

ECG-HRM

Generated by Doxygen 1.9.7

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	2
2.1 File List	2
3 Data Structure Documentation	3
3.1 FIFO_buffer_t Struct Reference	3
3.1.1 Detailed Description	3
3.1.2 Field Documentation	4
4 File Documentation	4
4.1 Debug.c File Reference	4
4.2 Debug.h File Reference	5
4.2.1 Detailed Description	5
4.3 Filter.c File Reference	6
4.4 Filter.h File Reference	6
4.4.1 Detailed Description	7
4.5 QRS.c File Reference	7
4.6 QRS.h File Reference	8
4.6.1 Detailed Description	9
4.7 UserCtrl.c File Reference	10
4.8 UserCtrl.h File Reference	10
4.8.1 Detailed Description	11
4.8.2 Function Documentation	11
4.9 fifo_buff.c File Reference	11
4.9.1 Detailed Description	12
4.9.2 Function Documentation	12
4.10 fifo_buff.h File Reference	14
4.10.1 Detailed Description	15
4.10.2 Function Documentation	15
4.11 ADC.c File Reference	17
4.11.1 Detailed Description	17
4.12 ADC.h File Reference	18
4.12.1 Detailed Description	18
4.13 GPIO.c File Reference	19
4.13.1 Function Documentation	19
4.14 GPIO.h File Reference	21
4.14.1 Detailed Description	22
4.14.2 Function Documentation	22
4.15 ILI9341.c File Reference	24
4.16 ILI9341.h File Reference	25
4.16.1 Detailed Description	26

4.17 isr.c File Reference	26
4.17.1 Detailed Description	27
4.18 PLL.c File Reference	27
4.18.1 Detailed Description	27
4.18.2 Function Documentation	28
4.19 PLL.h File Reference	28
4.19.1 Detailed Description	29
4.19.2 Function Documentation	29
4.20 SPI.c File Reference	29
4.21 SPI.h File Reference	29
4.21.1 Detailed Description	30
4.22 SysTick.c File Reference	30
4.22.1 Detailed Description	31
4.22.2 Function Documentation	31
4.23 SysTick.h File Reference	32
4.23.1 Detailed Description	33
4.23.2 Function Documentation	33
4.24 Timer.c File Reference	34
4.24.1 Function Documentation	34
4.25 Timer.h File Reference	36
4.25.1 Detailed Description	37
4.25.2 Function Documentation	37
4.26 UART.c File Reference	38
4.26.1 Detailed Description	39
4.26.2 Function Documentation	39
4.27 UART.h File Reference	41
4.27.1 Detailed Description	42
4.27.2 Function Documentation	42
4.28 main.c File Reference	43
4.28.1 Function Documentation	44
Index	45

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

FIFO_buffer_t

Array-based FIFO buffer type

3

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

Debug.c	4
Debug.h	
Functions to output debugging information to a serial port via UART	5
Filter.c	6
Filter.h	
Functions to implement digital filters via linear constant coefficient difference equations (LC-CDEs)	6
QRS.c	7
QRS.h	
QRS detection algorithm functions	8
UserCtrl.c	10
UserCtrl.h	
Interface for user control module	10
fifo_buff.c	
Source code file for FIFO buffer type	11
fifo_buff.h	
Header file for FIFO buffer type	14
ADC.c	
Implementation details for ADC functions	17
ADC.h	
Driver module for analog-to-digital conversion (ADC)	18
GPIO.c	19
GPIO.h	
Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts	21
ILI9341.c	24
ILI9341.h	
Driver module for interfacing with an ILI9341 LCD driver	25
isr.c	
Source code for interrupt service routines (ISRs)	26
PLL.c	
Implementation details for phase-lock-loop (PLL) functions	27
PLL.h	
Driver module for activating the phase-locked-loop (PLL)	28
SPI.c	29

SPI.h	
Driver module for using the serial peripheral interface (SPI) protocol	29
SysTick.c	
Implementation details for SysTick functions	30
SysTick.h	
Driver module for using SysTick-based timing and/or interrupts	32
Timer.c	34
Timer.h	
Implementation for timer module	36
UART.c	
Source code for UART module	38
UART.h	
Driver module for UART1	41
main.c	43

3 Data Structure Documentation

3.1 FIFO_buffer_t Struct Reference

Array-based FIFO buffer type.

Data Fields

- volatile uint16_t * [front_ptr](#)
- volatile uint16_t * [rear_ptr](#)
- volatile uint32_t [curr_size](#)
- uint32_t [MAX_SIZE](#)

3.1.1 Detailed Description

Array-based FIFO buffer type.

Parameters

<i>front_ptr</i>	pointer to the first element of the buffer.
<i>rear_ptr</i>	pointer to the last element of the buffer.
<i>curr_size</i>	current number of elements within the buffer.
<i>MAX_SIZE</i>	maximum number of elements allowed within buffer.

3.1.2 Field Documentation

curr_size

```
volatile uint32_t FIFO_buffer_t::curr_size
```

front_ptr

```
volatile uint16_t* FIFO_buffer_t::front_ptr
```

MAX_SIZE

```
uint32_t FIFO_buffer_t::MAX_SIZE
```

rear_ptr

```
volatile uint16_t* FIFO_buffer_t::rear_ptr
```

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

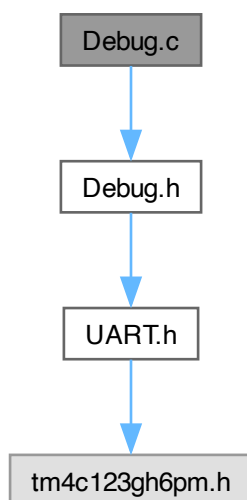
- [fifo_buff.c](#)

4 File Documentation

4.1 Debug.c File Reference

```
#include "Debug.h"
```

Include dependency graph for Debug.c:

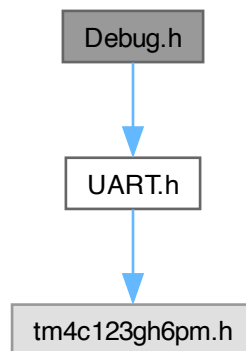


4.2 Debug.h File Reference

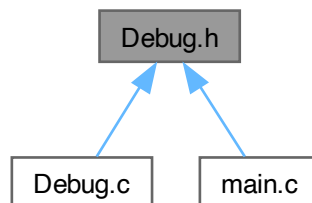
Functions to output debugging information to a serial port via UART.

```
#include "UART.h"
```

Include dependency graph for Debug.h:



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



4.2.1 Detailed Description

Functions to output debugging information to a serial port via UART.

