ECG-HRM

Generated by Doxygen 1.9.7

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

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

front_ptr	pointer to the first element of the buffer.
rear_ptr	pointer to the last element of the buffer.
curr_size	current number of elements within the buffer.
MAX_SIZE	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:

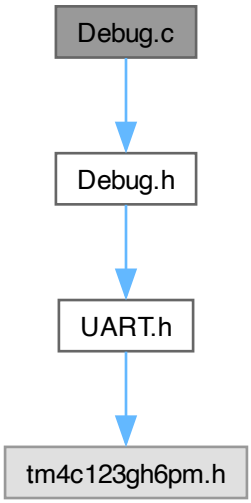
- fifo_buff.c

4 File Documentation

4.1 Debug.c File Reference

```
#include ¨Debug.h¨
```

Include dependency graph for Debug.c:

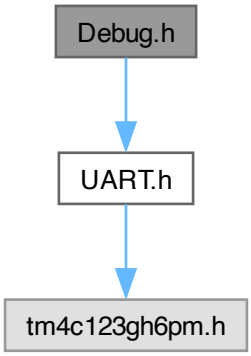


4.2 Debug.h File Reference

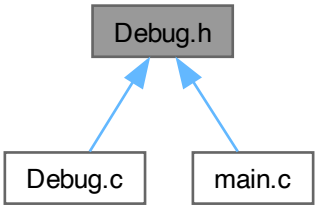
Functions to output debugging information to a serial port via UART.

```
#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

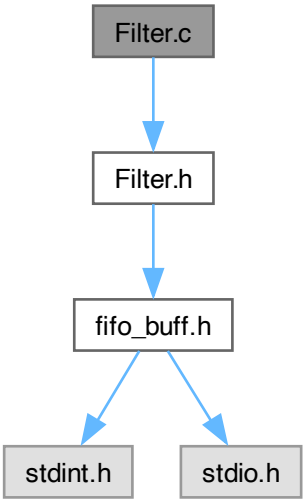
Functions to output debugging information to a serial port via UART.

Author

Bryan McElvy

4.3 Filter.c File Reference

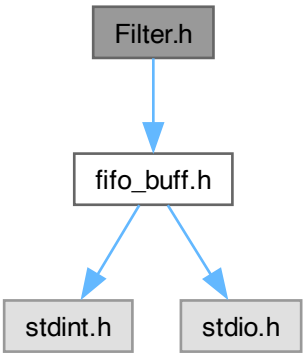
`#include ¨Filter.h¨`
Include dependency graph for Filter.c:



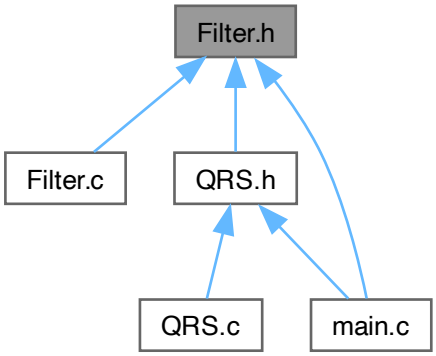
4.4 Filter.h File Reference

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).

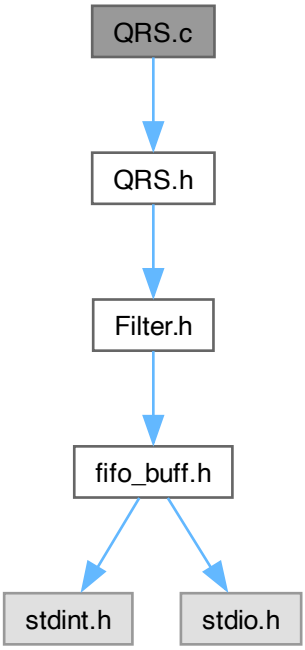
Author

Bryan McElvy

4.5 QRS.c File Reference

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

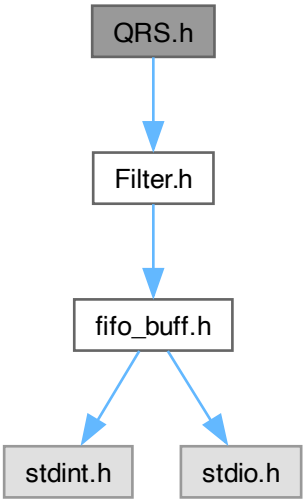
Include dependency graph for QRS.c:



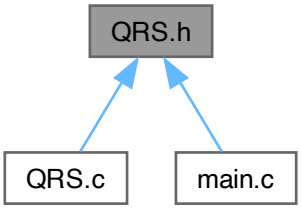
4.6 QRS.h File Reference

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.

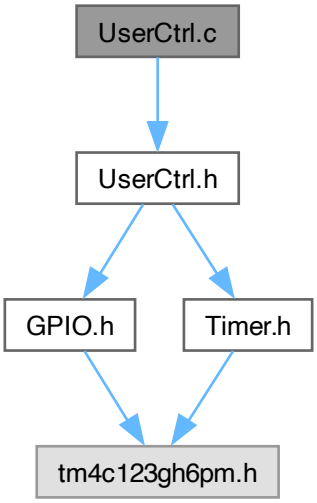
Author

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

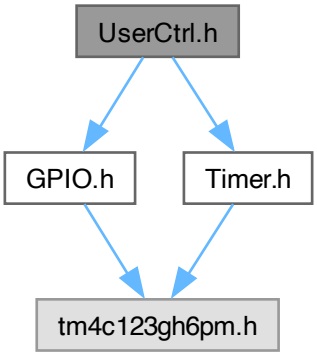
```
#include ¨UserCtrl.h¨
```
Include dependency graph for UserCtrl.c:



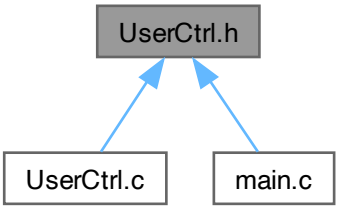
4.8 UserCtrl.h File Reference

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 ( )
```

