

## ECG-HRM

Generated by Doxygen 1.9.7

<b>1 Module Index</b>	<b>1</b>
1.1 Modules	1
<b>2 Data Structure Index</b>	<b>2</b>
2.1 Data Structures	2
<b>3 File Index</b>	<b>2</b>
3.1 File List	2
<b>4 Module Documentation</b>	<b>3</b>
4.1 Device Drivers	3
4.1.1 Detailed Description	4
4.1.2 ADC  	4
4.1.3 GPIO  	4
4.1.4 ILI9341  	7
4.1.5 PLL  	7
4.1.6 SPI  	7
4.1.7 SysTick  	8
4.1.8 Timer  	8
4.1.9 UART  	10
4.2 Application Software	13
4.3 Program Threads	13
4.3.1 Detailed Description	13
4.3.2 Function Documentation	14
<b>5 Data Structure Documentation</b>	<b>14</b>
5.1 FIFO_buffer_t Struct Reference	14
5.1.1 Detailed Description	14
<b>6 File Documentation</b>	<b>15</b>
6.1 Debug.h File Reference	15
6.1.1 Detailed Description	16
6.2 Filter.h File Reference	16
6.2.1 Detailed Description	17
6.3 QRS.h File Reference	17
6.3.1 Detailed Description	18
6.4 UserCtrl.h File Reference	18
6.4.1 Detailed Description	19
6.4.2 Function Documentation	19
6.5 fifo_buff.c File Reference	19
6.5.1 Detailed Description	20
6.5.2 Function Documentation	20
6.6 fifo_buff.h File Reference	22
6.6.1 Detailed Description	23

6.6.2 Function Documentation . . . . .	23
6.7 ADC.c File Reference . . . . .	25
6.7.1 Detailed Description . . . . .	25
6.8 ADC.h File Reference . . . . .	25
6.8.1 Detailed Description . . . . .	26
6.9 GPIO.h File Reference . . . . .	26
6.9.1 Detailed Description . . . . .	27
6.10 ILI9341.h File Reference . . . . .	28
6.10.1 Detailed Description . . . . .	28
6.11 isr.c File Reference . . . . .	29
6.11.1 Detailed Description . . . . .	29
6.12 PLL.c File Reference . . . . .	29
6.12.1 Detailed Description . . . . .	30
6.13 PLL.h File Reference . . . . .	30
6.13.1 Detailed Description . . . . .	31
6.14 SPI.h File Reference . . . . .	31
6.14.1 Detailed Description . . . . .	32
6.15 SysTick.c File Reference . . . . .	32
6.15.1 Detailed Description . . . . .	33
6.16 SysTick.h File Reference . . . . .	33
6.16.1 Detailed Description . . . . .	34
6.17 Timer.c File Reference . . . . .	34
6.17.1 Detailed Description . . . . .	35
6.18 Timer.h File Reference . . . . .	35
6.18.1 Detailed Description . . . . .	36
6.19 UART.c File Reference . . . . .	36
6.19.1 Detailed Description . . . . .	37
6.20 UART.h File Reference . . . . .	37
6.20.1 Detailed Description . . . . .	38
6.21 main.c File Reference . . . . .	38
6.21.1 Detailed Description . . . . .	39
<b>Index</b>	<b>41</b>

## 1 Module Index

### 1.1 Modules

Here is a list of all modules:

<b>Device Drivers</b>	<b>3</b>
<b>ADC &lt;br&gt;</b>	<b>4</b>

<b>GPIO &lt;br&gt;</b>	<b>4</b>
<b>ILI9341 &lt;br&gt;</b>	<b>7</b>
<b>PLL &lt;br&gt;</b>	<b>7</b>
<b>SPI &lt;br&gt;</b>	<b>7</b>
<b>SysTick &lt;br&gt;</b>	<b>8</b>
<b>Timer &lt;br&gt;</b>	<b>8</b>
<b>UART &lt;br&gt;</b>	<b>10</b>
<b>Application Software</b>	<b>13</b>
<b>Program Threads</b>	<b>13</b>

## 2 Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<b>FIFO_buffer_t</b> Array-based FIFO buffer type	<b>14</b>
--	-----------