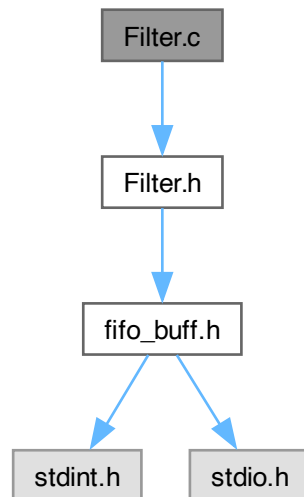
Author

Bryan McElvy

4.3 Filter.c File Reference

```
#include "Filter.h"
```

Include dependency graph for Filter.c:

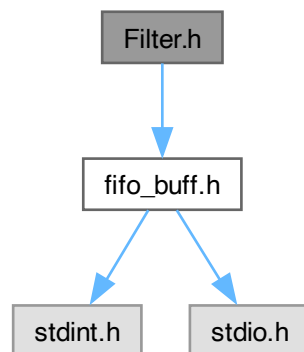


4.4 Filter.h File Reference

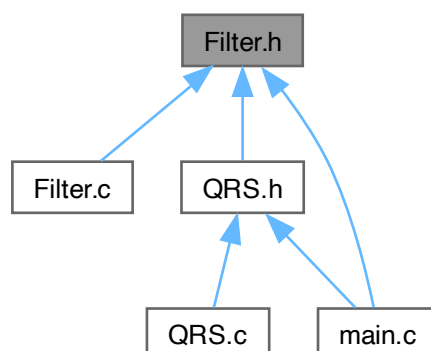
Functions to implement digital filters via linear constant coefficient difference equations (LCCDEs).

```
#include "fifo_buff.h"
```

Include dependency graph for Filter.h:



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



4.4.1 Detailed Description

Functions to implement digital filters via linear constant coefficient difference equations (LCCDEs).

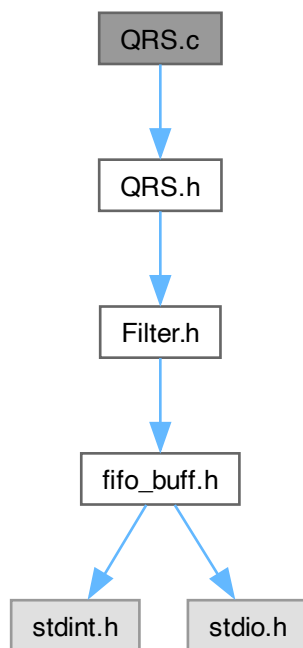
Author

Bryan McElvy

4.5 QRS.c File Reference

```
#include "QRS.h"
```

Include dependency graph for QRS.c:

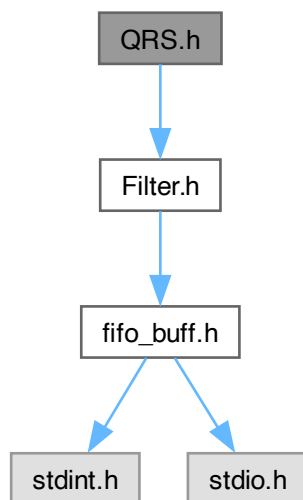


4.6 QRS.h File Reference

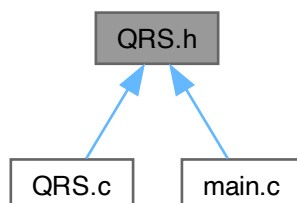
QRS detection algorithm functions.

```
#include "Filter.h"
```

Include dependency graph for QRS.h:



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



4.6.1 Detailed Description

QRS detection algorithm functions.

Author

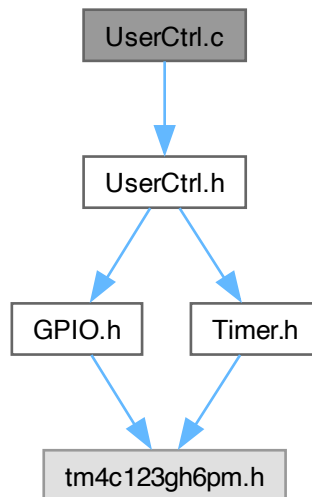
Bryan McElvy

This module contains functions for detecting heart rate (HR) using a simplified version of the Pan-Tompkins algorithm.

4.7 UserCtrl.c File Reference

```
#include "UserCtrl.h"
```

Include dependency graph for UserCtrl.c:



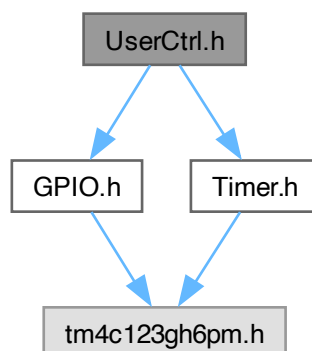
4.8 UserCtrl.h File Reference

Interface for user control module.

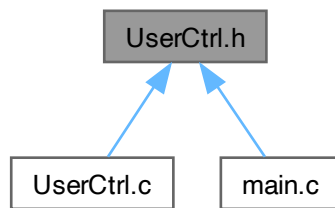
```
#include "GPIO.h"
```

```
#include "Timer.h"
```

Include dependency graph for UserCtrl.h:



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



Functions

- void [UserCtrl_Init](#) ()
Initializes the UserCtrl module and its dependencies (Timer0B and GPIO_PortF)

4.8.1 Detailed Description

Interface for user control module.

Author

Bryan McElvy

4.8.2 Function Documentation

UserCtrl_Init()

```
void UserCtrl_Init ( )
```

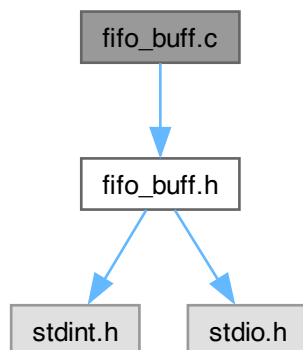
Initializes the UserCtrl module and its dependencies (Timer0B and GPIO_PortF)

4.9 fifo_buff.c File Reference

Source code file for FIFO buffer type.

```
#include "fifo_buff.h"
```

Include dependency graph for fifo_buff.c:



Data Structures

- struct [FIFO_buffer_t](#)
Array-based FIFO buffer type.

Functions

- `FIFO_buffer_t` [FIFO_init](#) (`uint32_t` `buffer_size`)
Initializes a FIFO buffer with the specified size.
- void [FIFO_add_sample](#) (`FIFO_buffer_t` *`FIFO_ptr`, `uint16_t` `sample`)
Adds a 16-bit sample to the end of the FIFO buffer at the specified address.
- `uint16_t` [FIFO_rem_sample](#) (`FIFO_buffer_t` *`FIFO_ptr`)
Removes the first element of the FIFO buffer at the specified address.
- `uint32_t` [FIFO_get_size](#) (`FIFO_buffer_t` *`FIFO_ptr`)
Gets the size of the FIFO buffer at the specified address.
- void [FIFO_show_data](#) (`FIFO_buffer_t` *`FIFO_ptr`)
Shows all of the items in the FIFO buffer at the specified address. NOTE: Intended for debugging purposes only.