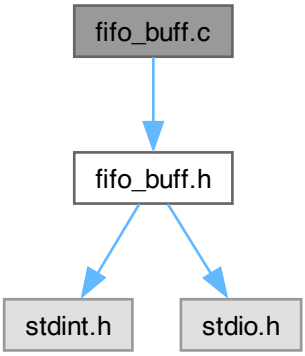
Initializes the UserCtrl module and its dependencies (Timer0B and GPIO_PortF)

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

FIFO_buffer	pointer to FIFO buffer
sample	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

FIFO_ptr	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

buffer_size	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

FIFO_ptr	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

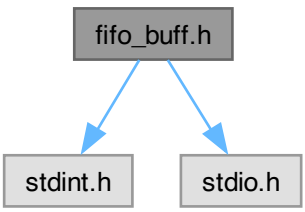
FIFO_ptr	pointer to FIFO buffer

4.10 fifo_buff.h File Reference

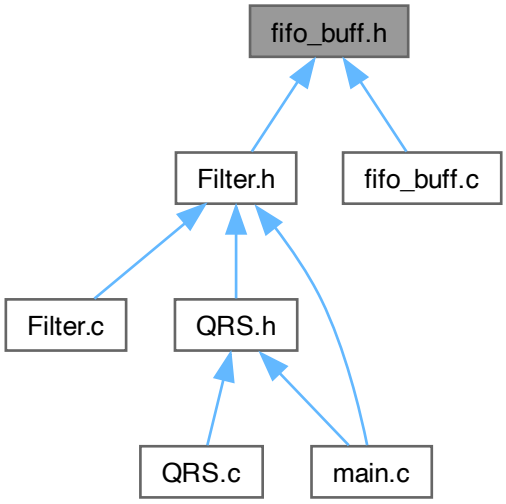
Header file for FIFO buffer type.

```
#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

FIFO_buffer	pointer to FIFO buffer
sample	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

FIFO_ptr	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

buffer_size	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

FIFO_ptr	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

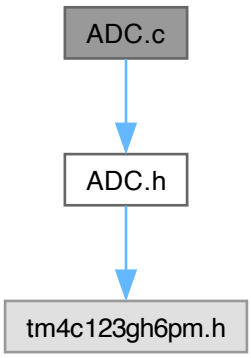
FIFO_ptr	pointer to FIFO buffer

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

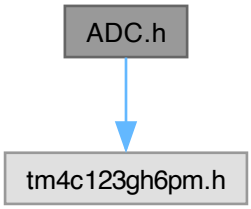
Bryan McElvy

4.12 ADC.h File Reference

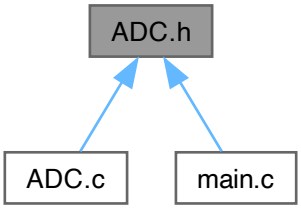
Driver module for analog-to-digital conversion (ADC)

```
#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)