## 3 File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

<b>Debug.h</b> Functions to output debugging information to a serial port via UART	<b>15</b>
<b>Filter.h</b> Functions to implement digital filters via linear constant coefficient difference equations (LC-CDEs)	<b>16</b>
<b>QRS.h</b> QRS detection algorithm functions	<b>17</b>
<b>UserCtrl.h</b> Interface for user control module	<b>18</b>
<b>fifo_buff.c</b> Source code file for FIFO buffer type	<b>19</b>
<b>fifo_buff.h</b> Header file for FIFO buffer type	<b>22</b>
<b>ADC.c</b>	<b>25</b>

<a href="#">ADC.h</a>	Driver module for analog-to-digital conversion (ADC)	25
<a href="#">GPIO.h</a>	Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts	26
<a href="#">ILI9341.h</a>	Driver module for interfacing with an ILI9341 LCD driver	28
<a href="#">isr.c</a>	Source code for interrupt service routines (ISRs)	29
<a href="#">PLL.c</a>	Implementation details for phase-lock-loop (PLL) functions	29
<a href="#">PLL.h</a>	Driver module for activating the phase-locked-loop (PLL)	30
<a href="#">SPI.h</a>	Driver module for using the serial peripheral interface (SPI) protocol	31
<a href="#">SysTick.c</a>	Implementation details for SysTick functions	32
<a href="#">SysTick.h</a>	Driver module for using SysTick-based timing and/or interrupts	33
<a href="#">Timer.c</a>	Implementation for timer module	34
<a href="#">Timer.h</a>	Driver module for timing (Timer0) and interrupts (Timer1)	35
<a href="#">UART.c</a>	Source code for UART module	36
<a href="#">UART.h</a>	Driver module for UART1	37
<a href="#">main.c</a>	Main program file for ECG-HRM	38

## 4 Module Documentation

### 4.1 Device Drivers

Device driver modules.

#### Modules

- [ADC <br>](#)  
*Analog-to-digital conversion module.*
- [GPIO <br>](#)  
*GPIO Port F module.*

- [ILI9341 <br>](#)  
*Module for interfacing ILI9341-based RGB LCD via [SPI](#) .*
- [PLL <br>](#)  
*Phase-locked loop module.*
- [SPI <br>](#)  
*Serial peripheral interface module.*
- [SysTick <br>](#)  
*SysTick timing module.*
- [Timer <br>](#)  
*Timer0A module.*
- [UART <br>](#)  
*UART0 module.*

#### 4.1.1 Detailed Description

Device driver modules.

#### 4.1.2 ADC <br>

Analog-to-digital conversion module.

##### Files

- file [ADC.h](#)  
*Driver module for analog-to-digital conversion (ADC)*

#### 4.1.2.1 Detailed Description

Analog-to-digital conversion module.

#### 4.1.3 GPIO <br>

GPIO Port F module.

##### Files

- file [GPIO.h](#)  
*Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts.*

## 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.*

### 4.1.3.1 Detailed Description

GPIO Port F module.

### 4.1.3.2 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:



#### 4.1.4 ILI9341 <br>

Module for interfacing ILI9341-based RGB LCD via [SPI](#) .

Module for interfacing ILI9341-based RGB LCD via [SPI](#) .

#### 4.1.5 PLL <br>

Phase-locked loop module.

##### Functions

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

##### 4.1.5.1 Detailed Description

Phase-locked loop module.

##### 4.1.5.2 Function Documentation

###### PLL\_Init()

```
void PLL_Init (  
    void )
```

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

#### 4.1.6 SPI <br>

Serial peripheral interface module.

Serial peripheral interface module.

#### 4.1.7 SysTick <br>

SysTick timing module.

##### 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.*

##### 4.1.7.1 Detailed Description

SysTick timing module.

##### 4.1.7.2 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.

#### 4.1.8 Timer <br>

Timer0A module.

##### Files

- file [Timer.c](#)  
*Implementation for timer module.*
- file [Timer.h](#)  
*Driver module for timing (Timer0) and interrupts (Timer1).*

## 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].*

### 4.1.8.1 Detailed Description

Timer0A module.