4.9.1 Detailed Description

Source code file for FIFO buffer type.

Author

Bryan McElvy

4.9.2 Function Documentation

`FIFO_add_sample()`

```
void FIFO_add_sample (
    FIFO_buffer_t * FIFO_ptr,
    uint16_t sample )
```

Adds a 16-bit sample to the end of the FIFO buffer at the specified address.

Parameters

<i>FIFO_buffer</i>	pointer to FIFO buffer
<i>sample</i>	data sample to be added

Returns

None

FIFO_get_size()

```
uint32_t FIFO_get_size (
    FIFO_buffer_t * FIFO_ptr )
```

Gets the size of the FIFO buffer at the specified address.

Parameters

<i>FIFO_ptr</i>	pointer to FIFO buffer
-----------------	------------------------

Returns

curr_size

FIFO_init()

```
FIFO_buffer_t FIFO_init (
    uint32_t buffer_size )
```

Initializes a FIFO buffer with the specified size.

Parameters

<i>buffer_size</i>	desired buffer size.
--------------------	----------------------

Returns

*FIFO_buffer***FIFO_rem_sample()**

```
uint16_t FIFO_rem_sample (
    FIFO_buffer_t * FIFO_ptr )
```

Removes the first element of the FIFO buffer at the specified address.

Parameters

<i>FIFO_ptr</i>	pointer to FIFO buffer
-----------------	------------------------

Returns

uint16_t

FIFO_show_data()

```
void FIFO_show_data (
    FIFO_buffer_t * FIFO_ptr )
```

Shows all of the items in the FIFO buffer at the specified address. NOTE: Intended for debugging purposes only.

Parameters

<i>FIFO_ptr</i>	pointer to FIFO buffer
-----------------	------------------------

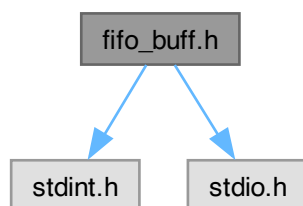
4.10 fifo_buff.h File Reference

Header file for FIFO buffer type.

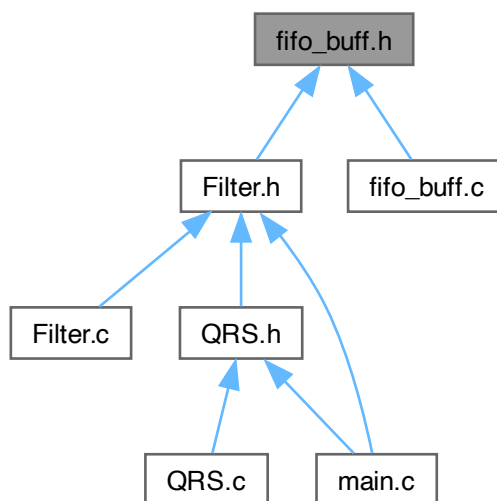
```
#include <stdint.h>
```

```
#include <stdio.h>
```

Include dependency graph for fifo_buff.h:



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



Functions

- `FIFO_buffer_t` [FIFO_init](#) (`uint32_t` buffer_size)
Initializes a FIFO buffer with the specified size.
- `void` [FIFO_add_sample](#) (`FIFO_buffer_t *`FIFO_ptr, `uint16_t` sample)
Adds a 16-bit sample to the end of the FIFO buffer at the specified address.
- `uint16_t` [FIFO_rem_sample](#) (`FIFO_buffer_t *`FIFO_ptr)
Removes the first element of the FIFO buffer at the specified address.
- `uint32_t` [FIFO_get_size](#) (`FIFO_buffer_t *`FIFO_ptr)
Gets the size of the FIFO buffer at the specified address.
- `void` [FIFO_show_data](#) (`FIFO_buffer_t *`FIFO_ptr)
Shows all of the items in the FIFO buffer at the specified address. NOTE: Intended for debugging purposes only.

4.10.1 Detailed Description

Header file for FIFO buffer type.

Author

Bryan McElvy

4.10.2 Function Documentation

FIFO_add_sample()

```

void FIFO_add_sample (
    FIFO_buffer_t * FIFO_ptr,
    uint16_t sample )
  
```

Adds a 16-bit sample to the end of the FIFO buffer at the specified address.

Parameters

<i>FIFO_buffer</i>	pointer to FIFO buffer
<i>sample</i>	data sample to be added

Returns

None

FIFO_get_size()

```
uint32_t FIFO_get_size (
    FIFO_buffer_t * FIFO_ptr )
```

Gets the size of the FIFO buffer at the specified address.

Parameters

<i>FIFO_ptr</i>	pointer to FIFO buffer
-----------------	------------------------

Returns

curr_size

FIFO_init()

```
FIFO_buffer_t FIFO_init (
    uint32_t buffer_size )
```

Initializes a FIFO buffer with the specified size.

Parameters

<i>buffer_size</i>	desired buffer size.
--------------------	----------------------

Returns

FIFO_buffer

FIFO_rem_sample()

```
uint16_t FIFO_rem_sample (
    FIFO_buffer_t * FIFO_ptr )
```

Removes the first element of the FIFO buffer at the specified address.

Parameters

<i>FIFO_ptr</i>	pointer to FIFO buffer
-----------------	------------------------

Returns

uint16_t

FIFO_show_data()

```
void FIFO_show_data (
    FIFO_buffer_t * FIFO_ptr )
```

Shows all of the items in the FIFO buffer at the specified address. NOTE: Intended for debugging purposes only.

Parameters

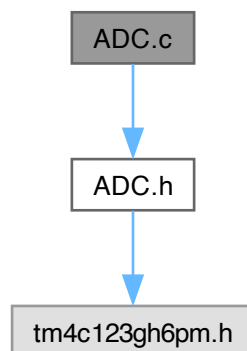
<i>FIFO_ptr</i>	pointer to FIFO buffer
-----------------	------------------------

4.11 ADC.c File Reference

Implementation details for ADC functions.

```
#include "ADC.h"
```

Include dependency graph for ADC.c:



4.11.1 Detailed Description

Implementation details for ADC functions.

