             __VA_ARGS__);                                                     \
    for (;;)                                                                   \
      ;                                                                        \
  } while (0)

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <pinmap.h>

char *const pin_names[] = {
    "AR0",
    "AR1",
    "AR2",
    "AR3",
    "AR4",
    "AR5",
    "AR6",
    "AR7",
    "AR8",
    "AR9",
    "AR10",
    "AR11",
    "AR12",
    "AR13",

    "A0",
    "A1",
    "A2",
    "A3",
    "A4",
    "A5",

    "SW0",
    "SW1",
    "BTN0",
    "BTN1",
    "BTN2",
    "BTN3",
    "LD0",
    "LD1",
    "LD2",
    "LD3",

    "AR_SDA",
    "AR_SCL",
    "LD4B",
    "LD4G",
    "LD4R",
    "LD5B",
    "LD5G",
    "LD5R",
    "RBPI40",
    "RBPI37",
    "RBPI38",
    "RBPI35",
    "RBPI36",
    "RBPI33",
    "RBPI18",
    "RBPI32",
    "RBPI10",
    "RBPI27",
    "RBPI28",
    "RBPI22",
    "RBPI23",
    "RBPI24",
    "RBPI21",
    "RBPI26",
    "RBPI19",
    "RBPI31",
    "RBPI15",
    "RBPI16",
    "RBPI13",
    "RBPI12",
    "RBPI29",
    "RBPI08",
    "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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef PINMAP_H
#define PINMAP_H

#define NUM_ANALOG_REFERENCE_PINS 14 /* # analog reference pins */
#define NUM_ANALOG_IN_PINS 6         /* # analog input pins */

typedef enum {
  IO_AR0 = 0,   /* reference pin 0 */
  IO_AR1 = 1,   /* reference pin 1 */
  IO_AR2 = 2,   /* reference pin 2 */
  IO_AR3 = 3,   /* reference pin 3 */
  IO_AR4 = 4,   /* reference pin 4 */
  IO_AR5 = 5,   /* reference pin 5 */
  IO_AR6 = 6,   /* reference pin 6 */
  IO_AR7 = 7,   /* reference pin 7 */
  IO_AR8 = 8,   /* reference pin 8 */
  IO_AR9 = 9,   /* reference pin 9 */
  IO_AR10 = 10, /* reference pin 10 */
  IO_AR11 = 11, /* reference pin 11 */
  IO_AR12 = 12, /* reference pin 12 */
  IO_AR13 = 13, /* reference pin 13 */

  IO_A0 = 14, /* analog input pin 0 */
  IO_A1 = 15, /* analog input pin 1 */
  IO_A2 = 16, /* analog input pin 2 */
  IO_A3 = 17, /* analog input pin 3 */
  IO_A4 = 18, /* analog input pin 4 */
  IO_A5 = 19, /* analog input pin 5 */

  IO_SW0 = 20, /* switch input pin 0 */
  IO_SW1 = 21, /* switch input pin 1 */

  IO_BTN0 = 22, /* button input pin 0 */
  IO_BTN1 = 23, /* button input pin 1 */
  IO_BTN2 = 24, /* button input pin 2 */
  IO_BTN3 = 25, /* button input pin 3 */

  IO_LD0 = 26, /* LED output pin 0 */
  IO_LD1 = 27, /* LED output pin 1 */
  IO_LD2 = 28, /* LED output pin 2 */
  IO_LD3 = 29, /* LED output pin 3 */

  IO_AR_SCL = 31, /* I2C clock pin */
  IO_AR_SDA = 30, /* I2C data pin */

  IO_LD4B = 32, /* color LED 0 blue input pin */
  IO_LD4R = 33, /* color LED 0 red input pin */
  IO_LD4G = 34, /* color LED 0 green input pin */

  IO_LD5B = 35, /* color LED 1 blue input pin */
  IO_LD5R = 36, /* color LED 1 red input pin */
  IO_LD5G = 37, /* color LED 1 green input pin */

  IO_RBPI40 = 38, /* RaspberryPi header pin */
  IO_RBPI37 = 39, /* RaspberryPi header pin */
  IO_RBPI38 = 40, /* RaspberryPi header pin */
  IO_RBPI35 = 41, /* RaspberryPi header pin */
  IO_RBPI36 = 42, /* RaspberryPi header pin */
  IO_RBPI33 = 43, /* RaspberryPi header pin */
  IO_RBPI18 = 44, /* RaspberryPi header pin */
  IO_RBPI32 = 45, /* RaspberryPi header pin */
  IO_RBPI10 = 46, /* RaspberryPi header pin */
  IO_RBPI27 = 47, /* RaspberryPi header pin */
  IO_RBPI28 = 48, /* RaspberryPi header pin */
  IO_RBPI22 = 49, /* RaspberryPi header pin */
  IO_RBPI23 = 50, /* RaspberryPi header pin */
  IO_RBPI24 = 51, /* RaspberryPi header pin */
  IO_RBPI21 = 52, /* RaspberryPi header pin */
  IO_RBPI26 = 53, /* RaspberryPi header pin */
  IO_RBPI19 = 54, /* RaspberryPi header pin */
  IO_RBPI31 = 55, /* RaspberryPi header pin */
  IO_RBPI15 = 56, /* RaspberryPi header pin */
  IO_RBPI16 = 57, /* RaspberryPi header pin */
  IO_RBPI13 = 58, /* RaspberryPi header pin */
  IO_RBPI12 = 59, /* RaspberryPi header pin */
  IO_RBPI29 = 60, /* RaspberryPi header pin */
  IO_RBPI08 = 61, /* RaspberryPi header pin */
  IO_RBPI07 = 62, /* RaspberryPi header pin */
  IO_RBPI05 = 63, /* RaspberryPi header pin */

  IO_NUM_PINS = 64,
} io_t;

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

extern char *const pin_names[64];
#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <libpynq.h>

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

static struct arm_shared_t channels[NUM_PWMS] = {
    0,
};
static volatile uint32_t *initializedChannel[NUM_PWMS] = {
    NULL,
};

bool pwm_initialized(const int pwm) {
  if (pwm < 0 || pwm >= NUM_PWMS) {
    pynq_error("pwm_initialized: invalid pwm=%d, must be 0..%d-1\n", pwm,
               NUM_PWMS);
  }
  if (initializedChannel[pwm] == NULL) {
    return false;
  }
  return true;
}

bool check_initialized_pwm(const int pwm) {
  if (pwm < 0 || pwm >= NUM_PWMS) {
    pynq_error("pwm_initialized: invalid pwm=%d, must be 0..%d-1\n", pwm,
               NUM_PWMS);
  }
  if (initializedChannel[pwm] == NULL) {
    pynq_error("pwm_initialized: channel of pwm %d has not been initialized\n",
               pwm);
  }
  return true;
}

void pwm_init(const int pwm, const uint32_t period) {
  if (pwm < 0 || pwm >= NUM_PWMS) {
    pynq_error("pwm_init: invalid pwm=%d, must be 0..%d-1\n", pwm, NUM_PWMS);
  }
  uint32_t channelAddr = axi_pwm_base + (pwm * 0x10000);
  initializedChannel[pwm] = arm_shared_init(&channels[pwm], channelAddr, 512);
  initializedChannel[pwm][PWM_REG_DUTY] = 0;
  initializedChannel[pwm][PWM_REG_PERIOD] = period;
  initializedChannel[pwm][PWM_REG_NEW_STEP_COUNT] = -1;
}

void pwm_destroy(const int pwm) {
  (void)check_initialized_pwm(pwm);
  arm_shared_close(&channels[pwm]);
  initializedChannel[pwm] = NULL;
}

uint32_t pwm_get_duty_cycle(const int pwm) {
  (void)check_initialized_pwm(pwm);
  return initializedChannel[pwm][PWM_REG_DUTY];
}

uint32_t pwm_get_period(const int pwm) {
  (void)check_initialized_pwm(pwm);
  return initializedChannel[pwm][PWM_REG_PERIOD];
}

void pwm_set_period(const int pwm, const uint32_t period) {
  (void)check_initialized_pwm(pwm);
  initializedChannel[pwm][PWM_REG_PERIOD] = period;
}

void pwm_set_duty_cycle(const int pwm, const uint32_t duty) {
  (void)check_initialized_pwm(pwm);
  initializedChannel[pwm][PWM_REG_DUTY] = duty;
}

uint32_t pwm_get_steps(const int pwm) {
  (void)check_initialized_pwm(pwm);
  return initializedChannel[pwm][PWM_REG_NEW_STEP_COUNT];
}

void pwm_set_steps(const int pwm, const uint32_t steps) {
  (void)check_initialized_pwm(pwm);
  initializedChannel[pwm][PWM_REG_NEW_STEP_COUNT] = steps;
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef PWM_H
#define PWM_H
#include <libpynq.h>