### 4.1.8.2 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

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

### Timer0A\_Wait1ms()

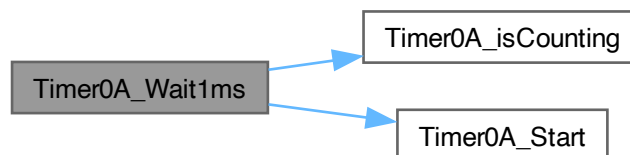
```
void Timer0A_Wait1ms (
    uint32_t time_ms )
```

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

#### Parameters

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

Here is the call graph for this function:



## 4.1.9 UART <br>

UART0 module.

### 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.*

#### 4.1.9.1 Detailed Description

UART0 module.

### 4.1.9.2 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

<code>str_ptr</code>	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:



## 4.2 Application Software

Application-specific modules.

Application-specific modules.

## 4.3 Program Threads

Program Threads.

### Functions

- void [GPIO\\_PortF\\_Handler](#) ()  
*ISR for facilitating user control of program state.*
- void [SysTick\\_Handler](#) ()  
*ISR for collecting ECG samples @  $f_s = 200$  [Hz].*
- void [Timer1A\\_Handler](#) ()  
*ISR for updating the LCD @  $f_s = 30$  [Hz].*
- void [Timer1A\\_Init](#) (uint32\_t time\_ms)  
*Initialize timer 1 as a 32-bit, periodic, countdown timer with interrupts.*
- int **main** ()

### 4.3.1 Detailed Description

Program Threads.

### 4.3.2 Function Documentation

#### GPIO\_PortF\_Handler()

```
void GPIO_PortF_Handler ( )
```

ISR for facilitating user control of program state.

#### SysTick\_Handler()

```
void SysTick_Handler ( )
```

ISR for collecting ECG samples @  $f_s = 200$  [Hz].

#### Timer1A\_Handler()

```
void Timer1A_Handler ( )
```

ISR for updating the LCD @  $f_s = 30$  [Hz].

#### 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 $\leq 53$ seconds.
----------------	---

## 5 Data Structure Documentation

### 5.1 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**

#### 5.1.1 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.

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

- [fifo\\_buff.c](#)

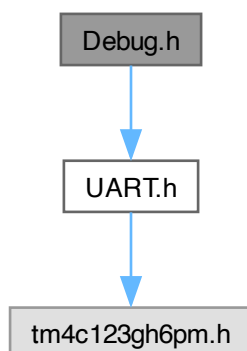
## 6 File Documentation

### 6.1 Debug.h File Reference

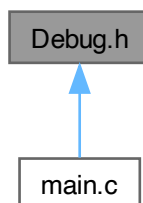
Functions to output debugging information to a serial port via UART.

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

Include dependency graph for Debug.h:



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



### 6.1.1 Detailed Description

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

Author

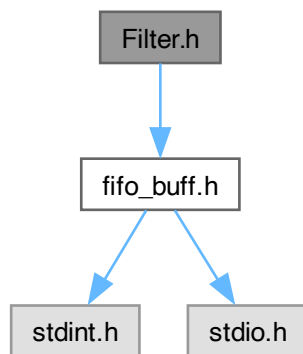
Bryan McElvy

## 6.2 Filter.h File Reference

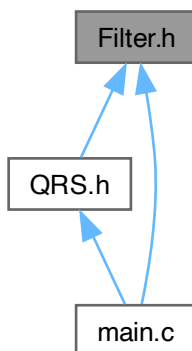
Functions to implement digital filters via linear constant coefficient difference equations (LCCDEs).

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

Include dependency graph for Filter.h:



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



### 6.2.1 Detailed Description

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

Author

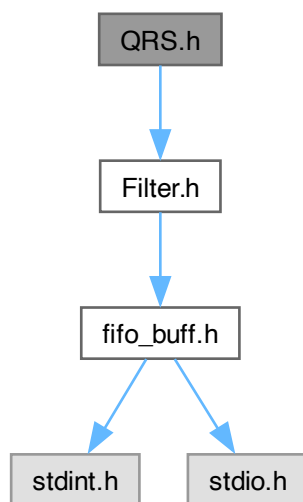
Bryan McElvy

## 6.3 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:



### 6.3.1 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.

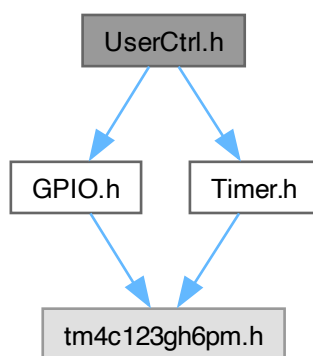
## 6.4 UserCtrl.h File Reference

Interface for user control module.