Author

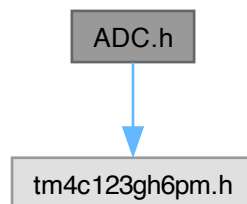
Bryan McElvy

4.12 ADC.h File Reference

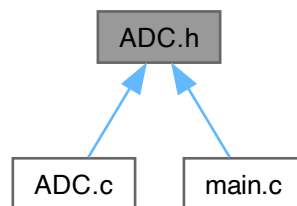
Driver module for analog-to-digital conversion (ADC)

```
#include "tm4c123gh6pm.h"
```

Include dependency graph for ADC.h:



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



4.12.1 Detailed Description

Driver module for analog-to-digital conversion (ADC)

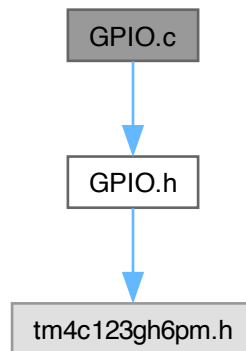
Author

Bryan McElvy

4.13 GPIO.c File Reference

```
#include "GPIO.h"
```

Include dependency graph for GPIO.c:



Functions

- void [GPIO_PF_Init](#) (void)
Initialize GPIO Port F.
- void [GPIO_PF_LED_Init](#) (void)
Initialize PF1-3 to interface the LaunchPad's onboard RGB LED.
- void [GPIO_PF_LED_Write](#) (uint8_t color_mask, uint8_t turn_on)
Write a 1 or 0 to the selected LED(s).
- void [GPIO_PF_LED_Toggle](#) (uint8_t color_mask)
Toggle the selected LED(s).
- void [GPIO_PF_Sw_Init](#) (void)
Initialize PF0/4 to interface the LaunchPad's onboard switches. PF4 is Sw1, and PF0 is Sw2.
- void [GPIO_PF_Interrupt_Init](#) (void)
Initialize GPIO Port F interrupts via Sw1 and Sw2.

4.13.1 Function Documentation

GPIO_PF_Init()

```
void GPIO_PF_Init (  
    void )
```

Initialize GPIO Port F.

GPIO_PF_Interrupt_Init()

```
void GPIO_PF_Interrupt_Init (  
    void )
```

Initialize GPIO Port F interrupts via Sw1 and Sw2.

Here is the call graph for this function:



GPIO_PF_LED_Init()

```
void GPIO_PF_LED_Init (  
    void )
```

Initialize PF1-3 to interface the LaunchPad's onboard RGB LED.

Here is the call graph for this function:



GPIO_PF_LED_Toggle()

```
void GPIO_PF_LED_Toggle (  
    uint8_t color_mask )
```

Toggle the selected LED(s).

Parameters

<i>color_mask</i>	Hex. number of LED pin(s) to write to. 0x02 (PF1) – RED; 0x04 (PF2) – BLUE; 0x08 (PF3) – GREEN
-------------------	--

GPIO_PF_LED_Write()

```
void GPIO_PF_LED_Write (
    uint8_t color_mask,
    uint8_t on_or_off )
```

Write a 1 or 0 to the selected LED(s).

Parameters

<i>color_mask</i>	Hex. number of LED pin(s) to write to. 0x02 (PF1) – RED; 0x04 (PF2) – BLUE; 0x08 (PF3) – GREEN
<i>on_or_off</i>	=0 for OFF, >=1 for ON

GPIO_PF_Sw_Init()

```
void GPIO_PF_Sw_Init (
    void )
```

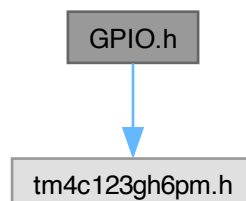
Initialize PF0/4 to interface the LaunchPad's onboard switches. PF4 is Sw1, and PF0 is Sw2.

Here is the call graph for this function:

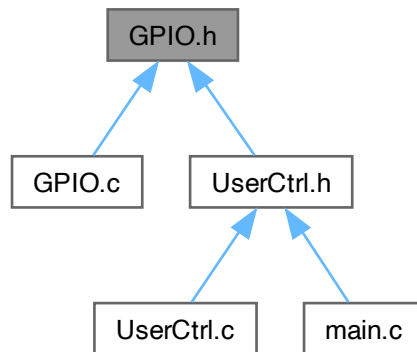
**4.14 GPIO.h File Reference**

Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts.

```
#include "tm4c123gh6pm.h"
Include dependency graph for GPIO.h:
```



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



Functions

- void `GPIO_PF_Init` (void)
Initialize GPIO Port F.
- void `GPIO_PF_LED_Init` (void)
Initialize PF1-3 to interface the LaunchPad's onboard RGB LED.
- void `GPIO_PF_LED_Write` (uint8_t color_mask, uint8_t on_or_off)
Write a 1 or 0 to the selected LED(s).
- void `GPIO_PF_LED_Toggle` (uint8_t color_mask)
Toggle the selected LED(s).
- void `GPIO_PF_Sw_Init` (void)
Initialize PF0/4 to interface the LaunchPad's onboard switches. PF4 is Sw1, and PF0 is Sw2.
- void `GPIO_PF_Interrupt_Init` (void)
Initialize GPIO Port F interrupts via Sw1 and Sw2.

4.14.1 Detailed Description

Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts.

Author

Bryan McElvy

4.14.2 Function Documentation

`GPIO_PF_Init()`

```
void GPIO_PF_Init (  
    void )
```

Initialize GPIO Port F.

GPIO_PF_Interrupt_Init()

```
void GPIO_PF_Interrupt_Init (
    void )
```

Initialize GPIO Port F interrupts via Sw1 and Sw2.

Here is the call graph for this function:

**GPIO_PF_LED_Init()**

```
void GPIO_PF_LED_Init (
    void )
```

Initialize PF1-3 to interface the LaunchPad's onboard RGB LED.

Here is the call graph for this function:

**GPIO_PF_LED_Toggle()**

```
void GPIO_PF_LED_Toggle (
    uint8_t color_mask )
```

Toggle the selected LED(s).

Parameters

<i>color_mask</i>	Hex. number of LED pin(s) to write to. 0x02 (PF1) – RED; 0x04 (PF2) – BLUE; 0x08 (PF3) – GREEN
-------------------	--

GPIO_PF_LED_Write()

```
void GPIO_PF_LED_Write (
    uint8_t color_mask,
    uint8_t on_or_off )
```

Write a 1 or 0 to the selected LED(s).

Parameters

<i>color_mask</i>	Hex. number of LED pin(s) to write to. 0x02 (PF1) – RED; 0x04 (PF2) – BLUE; 0x08 (PF3) – GREEN
<i>on_or_off</i>	=0 for OFF, >=1 for ON

GPIO_PF_Sw_Init()

```
void GPIO_PF_Sw_Init (
    void )
```

Initialize PF0/4 to interface the LaunchPad's onboard switches. PF4 is Sw1, and PF0 is Sw2.