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

extern bool pwm_initialized(const int pwm);

extern void pwm_init(const int pwm, const uint32_t period);

extern void pwm_destroy(const int pwm);

extern void pwm_set_duty_cycle(const int pwm, const uint32_t duty);

extern void pwm_set_period(const int pwm, const uint32_t period);

uint32_t pwm_get_period(const int pwm);

extern uint32_t pwm_get_duty_cycle(const int pwm);

extern void pwm_set_steps(const int pwm, const uint32_t steps);

extern uint32_t pwm_get_steps(const int pwm);

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
```

```
#include "switchbox.h"
#include <libpynq.h>

char *const switchbox_names[NUM_SWITCHBOX_NAMES] = {
    "SWB_GPIO",
    "SWB_Interrupt_In",
    "SWB_UART0_TX",
    "SWB_UART0_RX",
    "SWB_SPI0_CLK",
    "SWB_SPI0_MISO",
    "SWB_SPI0_MOSI",
    "SWB_SPI0_SS",
    "SWB_SPI1_CLK",
    "SWB_SPI1_MISO",
    "SWB_SPI1_MOSI",
    "SWB_SPI1_SS",
    "SWB_IIC0_SDA",
    "SWB_IIC0_SCL",
    "SWB_IIC1_SDA",
    "SWB_IIC1_SCL",
    "SWB_PWM0",
    "SWB_PWM1",
    "SWB_PWM2",
    "SWB_PWM3",
    "SWB_PWM4",
    "SWB_PWM5",
    "SWB_TIMER_G0",
    "SWB_TIMER_G1",
    "SWB_TIMER_G2",
    "SWB_TIMER_G3",
    "SWB_TIMER_G4",
    "SWB_TIMER_G5",
    "SWB_TIMER_G6",
    "SWB_TIMER_G7",
    "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",
};

arm_shared ioswitch_handle;
volatile uint32_t *ioswitch = NULL;

typedef struct {
  char *name;
  char *state;
  io_configuration_t channel; // was uint8_t
} pin;

void switchbox_init(void) {
  // allocate shared memory for the switchbox and store the pointer in
  // `ioswitch`
  check_version();
  ioswitch = arm_shared_init(&ioswitch_handle, io_switch_0, 4096);
}

void switchbox_destroy(void) {
  // free the sared memory in the switchbox
  arm_shared_close(&ioswitch_handle);
}

// reset all switchbox pins to 0
void switchbox_reset(void) {
  // 32 pins to remap, 4 per word.
  for (uint_fast32_t i = 0; i < (64 / 4); i++) {
    // set all words to 0
    ioswitch[i] = 0;
  }
}

void switchbox_set_pin(const io_t pin_number,
                       const io_configuration_t io_type) {
  int numWordstoPass, byteNumber;
  uint32_t switchConfigValue;

  PIN_CHECK(pin_number);

  // If gpio is initialized, set the pin as input, if PIN_TYPE is
  // not gpio
  if (io_type != SWB_GPIO && gpio_is_initialized()) {
    // set pin as input.
    if (gpio_get_direction(pin_number) != GPIO_DIR_INPUT) {
      pynq_warning("pin: %s is set as GPIO ouput, but not mapped as GPIO. "
                   "Reconfiguring as input.",
                   pin_names[pin_number]);
      gpio_set_direction(pin_number, GPIO_DIR_INPUT);
    }
  }

  // calculate the word and byte number for the given pin number
  numWordstoPass = pin_number / 4;
  byteNumber = pin_number % 4;

  // get the current value of the word containing the pin
  switchConfigValue = ioswitch[numWordstoPass];

  // clear the byte containing the pin type and set it to the new value
  switchConfigValue = (switchConfigValue & (~(0xFF « (byteNumber * 8)))) |
                      (io_type « (byteNumber * 8));

  // update the word in the switchbox with the new value
  ioswitch[numWordstoPass] = switchConfigValue;
}

// pin_number: the number of the pin to get
// returns: the type of the given pin
io_configuration_t switchbox_get_pin(const io_t pin_number) {
  int numWordstoPass, byteNumber;
  uint32_t switchConfigValue;

  PIN_CHECK(pin_number);

  // calculate the word and byte number for the given pin number
  numWordstoPass = pin_number / 4;
  byteNumber = pin_number % 4;

  // get the value of the word containing the pin and extract the value of the
  // byte containing the pin type
  switchConfigValue = ioswitch[numWordstoPass];
  switchConfigValue = (switchConfigValue » (byteNumber * 8)) & 0xFF;

  // return pintype
  return switchConfigValue;
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef SWITCHBOX_H
#define SWITCHBOX_H
#include <pinmap.h>
#include <stdbool.h>
#include <stdint.h>

typedef 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,
} io_configuration_t;

#define NUM_SWITCHBOX_NAMES 40
extern char *const switchbox_names[NUM_SWITCHBOX_NAMES];

extern void switchbox_init(void);

extern void switchbox_set_pin(const io_t pin_number,
                              const io_configuration_t pin_type);

extern void switchbox_reset(void);

extern void switchbox_destroy(void);

extern io_configuration_t switchbox_get_pin(const io_t pin_number);

#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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include "uart.h"
#include "arm_shared_memory_system.h"
```

```
#include "log.h"
#include <platform.h>
#include <stdio.h>

#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)

static arm_shared uart_handles[NUM_UARTS];
static volatile uint32_t *uart_ptrs[NUM_UARTS] = {
    NULL,
};

void uart_init(const int uart) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart == UART0) {
    uart_ptrs[uart] =
        arm_shared_init(&(uart_handles[uart]), axi_uartlite_0, 4096);
  } else if (uart == UART1) {
    uart_ptrs[uart] =
        arm_shared_init(&(uart_handles[uart]), axi_uartlite_1, 4096);
  }
}

void uart_destroy(const int uart) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart_ptrs[uart] == NULL) {
    pynq_error("UART%d has not been initialized.\n", uart);
  }
  arm_shared_close(&(uart_handles[uart]));
  uart_ptrs[uart] = NULL;
}

void uart_send(const int uart, const uint8_t data) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart_ptrs[uart] == NULL) {
    pynq_error("UART%d has not been initialized.\n", uart);
  }
  while ((uart_ptrs[uart][UART_REG_STATUS] &
          UART_REG_STATUS_BIT_TX_FIFO_FULL) == UART_REG_STATUS_BIT_TX_FIFO_FULL)
    ;
  uart_ptrs[uart][UART_REG_TRANSMIT_FIFO] = data;
}

uint8_t uart_recv(const int uart) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart_ptrs[uart] == NULL) {
    pynq_error("UART%d has not been initialized.\n", uart);
  }
  while ((uart_ptrs[uart][UART_REG_STATUS] &
          UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA) == 0) {
  }
  return uart_ptrs[uart][UART_REG_RECEIVE_FIFO];
}