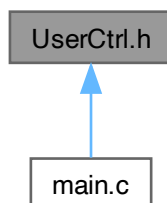
```
#include "GPIO.h"
```

```
#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)*

### 6.4.1 Detailed Description

Interface for user control module.

#### Author

Bryan McElvy

### 6.4.2 Function Documentation

#### UserCtrl\_Init()

```
void UserCtrl_Init ( )
```

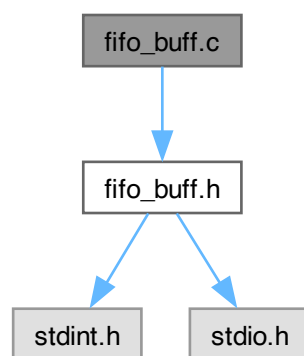
Initializes the UserCtrl module and its dependencies (Timer0B and GPIO\_PortF)

## 6.5 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.*

### 6.5.1 Detailed Description

Source code file for FIFO buffer type.

#### Author

Bryan McElvy

### 6.5.2 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

<code>FIFO_buffer</code>	pointer to FIFO buffer
<code>sample</code>	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.

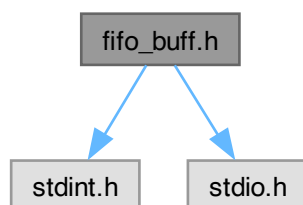
### Parameters

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

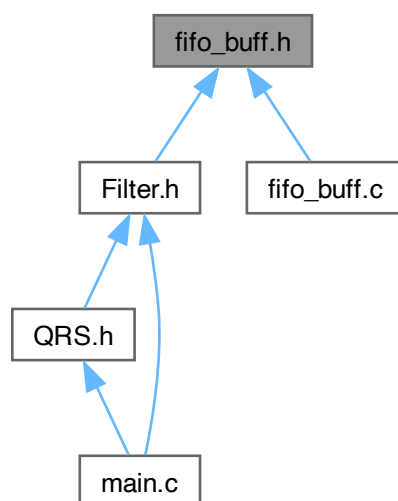
## 6.6 fifo\_buff.h File Reference

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.*

### 6.6.1 Detailed Description

Header file for FIFO buffer type.

#### Author

Bryan McElvy

### 6.6.2 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.

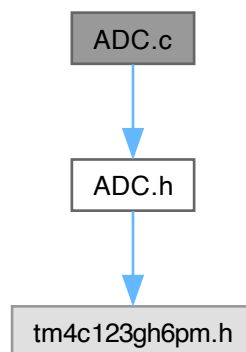
## Parameters

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

## 6.7 ADC.c File Reference

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

Include dependency graph for ADC.c:



### 6.7.1 Detailed Description

## Author

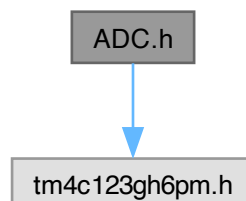
Bryan McElvy

## 6.8 ADC.h File Reference

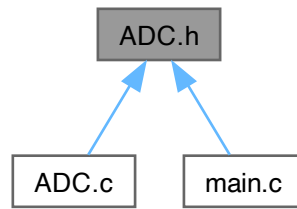
Driver module for analog-to-digital conversion (ADC)

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

Include dependency graph for ADC.h:



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



### 6.8.1 Detailed Description

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

#### Author

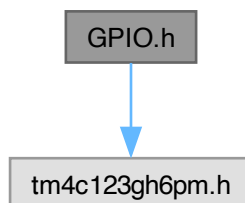
Bryan McElvy

## 6.9 GPIO.h File Reference

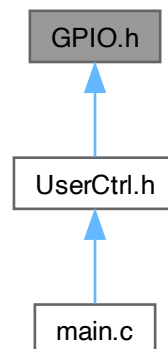
Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts.