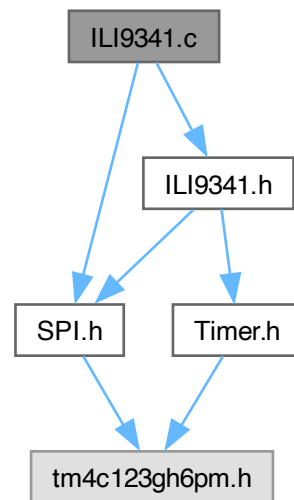
Here is the call graph for this function:



4.15 ILI9341.c File Reference

```
#include "ILI9341.h"
#include "SPI.h"
```

Include dependency graph for ILI9341.c:

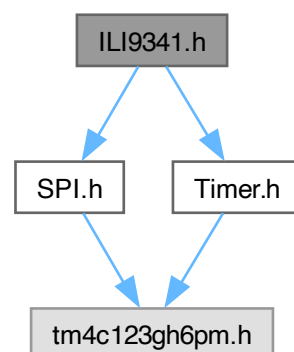


4.16 ILI9341.h File Reference

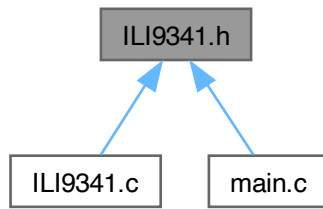
Driver module for interfacing with an ILI9341 LCD driver.

```
#include "SPI.h"  
#include "Timer.h"
```

Include dependency graph for ILI9341.h:



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



4.16.1 Detailed Description

Driver module for interfacing with an ILI9341 LCD driver.

Author

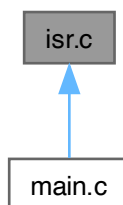
Bryan McElvy

This module contains functions for initializing and outputting graphical data to a 240RGBx320 resolution, 262K color-depth liquid crystal display (LCD). The module interfaces the LaunchPad (or any other board featuring the TM4C123GH6PM microcontroller) with an ILI9341 LCD driver chip via the SPI (serial peripheral interface) protocol.

4.17 isr.c File Reference

Source code for interrupt service routines (ISRs)

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



4.17.1 Detailed Description

Source code for interrupt service routines (ISRs)

Author

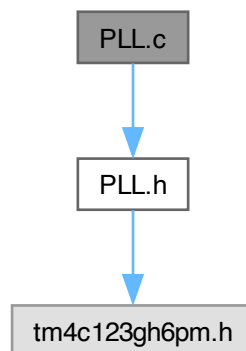
Bryan McElvy

4.18 PLL.c File Reference

Implementation details for phase-lock-loop (PLL) functions.

```
#include "PLL.h"
```

Include dependency graph for PLL.c:



Functions

- void [PLL_Init](#) (void)
Initializes the phase-locked-loop (PLL), allowing a bus frequency of 80[MHz].

4.18.1 Detailed Description

Implementation details for phase-lock-loop (PLL) functions.

Author

Bryan McElvy

4.18.2 Function Documentation

PLL_Init()

```
void PLL_Init (
    void )
```

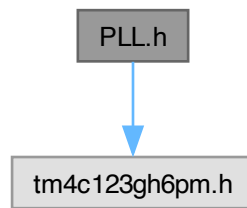
Initializes the phase-locked-loop (PLL), allowing a bus frequency of 80[MHz].

4.19 PLL.h File Reference

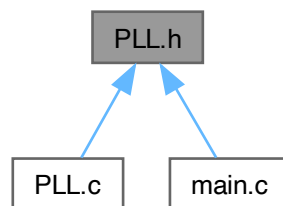
Driver module for activating the phase-locked-loop (PLL).

```
#include "tm4c123gh6pm.h"
```

Include dependency graph for PLL.h:



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



Functions

- void [PLL_Init](#) (void)
Initializes the phase-locked-loop (PLL), allowing a bus frequency of 80[MHz].

4.19.1 Detailed Description

Driver module for activating the phase-locked-loop (PLL).

Author

Bryan McElvy

4.19.2 Function Documentation

PLL_Init()

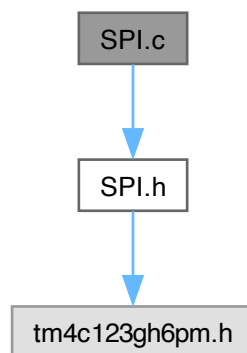
```
void PLL_Init (
    void )
```

Initializes the phase-locked-loop (PLL), allowing a bus frequency of 80[MHz].

4.20 SPI.c File Reference

```
#include "SPI.h"
```

Include dependency graph for SPI.c:

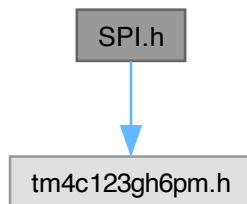


4.21 SPI.h File Reference

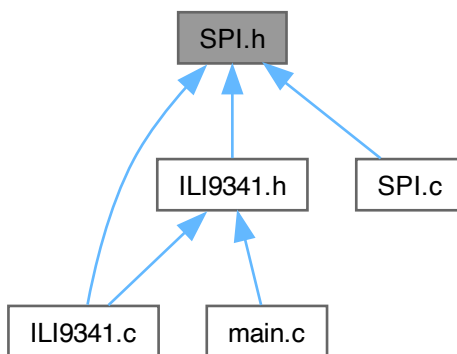
Driver module for using the serial peripheral interface (SPI) protocol.

```
#include "tm4c123gh6pm.h"
```

Include dependency graph for SPI.h:



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



4.21.1 Detailed Description

Driver module for using the serial peripheral interface (SPI) protocol.

Author

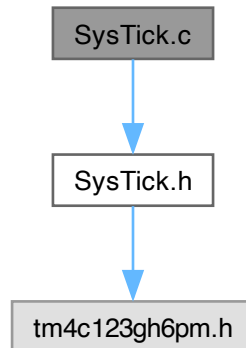
Bryan McElvy

4.22 SysTick.c File Reference

Implementation details for SysTick functions.


```
#include "SysTick.h"
```

Include dependency graph for SysTick.c:



Functions

- void [SysTick_Timer_Init](#) (void)
Initialize SysTick for timing purposes.
- void [SysTick_Wait1ms](#) (uint32_t time_ms)
Delay for specified amount of time in [ms]. Assumes $f_{bus} = 80\text{[MHz]}$.
- void [SysTick_Interrupt_Init](#) (uint32_t time_ms)
Initialize SysTick for interrupts.

4.22.1 Detailed Description

Implementation details for SysTick functions.

Author

Bryan McElvy

4.22.2 Function Documentation

SysTick_Interrupt_Init()

```
void SysTick_Interrupt_Init (
    uint32_t time_ms )
```

Initialize SysTick for interrupts.

Parameters

<i>time_ms</i>	Time in [ms] between interrupts. Cannot be more than 200[ms].
----------------	---

SysTick_Timer_Init()

```
void SysTick_Timer_Init (
    void )
```

Initialize SysTick for timing purposes.