bool uart_has_data(const int uart) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart_ptrs[uart] == NULL) {
    pynq_error("UART%d has not been initialized.\n", uart);
  }
  return ((uart_ptrs[uart][UART_REG_STATUS] &
           UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA) ==
          UART_REG_STATUS_BIT_RX_FIFO_HAS_DATA);
}

bool uart_has_space(const int uart) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart_ptrs[uart] == NULL) {
    pynq_error("UART%d has not been initialized.\n", uart);
  }
  return ((uart_ptrs[uart][UART_REG_STATUS] &
           UART_REG_STATUS_BIT_TX_FIFO_FULL) == 0);
}

void uart_reset_fifos(const int uart) {
  if (!(uart >= UART0 && uart < NUM_UARTS)) {
    pynq_error("invalid UART %d, must be 0..%d-1\n", uart, NUM_UARTS);
  }
  if (uart_ptrs[uart] == NULL) {
    pynq_error("UART%d has not been initialized.\n", uart);
  }
  uart_ptrs[uart][UART_REG_CONTROL] = UART_REG_CONTROL_BIT_CLEAR_FIFOS;
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

```
*/
#ifndef UART_H
#define UART_H
#include <stdbool.h>
#include <stdint.h>

typedef enum { UART0 = 0, UART1 = 1, NUM_UARTS } uart_index_t;

extern void uart_init(const int uart);

extern void uart_destroy(const int uart);

extern void uart_send(const int uart, const uint8_t data);

extern uint8_t uart_recv(const int uart);

extern bool uart_has_data(const int uart);

extern bool uart_has_space(const int uart);

extern void uart_reset_fifos(const int uart);

#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

Go to the documentation of this file.

```
/******************************************************************************
 *  Copyright (c) 2016, Xilinx, Inc.
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are met:
 *
 *  1.  Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 *
 *  2.  Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 *  3.  Neither the name of the copyright holder nor the names of its
 *      contributors may be used to endorse or promote products derived from
 *      this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
 *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *****************************************************************************/
/******************************************************************************
 *
 *
 * @file uio.c
 *
 * Functions to interact with linux UIO. No safe checks here, so users must
 * know what they are doing.
 *
 * <pre>
 * MODIFICATION HISTORY:
 *
 * Ver   Who  Date     Changes
 * ----- --- -------- -----------------------------------------------
 * 1.00a yrq 12/05/17 Initial release
 *
 * </pre>
 *
 *****************************************************************************/

#include "uio.h"
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <unistd.h>

/******************************************************************************
 * Function to set the UIO device.
 * @param   uio_index is the uio index in /dev list.
 * @param   length is the length of the MMAP in bytes.
 * @return  A pointer pointing to the MMAP of the UIO.
 *****************************************************************************/
void *setUIO(int uio_index, int length) {
  char uio_buf[32];
  int uio_fd;
  void *uio_ptr;

  sprintf(uio_buf, "/dev/uio%d", uio_index);
  uio_fd = open(uio_buf, O_RDWR);
  if (uio_fd < 1) {
    printf("Invalid UIO device file: %s.\n", uio_buf);
  }
  // mmap the UIO devices
  uio_ptr = mmap(NULL, length, PROT_READ | PROT_WRITE, MAP_SHARED, uio_fd, 0);
  return uio_ptr;
}

/******************************************************************************
 * Function to set the UIO device.
 * @param   uio_ptr is the uio pointer to be freed.
 * @param   length is the length of the MMAP.
 * @return  0 on success; -1 otherwise.
 *****************************************************************************/
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.

```
/******************************************************************************
 *  Copyright (c) 2016, Xilinx, Inc.
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are met:
 *
 *  1.  Redistributions of source code must retain the above copyright notice,
 *     this list of conditions and the following disclaimer.
 *
 *  2.  Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *
 *  3.  Neither the name of the copyright holder nor the names of its
 *      contributors may be used to endorse or promote products derived from
 *      this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 *  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 *  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 *  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 *  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 *  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 *  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
 *  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *****************************************************************************/

/*****************************************************************************/
#ifndef __UIO_H__
#define __UIO_H__

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

#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

Go to the documentation of this file.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <libpynq.h>
#include <unistd.h>

typedef struct {
  char *name;
  gpio_direction_t state;
  uint8_t channel;
  char *level;
} pin_state_t;

void sleep_msec(int msec) {
  if (msec > 0)
    usleep(msec * 1000);
}

void mapping_info(void) {
  const char *const dir[2] = {"Input", "Output"};
  printf("Pin\tName\tI/O\tLevel\tChannel\tCh_Name\t\tState\n");
  for (int i = 0; i < IO_NUM_PINS; i++) {
    pin_state_t pin_array = {
        0,
    };
    pin_array.name = pin_names[i];
    pin_array.state = gpio_get_direction(i);
    if (gpio_get_level(i) == GPIO_LEVEL_HIGH) {
      pin_array.level = "high";
    } else if (gpio_get_level(i) == GPIO_LEVEL_LOW) {
      pin_array.level = "low";
    } else {
      pin_array.level = "undef";
    }
    // get the index of the channel the pin is mapped to, 0 for none
    pin_array.channel = switchbox_get_pin(i);

    printf("%i\t%s\t%s\t%s\t%u\t", i, pin_array.name, dir[pin_array.state],
           pin_array.level, pin_array.channel);

    printf("%s\t", switchbox_names[pin_array.channel]);
    if (pin_array.channel != SWB_GPIO && pin_array.state != GPIO_DIR_INPUT) {
      printf("Invalid\n");
    } else {
      printf("Valid\n");
    }
  }
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef UTIL_H
#define UTIL_H

#include <stdlib.h>
#include <switchbox.h>

extern void sleep_msec(int msec);

extern void mapping_info(void);

#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 5
- #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 5
```

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.

```c
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <libpynq.h>

/**********************
 * WARNING
 * only change the numbers in these 4 #defs; do not change anything else
 * the libpynq version in doxygen ryb.doxy is updated automatically based
 * on the next 4 lines
 **********************/
#define LIBPYNQ_RELEASE "5EWC0-2023"
#define LIBPYNQ_VERSION_MAJOR 0
#define LIBPYNQ_VERSION_MINOR 2
#define LIBPYNQ_VERSION_PATCH 5
const version_t libpynq_version = {
    LIBPYNQ_RELEASE,
    LIBPYNQ_VERSION_MAJOR,
    LIBPYNQ_VERSION_MINOR,
    LIBPYNQ_VERSION_PATCH,
};

#undef LOG_DOMAIN
#define LOG_DOMAIN "version"

void print_version(void) {
  arm_shared t;
  version_t volatile *hardwareVersion =
      (version_t volatile *)arm_shared_init(&t, axi_version_0, 4096);
  printf("Bitstream version: %d.%d.%d\r\n", hardwareVersion->major,
         hardwareVersion->minor, hardwareVersion->patch);
  printf("Libpynq release %s version %d.%d.%d\r\n", libpynq_version.release,
         libpynq_version.major, libpynq_version.minor, libpynq_version.patch);
  if (libpynq_version.major != hardwareVersion->major) {
    pynq_error(
        "ERROR: the bitstream (hardware) and the libpynq library versions "
        "are incompatible. Please update your SD-card image and libpynq "
        "library.\n");
  } else if (libpynq_version.minor > hardwareVersion->minor) {
    printf("INFO: the libpynq library is newer than the bitstream (hardware). "
           "Please check if there is a newer version of the SD-card image.\n");
  } else if (libpynq_version.minor < hardwareVersion->minor) {
    printf(
        "INFO: the bitstream (hardware) is newer than the libpynq library. "
        "Please check if there is a newer version of the libpynq library.\n");
  }
  arm_shared_close(&t);
}

void check_version(void) {
  arm_shared t;
  version_t volatile *hardwareVersion =
      (version_t volatile *)arm_shared_init(&t, axi_version_0, 4096);
  if (libpynq_version.major != hardwareVersion->major) {
    print_version();
  }
  arm_shared_close(&t);
}
```

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.

```
/*
Copyright (c) 2023 Eindhoven University of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#ifndef VERSION_H
#define VERSION_H

#include <stdint.h>

typedef struct {
  uint8_t release[64];
  uint32_t major;
  uint32_t minor;
  uint32_t patch;
} version_t;

extern const version_t libpynq_version;

extern void print_version(void);

extern void check_version(void);

#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.