```
#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.*

### 6.9.1 Detailed Description

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

#### Author

Bryan McElvy

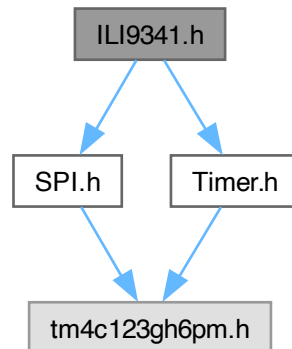
## 6.10 ILI9341.h File Reference

Driver module for interfacing with an ILI9341 LCD driver.

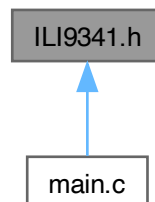
```
#include "SPI.h"
```

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

Include dependency graph for ILI9341.h:



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



### 6.10.1 Detailed Description

Driver module for interfacing with an ILI9341 LCD driver.

**Author**

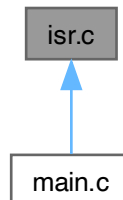
Bryan McElvy

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.

## 6.11 isr.c File Reference

Source code for interrupt service routines (ISRs)

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



### Functions

- void [GPIO\\_PortF\\_Handler](#) ()  
*ISR for facilitating user control of program state.*
- void [SysTick\\_Handler](#) ()  
*ISR for collecting ECG samples @  $f_s = 200$  [Hz].*
- void [Timer1A\\_Handler](#) ()  
*ISR for updating the LCD @  $f_s = 30$  [Hz].*

### 6.11.1 Detailed Description

Source code for interrupt service routines (ISRs)

#### Author

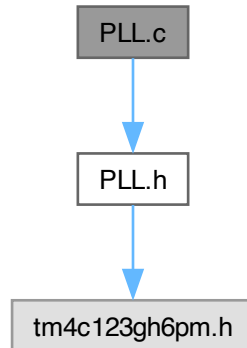
Bryan McElvy

## 6.12 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].*

#### 6.12.1 Detailed Description

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

Author

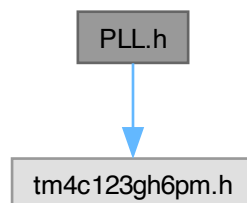
Bryan McElvy

### 6.13 PLL.h File Reference

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

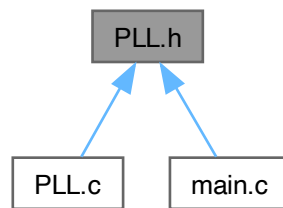
```
#include "tm4c123gh6pm.h"
```

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].*

#### 6.13.1 Detailed Description

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

#### Author

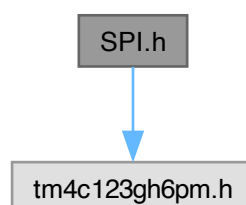
Bryan McElvy

## 6.14 SPI.h File Reference

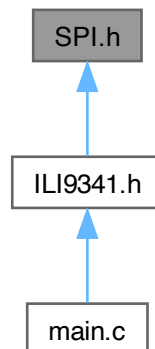
Driver module for using the serial peripheral interface (SPI) protocol.

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

Include dependency graph for `SPI.h`:



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



#### 6.14.1 Detailed Description

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

Author

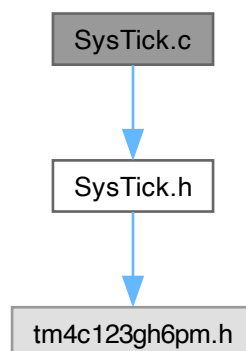
Bryan McElvy

### 6.15 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  $f_{bus} = 80$ [MHz].*
- void [SysTick\\_Interrupt\\_Init](#) (uint32\_t time\_ms)  
*Initialize SysTick for interrupts.*

### 6.15.1 Detailed Description

Implementation details for SysTick functions.

#### Author

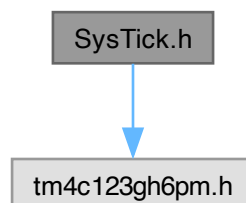
Bryan McElvy

## 6.16 SysTick.h File Reference

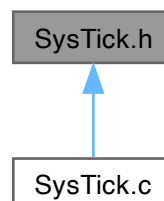
Driver module for using SysTick-based timing and/or interrupts.

```
#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.*

### 6.16.1 Detailed Description

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

Author

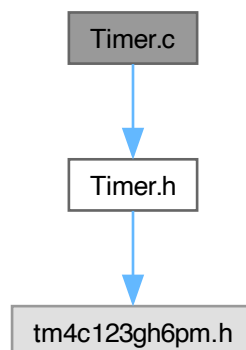
Bryan McElvy

## 6.17 Timer.c File Reference

Implementation for timer module.