SysTick_Wait1ms()

```
void SysTick_Wait1ms (
    uint32_t time_ms )
```

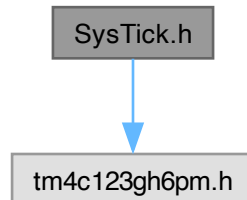
Delay for specified amount of time in [ms]. Assumes f_{bus} = 80[MHz].

4.23 SysTick.h File Reference

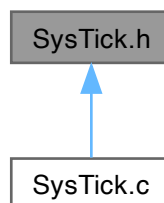
Driver module for using SysTick-based timing and/or interrupts.

```
#include "tm4c123gh6pm.h"
```

Include dependency graph for SysTick.h:



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



Functions

- void `SysTick_Timer_Init` (void)
Initialize SysTick for timing purposes.
- void `SysTick_Wait1ms` (uint32_t delay_ms)
Delay for specified amount of time in [ms]. Assumes f_bus = 80[MHz].
- void `SysTick_Interrupt_Init` (uint32_t time_ms)
Initialize SysTick for interrupts.

4.23.1 Detailed Description

Driver module for using SysTick-based timing and/or interrupts.

Author

Bryan McElvy

4.23.2 Function Documentation

`SysTick_Interrupt_Init()`

```
void SysTick_Interrupt_Init (  
    uint32_t time_ms )
```

Initialize SysTick for interrupts.

Parameters

<code>time_ms</code>	Time in [ms] between interrupts. Cannot be more than 200[ms].
----------------------	---

`SysTick_Timer_Init()`

```
void SysTick_Timer_Init (  
    void )
```

Initialize SysTick for timing purposes.

`SysTick_Wait1ms()`

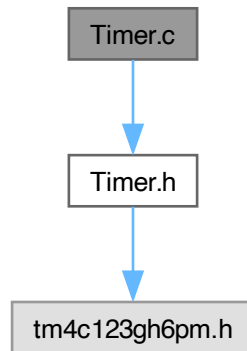
```
void SysTick_Wait1ms (  
    uint32_t delay_ms )
```

Delay for specified amount of time in [ms]. Assumes f_bus = 80[MHz].

4.24 Timer.c File Reference

```
#include "Timer.h"
```

Include dependency graph for Timer.c:



Functions

- void [Timer0A_Init](#) (void)
Initialize timer 0 as 32-bit, one-shot, countdown timer.
- void [Timer0A_Start](#) (uint32_t time_ms)
Count down starting from the inputted value.
- uint8_t [Timer0A_isCounting](#) (void)
Returns 1 if Timer0 is still counting and 0 if not.
- void [Timer0A_Wait1ms](#) (uint32_t time_ms)
Wait for the specified amount of time in [ms].
- void [Timer1A_Init](#) (uint32_t time_ms)
Initialize timer 1 as a 32-bit, periodic, countdown timer with interrupts.

4.24.1 Function Documentation

Timer0A_Init()

```
void Timer0A_Init (  
    void )
```

Initialize timer 0 as 32-bit, one-shot, countdown timer.

Timer0A_isCounting()

```
uint8_t Timer0A_isCounting (  
    void )
```

Returns 1 if Timer0 is still counting and 0 if not.

Returns

uint8_t status

Timer0A_Start()

```
void Timer0A_Start (
    uint32_t time_ms )
```

Count down starting from the inputted value.

Parameters

<i>time_ms</i>	Time in [ms] to load into Timer 0. Must be ≤ 53 seconds.
----------------	---

Timer0A_Wait1ms()

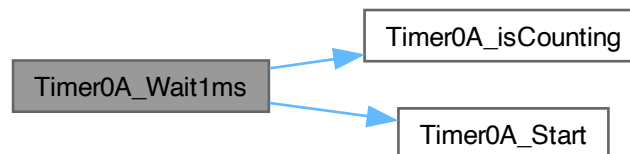
```
void Timer0A_Wait1ms (
    uint32_t time_ms )
```

Wait for the specified amount of time in [ms].

Parameters

<i>time_ms</i>	Time in [ms] to load into Timer 0. Must be ≤ 53 seconds.
----------------	---

Here is the call graph for this function:

**Timer1A_Init()**

```
void Timer1A_Init (
    uint32_t time_ms )
```

Initialize timer 1 as a 32-bit, periodic, countdown timer with interrupts.

Parameters

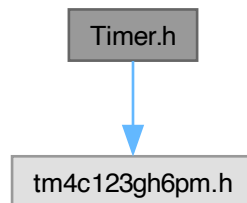
<i>time_ms</i>	Time in [ms] between interrupts. Must be ≤ 53 seconds.
----------------	---

4.25 Timer.h File Reference

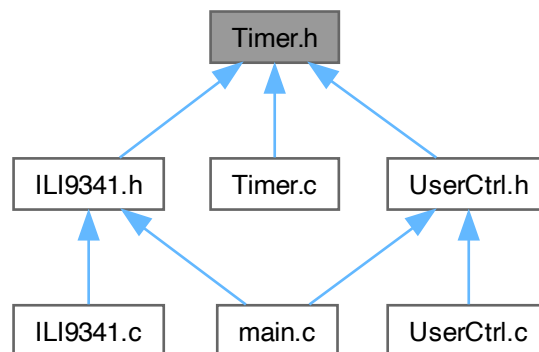
Implementation for timer module.

```
#include "tm4c123gh6pm.h"
```

Include dependency graph for Timer.h:



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



Functions

- void [Timer0A_Init](#) (void)
Initialize timer 0 as 32-bit, one-shot, countdown timer.
- void [Timer0A_Start](#) (uint32_t time_ms)
Count down starting from the inputted value.
- uint8_t [Timer0A_isCounting](#) (void)
Returns 1 if Timer0 is still counting and 0 if not.
- void [Timer0A_Wait1ms](#) (uint32_t time_ms)
Wait for the specified amount of time in [ms].
- void [Timer1A_Init](#) (uint32_t time_ms)
Initialize timer 1 as a 32-bit, periodic, countdown timer with interrupts.

4.25.1 Detailed Description

Implementation for timer module.

Driver module for timing (Timer0) and interrupts (Timer1).

Author

Bryan McElvy

4.25.2 Function Documentation

Timer0A_Init()

```
void Timer0A_Init (
    void )
```

Initialize timer 0 as 32-bit, one-shot, countdown timer.

Timer0A_isCounting()

```
uint8_t Timer0A_isCounting (
    void )
```

Returns 1 if Timer0 is still counting and 0 if not.

Returns

uint8_t status

Timer0A_Start()

```
void Timer0A_Start (
    uint32_t time_ms )
```

Count down starting from the inputted value.