```
/*****************************************************************************
 * Copyright (C) 2002 - 2021 Xilinx, Inc.  All rights reserved.
 * SPDX-License-Identifier: MIT
 ******************************************************************************/

/*****************************************************************************/
#ifndef XIIC_I_H /* prevent circular inclusions */
#define XIIC_I_H /* by using protection macros */

#ifdef __cplusplus
extern "C" {
#endif

/***************************** Include Files *********************************/

#include "xiic.h"
#include "xil_assert.h"
#include "xil_types.h"
#include "xstatus.h"

/************************** Constant Definitions *****************************/

/**************************** Type Definitions *******************************/

/***************** Macros (Inline Functions) Definitions *********************/

/*****************************************************************************
 *
 * This macro sends the first byte of the address for a 10 bit address during
 * both read and write operations. It takes care of the details to format the
 * address correctly.
 *
 * address = 1111_0xxD   xx = address MSBits
 *                        D = Tx direction = 0 = write
 *
 * @param   SlaveAddress contains the address of the slave to send to.
 * @param   Operation indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_Send10BitAddrByte1(u16 SlaveAddress, u8 Operation);
 *
 ******************************************************************************/
#define XIic_Send10BitAddrByte1(SlaveAddress, Operation)                       \
  {                                                                            \
    u8 LocalAddr = (u8)((SlaveAddress) » 7);                                  \
    LocalAddr = (LocalAddr & 0xF6) | 0xF0 | (Operation);                       \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,               \
                  (u32)LocalAddr);                                             \
  }

/******************************************************************************
 *
 * This macro sends the second byte of the address for a 10 bit address during
 * both read and write operations. It takes care of the details to format the
 * address correctly.
 *
 * @param   SlaveAddress contains the address of the slave to send to.
 *
 * @return  None.
 *
 * @note        Signature: void XIic_Send10BitAddrByte2(u16
 *SlaveAddress, u8 Operation);
 *
 ******************************************************************************/
#define XIic_Send10BitAddrByte2(SlaveAddress)                                  \
  XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,                 \
                (u32)(SlaveAddress));

/******************************************************************************
 *
 * 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.
 *
 * @param   SlaveAddress contains the address of the slave to send to.
 * @param   Operation indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_Send7BitAddr(u16 SlaveAddress, u8 Operation);
 *
 ******************************************************************************/
#define XIic_Send7BitAddr(SlaveAddress, Operation)                             \
  {                                                                            \
    u8 LocalAddr = (u8)(SlaveAddress « 1);                                    \
    LocalAddr = (LocalAddr & 0xFE) | (Operation);                              \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,               \
                  (u32)LocalAddr);                                             \
  }

/******************************************************************************
 *
 * This macro disables the specified interrupts in the Interrupt enable
 * register.  It is non-destructive in that the register is read and only the
 * interrupts specified is changed.
 *
 * @param   BaseAddress is the base address of the IIC device.
 * @param   InterruptMask contains the interrupts to be disabled
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_DisableIntr(u32 BaseAddress, u32 InterruptMask);
 *
 ******************************************************************************/
#define XIic_DisableIntr(BaseAddress, InterruptMask)                           \
  XIic_WriteIier((BaseAddress), XIic_ReadIier(BaseAddress) & ~(InterruptMask))

/******************************************************************************
 *
 * This macro enables the specified interrupts in the Interrupt enable
 * register.  It is non-destructive in that the register is read and only the
 * interrupts specified is changed.
 *
 * @param   BaseAddress is the base address of the IIC device.
 * @param   InterruptMask contains the interrupts to be disabled
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_EnableIntr(u32 BaseAddress, u32 InterruptMask);
 *
 ******************************************************************************/
#define XIic_EnableIntr(BaseAddress, InterruptMask)                            \
  XIic_WriteIier((BaseAddress), XIic_ReadIier(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.
 *
 * @param   BaseAddress is the base address of the IIC device.
 * @param   InterruptMask contains the interrupts to be disabled
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_ClearIntr(u32 BaseAddress, u32 InterruptMask);
 *
 ******************************************************************************/
#define XIic_ClearIntr(BaseAddress, InterruptMask)                          \
  XIic_WriteIisr((BaseAddress), XIic_ReadIisr(BaseAddress) & (InterruptMask))

/*****************************************************************************
 *
 * This macro clears and enables the specified interrupt in the Interrupt
 * status and enable registers.  It is non-destructive in that the registers are
 * read and only the interrupt specified is modified.
 * Clearing an interrupt acknowledges it.
 *
 * @param   BaseAddress is the base address of the IIC device.
 * @param   InterruptMask contains the interrupts to be cleared and enabled
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_ClearEnableIntr(u32 BaseAddress, u32 InterruptMask);
 *
 ******************************************************************************/
#define XIic_ClearEnableIntr(BaseAddress, InterruptMask)                    \
  {                                                                         \
    XIic_WriteIisr(BaseAddress,                                             \
                   (XIic_ReadIisr(BaseAddress) & (InterruptMask)));         \
                                                                            \
    XIic_WriteIier(BaseAddress,                                             \
                   (XIic_ReadIier(BaseAddress) | (InterruptMask)));         \
  }

/*****************************************************************************
 *
 * This macro flushes the receive FIFO such that all bytes contained within it
 * are discarded.
 *
 * @param   InstancePtr is a pointer to the IIC instance containing the FIFO
 *      to be flushed.
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_FlushRxFifo(XIic *InstancePtr);
 *
 ******************************************************************************/
#define XIic_FlushRxFifo(InstancePtr)                                       \
  {                                                                         \
    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);          \
    }                                                                       \
  }

/*****************************************************************************
 *
 * This macro flushes the transmit FIFO such that all bytes contained within it
 * are discarded.
 *
 * @param   InstancePtr is a pointer to the IIC instance containing the FIFO
 *      to be flushed.
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_FlushTxFifo(XIic *InstancePtr);
 *
 ******************************************************************************/
#define XIic_FlushTxFifo(InstancePtr)                                       \
  ;                                                                         \
  {                                                                         \
    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);      \
  }

/******************************************************************************
 *
 * This macro reads the next available received byte from the receive FIFO
 * and updates all the data structures to reflect it.
 *
 * @param   InstancePtr is a pointer to the IIC instance to be operated on.
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_ReadRecvByte(XIic *InstancePtr);
 *
 ******************************************************************************/
#define XIic_ReadRecvByte(InstancePtr)                                          \
  {                                                                             \
    *InstancePtr->RecvBufferPtr++ =                                             \
        XIic_ReadReg(InstancePtr->BaseAddress, XIIC_DRR_REG_OFFSET);            \
    InstancePtr->RecvByteCount--;                                               \
    InstancePtr->Stats.RecvBytes++;                                             \
  }

/******************************************************************************
 *
 * This macro writes the next byte to be sent to the transmit FIFO
 * and updates all the data structures to reflect it.
 *
 * @param   InstancePtr is a pointer to the IIC instance to be operated on.
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_WriteSendByte(XIic *InstancePtr);
 *
 ******************************************************************************/
#define XIic_WriteSendByte(InstancePtr)                                         \
  {                                                                             \
    XIic_WriteReg(InstancePtr->BaseAddress, XIIC_DTR_REG_OFFSET,                \
                  *InstancePtr->SendBufferPtr++);                               \
    InstancePtr->SendByteCount--;                                               \
    InstancePtr->Stats.SendBytes++;                                             \
  }