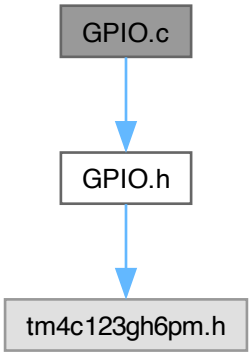
Author

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  )
```

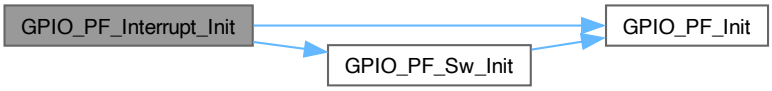
Initialize GPIO Port F.

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

color_mask	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

color_mask	Hex. number of LED pin(s) to write to. 0x02 (PF1) – RED; 0x04 (PF2) – BLUE; 0x08 (PF3) – GREEN
on_or_off	=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.

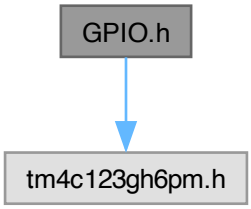
Here is the call graph for this function:

4.14 GPIO.h File Reference

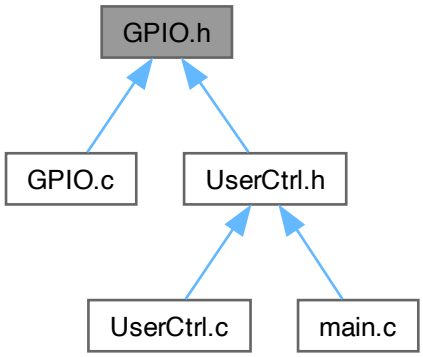
Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts.

```
#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  )
```

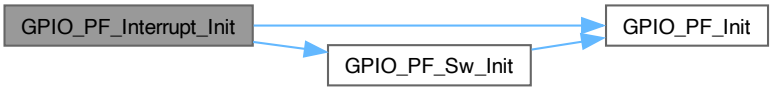
Initialize GPIO Port F.

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

color_mask	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

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

GPIO_PF_Sw_Init()

```
void GPIO_PF_Sw_Init (
             void  )
```

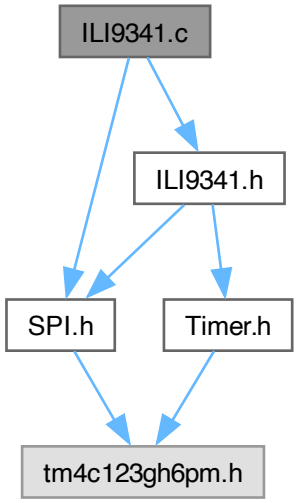
Initialize PF0/4 to interface the LaunchPad's onboard switches. PF4 is Sw1, and PF0 is Sw2.

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:

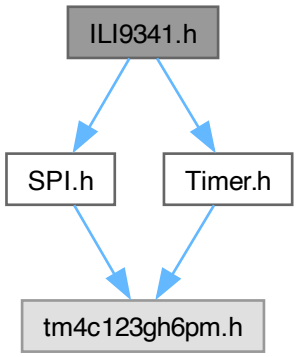


4.16 ILI9341.h File Reference

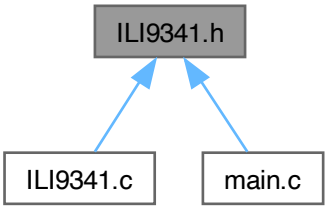
Driver module for interfacing with an ILI9341 LCD driver.

```
#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

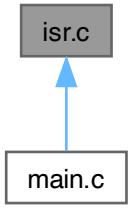
Bryan McElvy

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

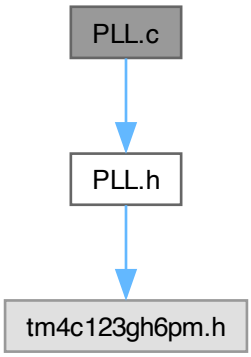
Bryan McElvy

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()

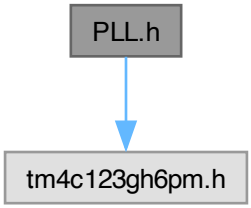
```
void PLL_Init (
            void  )
```

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

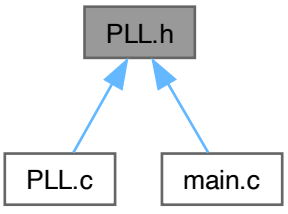
4.19 PLL.h File Reference

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()