Parameters

<i>time_ms</i>	Time in [ms] to load into Timer 0. Must be \leq 53 seconds.
----------------	---

Timer0A_Wait1ms()

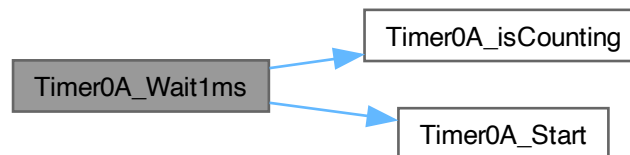
```
void Timer0A_Wait1ms (
    uint32_t time_ms )
```

Wait for the specified amount of time in [ms].

Parameters

<i>time_ms</i>	Time in [ms] to load into Timer 0. Must be ≤ 53 seconds.
----------------	---

Here is the call graph for this function:

**Timer1A_Init()**

```
void Timer1A_Init (
    uint32_t time_ms )
```

Initialize timer 1 as a 32-bit, periodic, countdown timer with interrupts.

Parameters

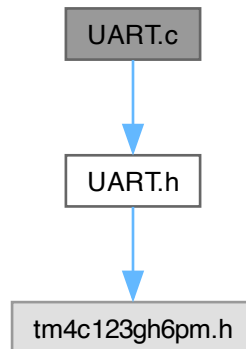
<i>time_ms</i>	Time in [ms] between interrupts. Must be ≤ 53 seconds.
----------------	---

4.26 UART.c File Reference

Source code for UART module.


```
#include "UART.h"
```

Include dependency graph for UART.c:



Functions

- void [UART0_Init](#) (void)
Initialize UART0 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.
- unsigned char [UART0_ReadChar](#) (void)
Read a single character from UART0.
- void [UART0_WriteChar](#) (unsigned char input_char)
Write a single character to UART0.
- void [UART0_WriteStr](#) (unsigned char *str_ptr)
Write a C string to UART0.

4.26.1 Detailed Description

Source code for UART module.

Author

Bryan McElvy

4.26.2 Function Documentation

UART0_Init()

```
void UART0_Init (
    void )
```

Initialize UART0 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.

Given the bus frequency (f_{bus}) and desired baud rate (BR), the baud rate divisor (BRD) can be calculated:
 $BRD = f_{bus} / (16 * BR)$

The integer BRD (IBRD) is simply the integer part of the BRD: $IBRD = int(BRD)$

The fractional BRD (FBRD) is calculated using the fractional part ($mod(BRD, 1)$) of the BRD: $FBRD = int((mod(BRD, 1) * 64) + 0.5)$

NOTE: LCRH must be accessed *AFTER* setting the BRD register0

UART0_ReadChar()

```
unsigned char UART0_ReadChar (
    void )
```

Read a single character from UART0.

Returns

input_char

This function uses busy-wait synchronization to read a character from UART0.

UART0_WriteChar()

```
void UART0_WriteChar (
    unsigned char input_char )
```

Write a single character to UART0.

Parameters

<i>input_char</i>	
-------------------	--

This function uses busy-wait synchronization to write a character to UART0.

UART0_WriteStr()

```
void UART0_WriteStr (
    unsigned char * str_ptr )
```

Write a C string to UART0.

Parameters

<i>str_ptr</i>	pointer to C string
----------------	---------------------

This function uses [UART0_WriteChar\(\)](#) function to write a C string to UART0. The function writes until either the entire string has been written or a null-terminated character has been reached. Here is the call graph for this function:

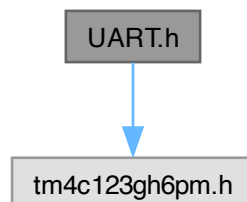


4.27 UART.h File Reference

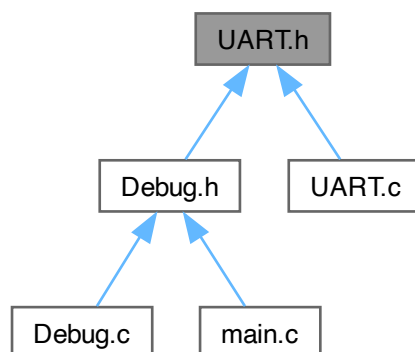
Driver module for UART1.

```
#include "tm4c123gh6pm.h"
```

Include dependency graph for UART.h:



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



Functions

- void [UART0_Init](#) (void)
Initialize UART0 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.
- unsigned char [UART0_ReadChar](#) (void)
Read a single character from UART0.
- void [UART0_WriteChar](#) (unsigned char input_char)
Write a single character to UART0.
- void [UART0_WriteStr](#) (unsigned char *str_ptr)
Write a C string to UART0.

4.27.1 Detailed Description

Driver module for UART1.

Author

Bryan McElvy

4.27.2 Function Documentation

UART0_Init()

```
void UART0_Init (
    void )
```

Initialize UART0 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.

Given the bus frequency (f_{bus}) and desired baud rate (BR), the baud rate divisor (BRD) can be calculated:
 $BRD = f_{bus} / (16 * BR)$

The integer BRD (IBRD) is simply the integer part of the BRD: $IBRD = int(BRD)$

The fractional BRD (FBRD) is calculated using the fractional part ($mod(BRD, 1)$) of the BRD: $FBRD = int((mod(BRD, 1) * 64) + 0.5)$

NOTE: LCRH must be accessed *AFTER* setting the BRD register0

UART0_ReadChar()

```
unsigned char UART0_ReadChar (
    void )
```

Read a single character from UART0.

Returns

input_char

This function uses busy-wait synchronization to read a character from UART0.

UART0_WriteChar()

```
void UART0_WriteChar (
    unsigned char input_char )
```

Write a single character to UART0.

Parameters

input_char	
------------	--

This function uses busy-wait synchronization to write a character to UART0.

UART0_WriteStr()

```
void UART0_WriteStr (
    unsigned char * str_ptr )
```

Write a C string to UART0.

Parameters

<i>str_ptr</i>	pointer to C string
----------------	---------------------

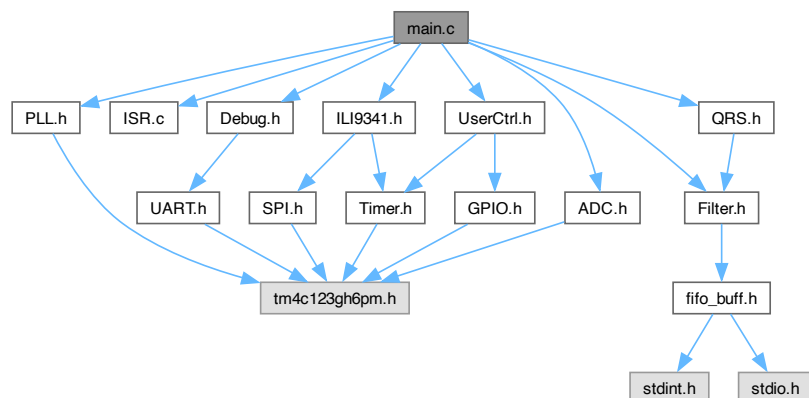
This function uses [UART0_WriteChar\(\)](#) function to write a C string to UART0. The function writes until either the entire string has been written or a null-terminated character has been reached. Here is the call graph for this function:



4.28 main.c File Reference

```
#include "ADC.h"
#include "ISR.c"
#include "ILI9341.h"
#include "PLL.h"
#include "Debug.h"
#include "Filter.h"
#include "QRS.h"
#include "UserCtrl.h"
```

Include dependency graph for main.c:



Functions

- `int main ()`

Main program file for ECG-HRM.