/******************************************************************************
 *
 * This macro sets up the control register for a master receive operation.
 * A write is necessary if a 10 bit operation is being performed.
 *
 * @param   InstancePtr is a pointer to the IIC instance to be operated on.
 * @param   ControlRegister contains the contents of the IIC device control
 *      register
 * @param   ByteCount contains the number of bytes to be received for the
 *      master receive operation
 *
 * @return  None.
 *
 * @note        Signature:
 *      void XIic_SetControlRegister(XIic *InstancePtr,
 *                      u8 ControlRegister,
 *                      int ByteCount);
 *
 ******************************************************************************/
#define XIic_SetControlRegister(InstancePtr, ControlRegister, ByteCount)        \
  {                                                                             \
    (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;                               \
      }                                                                         \
    }                                                                           \
  }

/************************** Function Prototypes ******************************/

extern XIic_Config XIic_ConfigTable[];

/* The following variables are shared across files of the driver and
 * are function pointers that are necessary to break dependencies allowing
 * optional parts of the driver to be used without condition compilation
 */
extern void (*XIic_AddrAsSlaveFuncPtr)(XIic *InstancePtr);
extern void (*XIic_NotAddrAsSlaveFuncPtr)(XIic *InstancePtr);
extern void (*XIic_RecvSlaveFuncPtr)(XIic *InstancePtr);
extern void (*XIic_SendSlaveFuncPtr)(XIic *InstancePtr);
extern void (*XIic_RecvMasterFuncPtr)(XIic *InstancePtr);
extern void (*XIic_SendMasterFuncPtr)(XIic *InstancePtr);
extern void (*XIic_ArbLostFuncPtr)(XIic *InstancePtr);
extern void (*XIic_BusNotBusyFuncPtr)(XIic *InstancePtr);

void XIic_TransmitFifoFill(XIic *InstancePtr, int Role);