```
#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.*

### 6.17.1 Detailed Description

Implementation for timer module.

Author

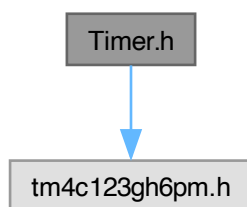
Bryan McElvy

## 6.18 Timer.h File Reference

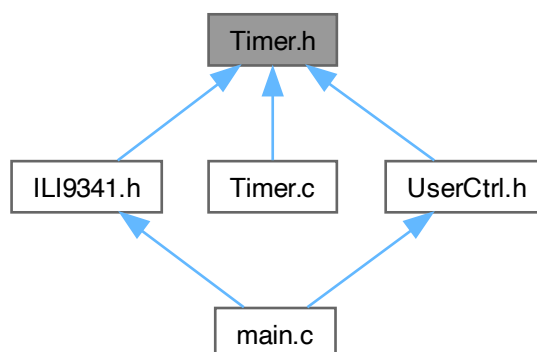
Driver module for timing (Timer0) and interrupts (Timer1).

```
#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.*

### 6.18.1 Detailed Description

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

#### Author

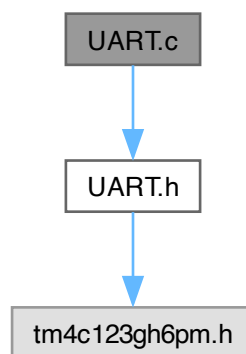
Bryan McElvy

## 6.19 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.*

### 6.19.1 Detailed Description

Source code for UART module.

Author

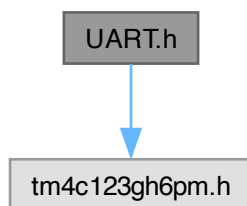
Bryan McElvy

## 6.20 UART.h File Reference

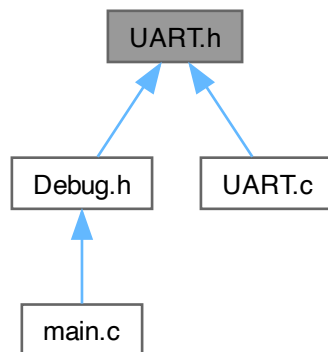
Driver module for UART1.

```
#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.*

### 6.20.1 Detailed Description

Driver module for UART1.