4.28.1 Function Documentation

`main()`

```
int main ( )
```

Main program file for ECG-HRM.

Author

Bryan McElvy

Index

ADC.c, [17](#)
ADC.h, [18](#)

curr_size
 FIFO_buffer_t, [4](#)

Debug.c, [4](#)
Debug.h, [5](#)

FIFO_add_sample
 fifo_buff.c, [12](#)
 fifo_buff.h, [15](#)

fifo_buff.c, [11](#)
 FIFO_add_sample, [12](#)
 FIFO_get_size, [13](#)
 FIFO_init, [13](#)
 FIFO_rem_sample, [13](#)
 FIFO_show_data, [14](#)

fifo_buff.h, [14](#)
 FIFO_add_sample, [15](#)
 FIFO_get_size, [16](#)
 FIFO_init, [16](#)
 FIFO_rem_sample, [16](#)
 FIFO_show_data, [17](#)

FIFO_buffer_t, [3](#)
 curr_size, [4](#)
 front_ptr, [4](#)
 MAX_SIZE, [4](#)
 rear_ptr, [4](#)

FIFO_get_size
 fifo_buff.c, [13](#)
 fifo_buff.h, [16](#)

FIFO_init
 fifo_buff.c, [13](#)
 fifo_buff.h, [16](#)

FIFO_rem_sample
 fifo_buff.c, [13](#)
 fifo_buff.h, [16](#)

FIFO_show_data
 fifo_buff.c, [14](#)
 fifo_buff.h, [17](#)

Filter.c, [6](#)
Filter.h, [6](#)

front_ptr
 FIFO_buffer_t, [4](#)

GPIO.c, [19](#)
 GPIO_PF_Init, [19](#)
 GPIO_PF_Interrupt_Init, [19](#)
 GPIO_PF_LED_Init, [20](#)
 GPIO_PF_LED_Toggle, [20](#)
 GPIO_PF_LED_Write, [20](#)
 GPIO_PF_Sw_Init, [21](#)

GPIO.h, [21](#)
 GPIO_PF_Init, [22](#)
 GPIO_PF_Interrupt_Init, [22](#)
 GPIO_PF_LED_Init, [23](#)
 GPIO_PF_LED_Toggle, [23](#)
 GPIO_PF_LED_Write, [23](#)
 GPIO_PF_Sw_Init, [24](#)

GPIO_PF_Init
 GPIO.c, [19](#)
 GPIO.h, [22](#)

GPIO_PF_Interrupt_Init
 GPIO.c, [19](#)
 GPIO.h, [22](#)

GPIO_PF_LED_Init
 GPIO.c, [20](#)
 GPIO.h, [23](#)

GPIO_PF_LED_Toggle
 GPIO.c, [20](#)
 GPIO.h, [23](#)

GPIO_PF_LED_Write
 GPIO.c, [20](#)
 GPIO.h, [23](#)

GPIO_PF_Sw_Init
 GPIO.c, [21](#)
 GPIO.h, [24](#)

ILI9341.c, [24](#)
ILI9341.h, [25](#)
isr.c, [26](#)

main
 main.c, [44](#)

main.c, [43](#)
 main, [44](#)

MAX_SIZE
 FIFO_buffer_t, [4](#)

PLL.c, [27](#)
 PLL_Init, [28](#)

PLL.h, [28](#)
 PLL_Init, [29](#)

PLL_Init
 PLL.c, [28](#)
 PLL.h, [29](#)

QRS.c, [7](#)
QRS.h, [8](#)

rear_ptr
 FIFO_buffer_t, [4](#)

SPI.c, [29](#)
SPI.h, [29](#)

SysTick.c, [30](#)
 SysTick_Interrupt_Init, [31](#)
 SysTick_Timer_Init, [32](#)
 SysTick_Wait1ms, [32](#)

SysTick.h, [32](#)
 SysTick_Interrupt_Init, [33](#)
 SysTick_Timer_Init, [33](#)

- SysTick_Wait1ms, 33
- SysTick_Interrupt_Init
 - SysTick.c, 31
 - SysTick.h, 33
- SysTick_Timer_Init
 - SysTick.c, 32
 - SysTick.h, 33
- SysTick_Wait1ms
 - SysTick.c, 32
 - SysTick.h, 33
- Timer.c, 34
 - Timer0A_Init, 34
 - Timer0A_isCounting, 34
 - Timer0A_Start, 34
 - Timer0A_Wait1ms, 35
 - Timer1A_Init, 35
- Timer.h, 36
 - Timer0A_Init, 37
 - Timer0A_isCounting, 37
 - Timer0A_Start, 37
 - Timer0A_Wait1ms, 37
 - Timer1A_Init, 38
- Timer0A_Init
 - Timer.c, 34
 - Timer.h, 37
- Timer0A_isCounting
 - Timer.c, 34
 - Timer.h, 37
- Timer0A_Start
 - Timer.c, 34
 - Timer.h, 37
- Timer0A_Wait1ms
 - Timer.c, 35
 - Timer.h, 37
- Timer1A_Init
 - Timer.c, 35
 - Timer.h, 38
- UART.c, 38
 - UART0_Init, 39
 - UART0_ReadChar, 39
 - UART0_WriteChar, 40
 - UART0_WriteStr, 40
- UART.h, 41
 - UART0_Init, 42
 - UART0_ReadChar, 42
 - UART0_WriteChar, 42
 - UART0_WriteStr, 43
- UART0_Init
 - UART.c, 39
 - UART.h, 42
- UART0_ReadChar
 - UART.c, 39
 - UART.h, 42
- UART0_WriteChar
 - UART.c, 40
 - UART.h, 42
- UART0_WriteStr
 - UART.c, 40
 - UART.h, 43
- UART.c, 40
- UART.h, 43
- UserCtrl.c, 10
- UserCtrl.h, 10
 - UserCtrl_Init, 11
- UserCtrl_Init
 - UserCtrl.h, 11