#ifdef __cplusplus
}
#endif

#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.

```
/******************************************************************************
 * Copyright (C) 2002 - 2021 Xilinx, Inc.  All rights reserved.
 * SPDX-License-Identifier: MIT
 ******************************************************************************/

/*****************************************************************************/
/***************************** Include Files *********************************/
#define _DEFAULT_SOURCE
#include <stdio.h>
#include <time.h>
#include <unistd.h>

#define IIC_TIMEOUT 5

#include "xiic_l.h"
#include "xil_types.h"
```

```

/************************** Constant Definitions ***************************/

/**************************** Type Definitions *****************************/

/***************** Macros (Inline Functions) Definitions *******************/

/************************** Function Prototypes ****************************/

static unsigned RecvData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
                         u8 Option);
static unsigned SendData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
                         u8 Option);

/************************** Variable Definitions *************************/

/*****************************************************************************/
unsigned XIic_Recv(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
                   unsigned ByteCount, u8 Option) {
  u32 CntlReg;
  unsigned RemainingByteCount;
  volatile u32 StatusReg;

  /* Tx error is enabled in case the address (7 or 10) has no device to
   * answer with Ack. When only one byte of data, must set NO ACK before
   * address goes out therefore Tx error must not be enabled as it will go
   * off immediately and the Rx full interrupt will be checked.  If full,
   * then the one byte was received and the Tx error will be disabled
   * without sending an error callback msg
   */
  XIic_ClearIisr(BaseAddress, XIIC_INTR_RX_FULL_MASK | XIIC_INTR_TX_ERROR_MASK |
                                  XIIC_INTR_ARB_LOST_MASK);

  /* Set receive FIFO occupancy depth for 1 byte (zero based) */
  XIic_WriteReg(BaseAddress, XIIC_RFD_REG_OFFSET, 0);

  /* Check to see if already Master on the Bus.
   * If Repeated Start bit is not set send Start bit by setting MSMS bit
   * else Send the address
   */
  CntlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
  if ((CntlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
    /* 7 bit slave address, send the address for a read operation
     * and set the state to indicate the address has been sent
     */
    XIic_Send7BitAddress(BaseAddress, Address, XIIC_READ_OPERATION);

    /* MSMS gets set after putting data in FIFO. Start the master
     * receive operation by setting CR Bits MSMS to Master, if the
     * buffer is only one byte, then it should not be acknowledged
     * to indicate the end of data
     */
    CntlReg = XIIC_CR_MSMS_MASK | XIIC_CR_ENABLE_DEVICE_MASK;
    if (ByteCount == 1) {
      CntlReg |= XIIC_CR_NO_ACK_MASK;
    }

    /* Write out the control register to start receiving data and
     * call the function to receive each byte into the buffer
     */
    XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, CntlReg);

    /* Clear the latched interrupt status for the bus not busy bit
     * which must be done while the bus is busy
     */
    StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);

    while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) == 0) {
      StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
    }

    XIic_ClearIisr(BaseAddress, XIIC_INTR_BNB_MASK);
  } else {
    /* Before writing 7bit slave address the Direction of Tx bit
     * must be disabled
     */
    CntlReg &= ~XIIC_CR_DIR_IS_TX_MASK;
    if (ByteCount == 1) {
      CntlReg |= XIIC_CR_NO_ACK_MASK;
    }
    XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, CntlReg);
    /* Already owns the Bus indicating that its a Repeated Start
     * call. 7 bit slave address, send the address for a read
     * operation and set the state to indicate the address has been
     * sent
     */
    XIic_Send7BitAddress(BaseAddress, Address, XIIC_READ_OPERATION);
  }
  /* Try to receive the data from the IIC bus */

  RemainingByteCount = RecvData(BaseAddress, BufferPtr, ByteCount, Option);

  CntlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
  if ((CntlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
    /* The receive is complete, disable the IIC device if the Option
     * is to release the Bus after Reception of data and return the
     * number of bytes that was received
     */
    XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, 0);
  }

  /* Wait until I2C bus is freed, exit if timed out. */
  if (XIic_WaitBusFree(BaseAddress) != XST_SUCCESS) {
    return 0;
  }

  /* Return the number of bytes that was received */
  return ByteCount - RemainingByteCount;
}

/******************************************************************************
 *
 * Receive the specified data from the device that has been previously addressed
 * on the IIC bus.  This function assumes that the 7 bit address has been sent
 * and it should wait for the transmit of the address to complete.
 *
 * @param   BaseAddress contains the base address of the IIC device.
 * @param   BufferPtr points to the buffer to hold the data that is
 *      received.
 * @param   ByteCount is the number of bytes to be received.
 * @param   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.
 *
 * @return  The number of bytes remaining to be received.
 *
 * @note
 *
 * This function does not take advantage of the receive FIFO because it is
 * designed for minimal code space and complexity.  It contains loops that
 * that could cause the function not to return if the hardware is not working.
 *
 * This function assumes that the calling function will disable the IIC device
 * after this function returns.
 *
 ******************************************************************************/
static unsigned RecvData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
                         u8 Option) {
  u32 CntlReg;
  u32 IntrStatusMask;
  u32 IntrStatus;

  /* Attempt to receive the specified number of bytes on the IIC bus */

  while (ByteCount > 0) {
    /* Setup the mask to use for checking errors because when
     * receiving one byte OR the last byte of a multibyte message an
     * error naturally occurs when the no ack is done to tell the
     * slave the last byte
     */
    if (ByteCount == 1) {
      IntrStatusMask = XIIC_INTR_ARB_LOST_MASK | XIIC_INTR_BNB_MASK;
    } else {
      IntrStatusMask = XIIC_INTR_ARB_LOST_MASK | XIIC_INTR_TX_ERROR_MASK |
                       XIIC_INTR_BNB_MASK;
    }

    /* Wait for the previous transmit and the 1st receive to
     * complete by checking the interrupt status register of the
     * IPIF
     */
    while (1) {
      IntrStatus = XIic_ReadIisr(BaseAddress);
      if (IntrStatus & XIIC_INTR_RX_FULL_MASK) {
        break;
      }
      /* Check the transmit error after the receive full
       * because when sending only one byte transmit error
       * will occur because of the no ack to indicate the end
       * of the data
       */
      if (IntrStatus & IntrStatusMask) {
        return ByteCount;
      }
    }

    CntlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);

    /* Special conditions exist for the last two bytes so check for
     * them. Note that the control register must be setup for these
     * conditions before the data byte which was already received is
     * read from the receive FIFO (while the bus is throttled
     */
    if (ByteCount == 1) {
      if (Option == XIIC_STOP) {

        /* If the Option is to release the bus after the
         * last data byte, it has already been read and
         * no ack has been done, so clear MSMS while
         * leaving the device enabled so it can get off
         * the IIC bus appropriately with a stop
         */
        XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                      XIIC_CR_ENABLE_DEVICE_MASK);
      }
    }

    /* Before the last byte is received, set NOACK to tell the slave
     * IIC device that it is the end, this must be done before
     * reading the byte from the FIFO
     */
    if (ByteCount == 2) {
      /* Write control reg with NO ACK allowing last byte to
       * have the No ack set to indicate to slave last byte
       * read
       */
      XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                    CntlReg | XIIC_CR_NO_ACK_MASK);
    }

    /* Read in data from the FIFO and unthrottle the bus such that
     * the next byte is read from the IIC bus
     */
    *BufferPtr++ = (u8)XIic_ReadReg(BaseAddress, XIIC_DRR_REG_OFFSET);

    if ((ByteCount == 1) && (Option == XIIC_REPEATED_START)) {

      /* RSTA bit should be set only when the FIFO is
       * completely Empty.
       */
      XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                    XIIC_CR_ENABLE_DEVICE_MASK | XIIC_CR_MSMS_MASK |
                        XIIC_CR_REPEATED_START_MASK);
    }

    /* Clear the latched interrupt status so that it will be updated
     * with the new state when it changes, this must be done after
     * the receive register is read
     */
    XIic_ClearIisr(BaseAddress, XIIC_INTR_RX_FULL_MASK |
                                    XIIC_INTR_TX_ERROR_MASK |
                                    XIIC_INTR_ARB_LOST_MASK);
    ByteCount--;
  }

  if (Option == XIIC_STOP) {

    /* If the Option is to release the bus after Reception of data,
     * wait for the bus to transition to not busy before returning,
     * the IIC device cannot be disabled until this occurs. It
     * should transition as the MSMS bit of the control register was
     * cleared before the last byte was read from the FIFO
     */
    while (1) {
      if (XIic_ReadIisr(BaseAddress) & XIIC_INTR_BNB_MASK) {
        break;
      }
    }
  }

  return ByteCount;
}

/******************************************************************************/
unsigned XIic_Send(UINTPTR BaseAddress, u8 Address, u8 *BufferPtr,
                   unsigned ByteCount, u8 Option) {
  unsigned RemainingByteCount;
  u32 ControlReg;
  volatile u32 StatusReg;

  /* Wait until I2C bus is freed, exit if timed out. */
```

```
    if (XIic_WaitBusFree(BaseAddress) != XST_SUCCESS) {
      return 0;
    }

    /* Check to see if already Master on the Bus.
     * If Repeated Start bit is not set send Start bit by setting
     * MSMS bit else Send the address.
     */
    ControlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
    if ((ControlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
      /*
       * Put the address into the FIFO to be sent and indicate
       * that the operation to be performed on the bus is a
       * write operation
       */
      XIic_Send7BitAddress(BaseAddress, Address, XIIC_WRITE_OPERATION);
      /* Clear the latched interrupt status so that it will
       * be updated with the new state when it changes, this
       * must be done after the address is put in the FIFO
       */
      XIic_ClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK |
                                  XIIC_INTR_TX_ERROR_MASK |
                                  XIIC_INTR_ARB_LOST_MASK);

      /*
       * MSMS must be set after putting data into transmit FIFO,
       * indicate the direction is transmit, this device is master
       * and enable the IIC device
       */
      XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                    XIIC_CR_MSMS_MASK | XIIC_CR_DIR_IS_TX_MASK |
                        XIIC_CR_ENABLE_DEVICE_MASK);

      /*
       * Clear the latched interrupt
       * status for the bus not busy bit which must be done while
       * the bus is busy
       */
      time_t s = time(NULL);
      StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
      while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) == 0) {
        StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
        time_t n = time(NULL);
        if ((n - s) > IIC_TIMEOUT) {
          printf("IIC timeout bus not busy.\n");
          return 0;
        }
      }

      XIic_ClearIisr(BaseAddress, XIIC_INTR_BNB_MASK);
    } else {
      /*
       * Already owns the Bus indicating that its a Repeated Start
       * call. 7 bit slave address, send the address for a write
       * operation and set the state to indicate the address has
       * been sent.
       */
      XIic_Send7BitAddress(BaseAddress, Address, XIIC_WRITE_OPERATION);
    }

    /* Send the specified data to the device on the IIC bus specified by the
     * the address
     */
    RemainingByteCount = SendData(BaseAddress, BufferPtr, ByteCount, Option);

    ControlReg = XIic_ReadReg(BaseAddress, XIIC_CR_REG_OFFSET);
    if ((ControlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
      /*
       * The Transmission is completed, disable the IIC device if
       * the Option is to release the Bus after transmission of data
       * and return the number of bytes that was received. Only wait
       * if master, if addressed as slave just reset to release
       * the bus.
       */
      if ((ControlReg & XIIC_CR_MSMS_MASK) != 0) {
        XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                      (ControlReg & ~XIIC_CR_MSMS_MASK));
      }

      if ((XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET) &
           XIIC_SR_ADDR_AS_SLAVE_MASK) != 0) {
        XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET, 0);
      } else {
        StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
        while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) != 0) {
          StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
        }
      }
    }

    return ByteCount - RemainingByteCount;
  }

  /****************************************************************************
   *
   * Send the specified buffer to the device that has been previously addressed
   * on the IIC bus.  This function assumes that the 7 bit address has been sent
   * and it should wait for the transmit of the address to complete.
   *
   * @param   BaseAddress contains the base address of the IIC device.
   * @param   BufferPtr points to the data to be sent.
   * @param   ByteCount is the number of bytes to be sent.
   * @param   Option indicates whether to hold or free the bus after
   *      transmitting the data.
   *
   * @return  The number of bytes remaining to be sent.
   *
   * @note
   *
   * This function does not take advantage of the transmit FIFO because it is
   * designed for minimal code space and complexity.  It contains loops that
   * that could cause the function not to return if the hardware is not working.
   *
   ******************************************************************************/
  static unsigned SendData(UINTPTR BaseAddress, u8 *BufferPtr, unsigned ByteCount,
                           u8 Option) {
    u32 IntrStatus;

    /*
     * Send the specified number of bytes in the specified buffer by polling
     * the device registers and blocking until complete
     */
    while (ByteCount > 0) {
      /*
       * Wait for the transmit to be empty before sending any more
       * data by polling the interrupt status register
       */
      while (1) {
        IntrStatus = XIic_ReadIisr(BaseAddress);

        if (IntrStatus & (XIIC_INTR_TX_ERROR_MASK | XIIC_INTR_ARB_LOST_MASK |
                          XIIC_INTR_BNB_MASK)) {
          return ByteCount;
        }

        if (IntrStatus & XIIC_INTR_TX_EMPTY_MASK) {
          break;
        }
      }
      /* If there is more than one byte to send then put the
       * next byte to send into the transmit FIFO
       */
      if (ByteCount > 1) {
        XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, *BufferPtr++);
      } else {
        if (Option == XIIC_STOP) {
          /*
           * If the Option is to release the bus after
           * the last data byte, Set the stop Option
           * before sending the last byte of data so
           * that the stop Option will be generated
           * immediately following the data. This is
           * done by clearing the MSMS bit in the
           * control register.
           */
          XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                        XIIC_CR_ENABLE_DEVICE_MASK | XIIC_CR_DIR_IS_TX_MASK);
        }

        /*
         * Put the last byte to send in the transmit FIFO
         */
        XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, *BufferPtr++);

        if (Option == XIIC_REPEATED_START) {
          XIic_ClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK);
          /*
           * Wait for the transmit to be empty before
           * setting RSTA bit.
           */
          while (1) {
            IntrStatus = XIic_ReadIisr(BaseAddress);
            if (IntrStatus & XIIC_INTR_TX_EMPTY_MASK) {
              /*
            * RSTA bit should be set only
            * when the FIFO is completely
            * Empty.
            */
           XIic_WriteReg(BaseAddress, XIIC_CR_REG_OFFSET,
                         XIIC_CR_REPEATED_START_MASK |
                             XIIC_CR_ENABLE_DEVICE_MASK |
                             XIIC_CR_DIR_IS_TX_MASK | XIIC_CR_MSMS_MASK);
           break;
         }
       }
     }
   }

   /*
    * Clear the latched interrupt status register and this must be
    * done after the transmit FIFO has been written to or it won't
    * clear
    */
   XIic_ClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK);

   /*
    * Update the byte count to reflect the byte sent and clear
    * the latched interrupt status so it will be updated for the
    * new state
    */
   ByteCount--;
  }

  if (Option == XIIC_STOP) {
   /*
    * If the Option is to release the bus after transmission of
    * data, Wait for the bus to transition to not busy before
    * returning, the IIC device cannot be disabled until this
    * occurs. Note that this is different from a receive operation
    * because the stop Option causes the bus to go not busy.
    */
   while (1) {
     if (XIic_ReadIisr(BaseAddress) & XIIC_INTR_BNB_MASK) {
       break;
     }
   }
  }

  return ByteCount;
}

/*****************************************************************************
 *
 * This is a function which tells whether the I2C bus is busy or free.
 *
 * @param   BaseAddr is the base address of the I2C core to work on.
 *
 * @return
 *      - TRUE if the bus is busy.
 *      - FALSE if the bus is NOT busy.
 *
 * @note        None.
 *
 ******************************************************************************/
u32 XIic_CheckIsBusBusy(UINTPTR BaseAddress) {
  u32 StatusReg;

  StatusReg = XIic_ReadReg(BaseAddress, XIIC_SR_REG_OFFSET);
  if (StatusReg & XIIC_SR_BUS_BUSY_MASK) {
    return TRUE;
  } else {
    return FALSE;
  }
}

/*****************************************************************************/
u32 XIic_WaitBusFree(UINTPTR BaseAddress) {
  u32 BusyCount = 0;

  while (XIic_CheckIsBusBusy(BaseAddress)) {
    if (BusyCount++ > 10000) {
      return XST_FAILURE;
    }
    usleep(100);
  }

  return XST_SUCCESS;
}
```