#### Author

Bryan McElvy

## 6.21 main.c File Reference

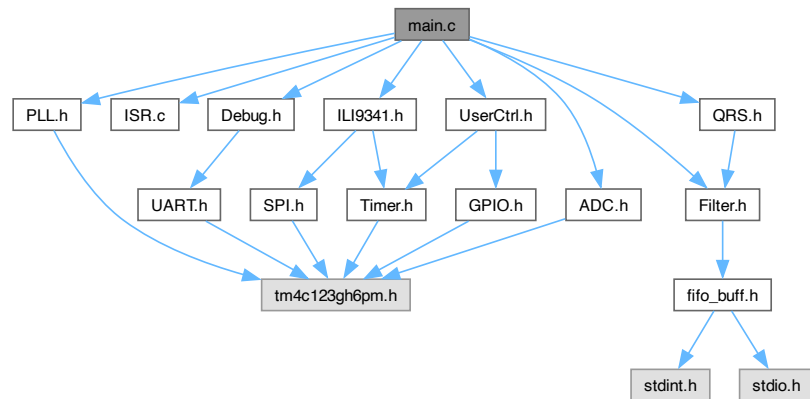
Main program file for ECG-HRM.

```
#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 ()`

### 6.21.1 Detailed Description

Main program file for ECG-HRM.

#### Author

Bryan McElvy



## Index

ADC <br>, 4  
ADC.c, 25  
ADC.h, 25  
Application Software, 13  
  
Debug.h, 15  
Device Drivers, 3  
  
FIFO\_add\_sample  
    fifo\_buff.c, 20  
    fifo\_buff.h, 23  
fifo\_buff.c, 19  
    FIFO\_add\_sample, 20  
    FIFO\_get\_size, 20  
    FIFO\_init, 21  
    FIFO\_rem\_sample, 21  
    FIFO\_show\_data, 21  
fifo\_buff.h, 22  
    FIFO\_add\_sample, 23  
    FIFO\_get\_size, 23  
    FIFO\_init, 24  
    FIFO\_rem\_sample, 24  
    FIFO\_show\_data, 24  
FIFO\_buffer\_t, 14  
FIFO\_get\_size  
    fifo\_buff.c, 20  
    fifo\_buff.h, 23  
FIFO\_init  
    fifo\_buff.c, 21  
    fifo\_buff.h, 24  
FIFO\_rem\_sample  
    fifo\_buff.c, 21  
    fifo\_buff.h, 24  
FIFO\_show\_data  
    fifo\_buff.c, 21  
    fifo\_buff.h, 24  
Filter.h, 16  
  
GPIO <br>, 4  
    GPIO\_PF\_Init, 5  
    GPIO\_PF\_Interrupt\_Init, 5  
    GPIO\_PF\_LED\_Init, 5  
    GPIO\_PF\_LED\_Toggle, 6  
    GPIO\_PF\_LED\_Write, 6  
    GPIO\_PF\_Sw\_Init, 6  
GPIO.h, 26  
GPIO\_PF\_Init  
    GPIO <br>, 5  
GPIO\_PF\_Interrupt\_Init  
    GPIO <br>, 5  
GPIO\_PF\_LED\_Init  
    GPIO <br>, 5  
GPIO\_PF\_LED\_Toggle  
    GPIO <br>, 6  
GPIO\_PF\_LED\_Write  
    GPIO <br>, 6  
  
GPIO\_PF\_Sw\_Init  
    GPIO <br>, 6  
GPIO\_PortF\_Handler  
    Program Threads, 14  
  
ILI9341 <br>, 7  
ILI9341.h, 28  
isr.c, 29  
  
main.c, 38  
  
PLL <br>, 7  
    PLL\_Init, 7  
PLL.c, 29  
PLL.h, 30  
PLL\_Init  
    PLL <br>, 7  
Program Threads, 13  
    GPIO\_PortF\_Handler, 14  
    SysTick\_Handler, 14  
    Timer1A\_Handler, 14  
    Timer1A\_Init, 14  
  
QRS.h, 17  
  
SPI <br>, 7  
SPI.h, 31  
SysTick <br>, 8  
    SysTick\_Interrupt\_Init, 8  
    SysTick\_Timer\_Init, 8  
SysTick.c, 32  
SysTick.h, 33  
SysTick\_Handler  
    Program Threads, 14  
SysTick\_Interrupt\_Init  
    SysTick <br>, 8  
SysTick\_Timer\_Init  
    SysTick <br>, 8  
  
Timer <br>, 8  
    Timer0A\_Init, 9  
    Timer0A\_isCounting, 9  
    Timer0A\_Start, 9  
    Timer0A\_Wait1ms, 9  
Timer.c, 34  
Timer.h, 35  
Timer0A\_Init  
    Timer <br>, 9  
Timer0A\_isCounting  
    Timer <br>, 9  
Timer0A\_Start  
    Timer <br>, 9  
Timer0A\_Wait1ms  
    Timer <br>, 9  
Timer1A\_Handler  
    Program Threads, 14  
Timer1A\_Init

- Program Threads, [14](#)
- UART [<br>](#), [10](#)
  - UART0\_Init, [11](#)
  - UART0\_ReadChar, [11](#)
  - UART0\_WriteChar, [11](#)
  - UART0\_WriteStr, [11](#)
- UART.c, [36](#)
- UART.h, [37](#)
- UART0\_Init
  - UART [<br>](#), [11](#)
- UART0\_ReadChar
  - UART [<br>](#), [11](#)
- UART0\_WriteChar
  - UART [<br>](#), [11](#)
- UART0\_WriteStr
  - UART [<br>](#), [11](#)
- UserCtrl.h, [18](#)
  - UserCtrl\_Init, [19](#)
- UserCtrl\_Init
  - UserCtrl.h, [19](#)