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 XIic_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 XIic_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.

```
/******************************************************************************
 * Copyright (C) 2002 - 2021 Xilinx, Inc.  All rights reserved.
 * SPDX-License-Identifier: MIT
 ******************************************************************************/

/****************************************************************************/
#ifndef XIIC_L_H /* prevent circular inclusions */
#define XIIC_L_H /* by using protection macros */

#ifdef __cplusplus
extern "C" {
#endif

/***************************** Include Files *******************************/

#include "xil_io.h"
#include "xil_types.h"

/************************** Constant Definitions ***************************/

#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
/* @} */

#define XIIC_GINTR_ENABLE_MASK 0x80000000
/* @} */
```

```

#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)

/* @} */

#define XIIC_RESET_MASK 0x0000000A
/* @} */

#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
/* @} */

#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
/* @} */

#define XIIC_TX_DYN_START_MASK 0x00000100
#define XIIC_TX_DYN_STOP_MASK 0x00000200
#define IIC_TX_FIFO_DEPTH 16
/* @} */

#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
/***************** Macros (Inline Functions) Definitions *********************/

#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(BaseAddress) & (InterruptMask))

/***************************************************************************/
#define XIic_Send7BitAddress(BaseAddress, SlaveAddress, Operation)             \
  {                                                                            \
    u8 LocalAddr = (u8)(SlaveAddress « 1);                                    \
    LocalAddr = (LocalAddr & 0xFE) | (Operation);                              \
    XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET, LocalAddr);                \
  }

/***************************************************************************/
#define XIic_DynSend7BitAddress(BaseAddress, SlaveAddress, Operation)          \
  {                                                                            \
    u8 LocalAddr = (u8)(SlaveAddress « 1);                                    \
    LocalAddr = (LocalAddr & 0xFE) | (Operation);                              \
    XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                            \
                  XIIC_TX_DYN_START_MASK | LocalAddr);                         \
  }

/***************************************************************************/
#define XIic_DynSendStartStopAddress(BaseAddress, SlaveAddress, Operation)     \
  {                                                                            \
    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); \
  }

/***************************************************************************/
#define XIic_DynSendStop(BaseAddress, ByteCount)                               \
  {                                                                            \
    XIic_WriteReg(BaseAddress, XIIC_DTR_REG_OFFSET,                            \
                  XIIC_TX_DYN_STOP_MASK | ByteCount);                          \
  }

/************************** Function Prototypes ****************************/

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);

#ifdef __cplusplus
}
#endif

#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.

```
/******************************************************************************
 * Copyright (c) 2014 - 2021 Xilinx, Inc.  All rights reserved.
 * SPDX-License-Identifier: MIT
 ******************************************************************************/

/*****************************************************************************/
#ifndef XIL_IO_H /* prevent circular inclusions */
#define XIL_IO_H /* by using protection macros */

#ifdef __cplusplus
extern "C" {
#endif

/***************************** Include Files *********************************/

#include "xil_types.h"

/************************** Function Prototypes ******************************/
#ifdef ENABLE_SAFETY
extern u32 XStl_RegUpdate(u32 RegAddr, u32 RegVal);
#endif

/***************** Macros (Inline Functions) Definitions *********************/
#if defined __GNUC__
#if defined(__MICROBLAZE__)
#define INST_SYNC mbar(0)
#define DATA_SYNC mbar(1)
#else
#define SYNCHRONIZE_IO dmb()
#define INST_SYNC isb()
#define DATA_SYNC dsb()
#endif
#else
#define SYNCHRONIZE_IO
#define INST_SYNC
```

```
#define DATA_SYNC
#define INST_SYNC
#define DATA_SYNC
#endif

#if defined(__GNUC__) || defined(__ICCARM__) || defined(__MICROBLAZE__)
#define INLINE inline
#else
#define INLINE __inline
#endif

/*****************************************************************************/
static INLINE u8 Xil_In8(UINTPTR Addr) { return *(volatile u8 *)Addr; }

/*****************************************************************************/
static INLINE u16 Xil_In16(UINTPTR Addr) { return *(volatile u16 *)Addr; }

/*****************************************************************************/
static INLINE u32 Xil_In32(UINTPTR Addr) { return *(volatile u32 *)Addr; }

/*****************************************************************************/
static INLINE u64 Xil_In64(UINTPTR Addr) { return *(volatile u64 *)Addr; }

/*****************************************************************************/
static INLINE void Xil_Out8(UINTPTR Addr, u8 Value) {
  /* write 8 bit value to specified address */
  volatile u8 *LocalAddr = (volatile u8 *)Addr;
  *LocalAddr = Value;
}

/*****************************************************************************/
static INLINE void Xil_Out16(UINTPTR Addr, u16 Value) {
  /* write 16 bit value to specified address */
  volatile u16 *LocalAddr = (volatile u16 *)Addr;
  *LocalAddr = Value;
}

/*****************************************************************************/
static INLINE void Xil_Out32(UINTPTR Addr, u32 Value) {
  /* write 32 bit value to specified address */
#ifndef ENABLE_SAFETY
  volatile u32 *LocalAddr = (volatile u32 *)Addr;
  *LocalAddr = Value;
#else
  XStl_RegUpdate(Addr, Value);
#endif
}

/*****************************************************************************/
static INLINE void Xil_Out64(UINTPTR Addr, u64 Value) {
  /* write 64 bit value to specified address */
  volatile u64 *LocalAddr = (volatile u64 *)Addr;
  *LocalAddr = Value;
}

/*****************************************************************************/
static INLINE int Xil_SecureOut32(UINTPTR Addr, u32 Value) {
  int Status = XST_FAILURE;
  u32 ReadReg;
  u32 ReadRegTemp;

  /* writing 32 bit value to specified address */
  Xil_Out32(Addr, Value);

  /* verify value written to specified address with multiple reads */
  ReadReg = Xil_In32(Addr);
  ReadRegTemp = Xil_In32(Addr);

  if ((ReadReg == Value) && (ReadRegTemp == Value)) {
    Status = XST_SUCCESS;
  }

  return Status;
}

/*****************************************************************************/
static INLINE __attribute__((always_inline)) u16 Xil_EndianSwap16(u16 Data) {
  return (u16)(((Data & 0xFF00U) >> 8U) | ((Data & 0x00FFU) << 8U));
}

/*****************************************************************************/
static INLINE __attribute__((always_inline)) u32 Xil_EndianSwap32(u32 Data) {
  u16 LoWord;
  u16 HiWord;

  /* get each of the half words from the 32 bit word */

  LoWord = (u16)(Data & 0x0000FFFFU);
  HiWord = (u16)((Data & 0xFFFF0000U) » 16U);

  /* byte swap each of the 16 bit half words */

  LoWord = (u16)(((LoWord & 0xFF00U) » 8U) | ((LoWord & 0x00FFU) « 8U));
  HiWord = (u16)(((HiWord & 0xFF00U) » 8U) | ((HiWord & 0x00FFU) « 8U));

  /* swap the half words before returning the value */

  return (((u32)LoWord) « (u32)16U) | (u32)HiWord);
}

#if defined(__MICROBLAZE__)
#ifdef __LITTLE_ENDIAN__
#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
#else
#define Xil_In16BE Xil_In16
#define Xil_In32BE Xil_In32
#define Xil_Out16BE Xil_Out16
#define Xil_Out32BE Xil_Out32
#define Xil_Htons(Data) (Data)
#define Xil_Htonl(Data) (Data)
#define Xil_Ntohs(Data) (Data)
#define Xil_Ntohl(Data) (Data)
#endif
#else
#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
#endif

#if defined(__MICROBLAZE__)
#ifdef __LITTLE_ENDIAN__
static INLINE u16 Xil_In16BE(UINTPTR Addr)
#else
static INLINE u16 Xil_In16LE(UINTPTR Addr)
#endif
#else
static INLINE u16 Xil_In16BE(UINTPTR Addr)
#endif
{
  u16 value = Xil_In16(Addr);
  return Xil_EndianSwap16(value);
}

#if defined(__MICROBLAZE__)
#ifdef __LITTLE_ENDIAN__
static INLINE u32 Xil_In32BE(UINTPTR Addr)
#else
static INLINE u32 Xil_In32LE(UINTPTR Addr)
#endif
#else
static INLINE u32 Xil_In32BE(UINTPTR Addr)
#endif
{
  u32 value = Xil_In32(Addr);
  return Xil_EndianSwap32(value);
}

#if defined(__MICROBLAZE__)
#ifdef __LITTLE_ENDIAN__
static INLINE void Xil_Out16BE(UINTPTR Addr, u16 Value)
#else
static INLINE void Xil_Out16LE(UINTPTR Addr, u16 Value)
#endif
#else
static INLINE void Xil_Out16BE(UINTPTR Addr, u16 Value)
#endif
{
  Value = Xil_EndianSwap16(Value);
  Xil_Out16(Addr, Value);
}

#if defined(__MICROBLAZE__)
```

```c
#ifdef __LITTLE_ENDIAN__
static INLINE void Xil_Out32BE(UINTPTR Addr, u32 Value)
#else
static INLINE void Xil_Out32LE(UINTPTR Addr, u32 Value)
#endif
#else
static INLINE void Xil_Out32BE(UINTPTR Addr, u32 Value)
#endif
{
  Value = Xil_EndianSwap32(Value);
  Xil_Out32(Addr, Value);
}

#ifdef __cplusplus
}
#endif

#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.

```c
/******************************************************************************
 * Copyright (c) 2010 - 2021 Xilinx, Inc.  All rights reserved.
 * Copyright (c) 2022 Advanced Micro Devices, Inc. All Rights Reserved.
 * SPDX-License-Identifier: MIT
 ******************************************************************************/

/*****************************************************************************/
#ifndef XIL_TYPES_H /* prevent circular inclusions */
#define XIL_TYPES_H /* by using protection macros */

#ifdef __cplusplus
extern "C" {
#endif

#include <stddef.h>
#include <stdint.h>

/************************** Constant Definitions *****************************/

#define XST_SUCCESS 0L
#define XST_FAILURE 1L
#ifndef TRUE
#define TRUE 1U
#endif

#ifndef FALSE
#define FALSE 0U
#endif

#ifndef NULL
#define NULL 0U
#endif

#define XIL_COMPONENT_IS_READY                                                \
  0x11111111U
#define XIL_COMPONENT_IS_STARTED                                              \
  0x22222222U
/* @name New types
 * New simple types.
 * @{
 */
#ifndef __KERNEL__
#ifndef XBASIC_TYPES_H
/*
 * guarded against xbasic_types.h.
 */
typedef uint8_t u8;
typedef uint16_t u16;
typedef uint32_t u32;
#define __XUINT64__
typedef struct {
  u32 Upper;
  u32 Lower;
} Xuint64;

/*****************************************************************************/
#define XUINT64_MSW(x) ((x).Upper)

/*****************************************************************************/
#define XUINT64_LSW(x) ((x).Lower)

#endif /* XBASIC_TYPES_H */

/*
 * xbasic_types.h does not typedef s* or u64
 */
typedef char char8;
typedef int8_t s8;
typedef int16_t s16;
typedef int32_t s32;
typedef int64_t s64;
typedef uint64_t u64;
typedef int sint32;

#if defined(__MICROBLAZE__) && !defined(__arch64__) &&                          \
    (XPAR_MICROBLAZE_ADDR_SIZE > 32)
typedef uint64_t UINTPTR;
typedef int64_t INTPTR;
#else
typedef uintptr_t UINTPTR;
typedef intptr_t INTPTR;
#endif

typedef ptrdiff_t PTRDIFF;
#if !defined(LONG) || !defined(ULONG)
typedef long LONG;
typedef unsigned long ULONG;
#endif

#define ULONG64_HI_MASK 0xFFFFFFFF00000000U
#define ULONG64_LO_MASK ~ULONG64_HI_MASK

#else
#include <linux/types.h>
#endif

typedef void (*XInterruptHandler)(void *InstancePtr);

typedef void (*XExceptionHandler)(void *InstancePtr);

#if defined(__aarch64__) || defined(__arch64__)
#define UPPER_32_BITS(n) ((u32)(((n) » 16) » 16))
#else
#define UPPER_32_BITS(n) 0U
#endif
#define LOWER_32_BITS(n) ((u32)(n))

#if defined(__aarch64__) || defined(__arch64__)
#define LEFT_SHIFT_BY_32_BITS(n) (u64)(((u64)n) « 32)
#else
#define LEFT_SHIFT_BY_32_BITS(n) 0U
#endif

/************************** Constant Definitions *****************************/

#ifndef TRUE
#define TRUE 1U
#endif

#ifndef FALSE
#define FALSE 0U
#endif

#ifndef NULL
#define NULL 0U
#endif

#ifdef __cplusplus
}
#endif

#endif /* end of protection macro */
```

Index