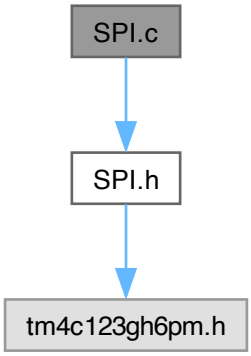
```
void PLL_Init (
            void  )
```

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

4.20 SPI.c File Reference

```
#include ¨SPI.h¨
```

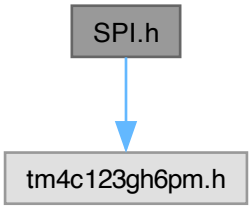
Include dependency graph for SPI.c:



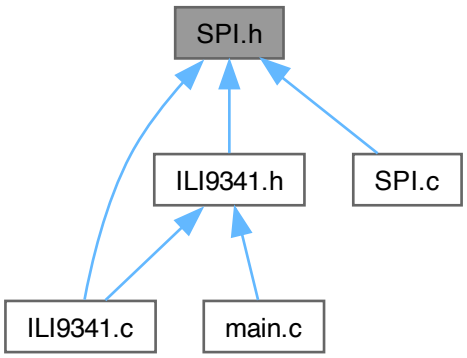
4.21 SPI.h File Reference

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

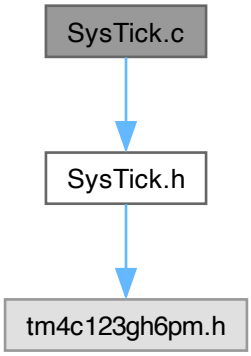
Bryan McElvy

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[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

time_ms	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 )
```

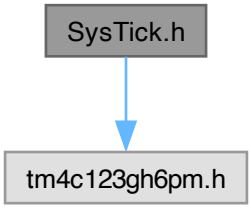
Delay for specified amount of time in [ms]. Assumes f_bus = 80[MHz].

4.23 SysTick.h File Reference

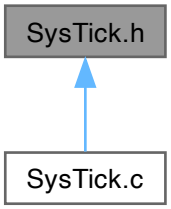
Driver module for using SysTick-based timing and/or interrupts.

```
#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

time_ms	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()

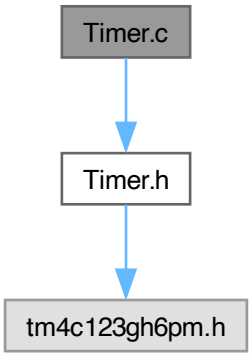
```
void SysTick_Wait1ms (
            uint32_t delay_ms )
```

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

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

Timer0A_Wait1ms()

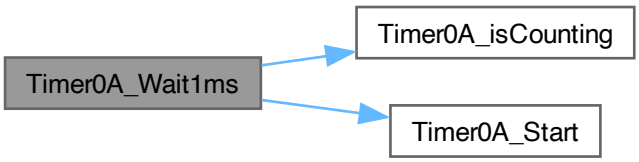
```
void Timer0A_Wait1ms (
            uint32_t time_ms )
```

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

Parameters

time_ms	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

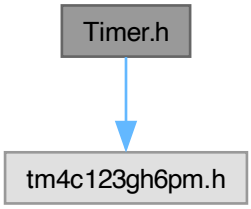
time_ms	Time in [ms] between interrupts. Must be <= 53 seconds.

4.25 Timer.h File Reference

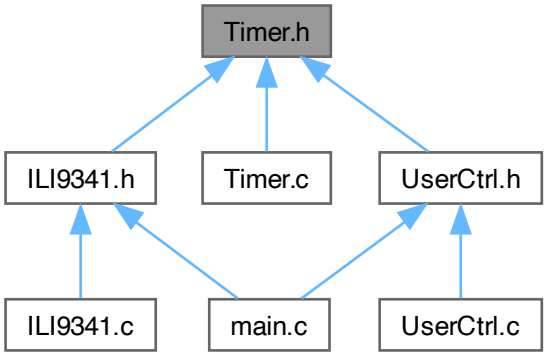
Implementation for timer module.

```
#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

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

Timer0A_Wait1ms()

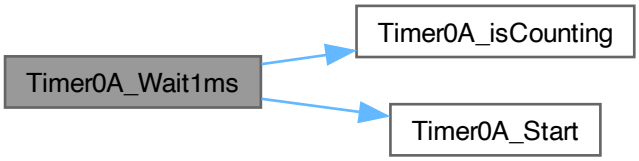
```
void Timer0A_Wait1ms (
            uint32_t time_ms )
```

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

Parameters

time_ms	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.

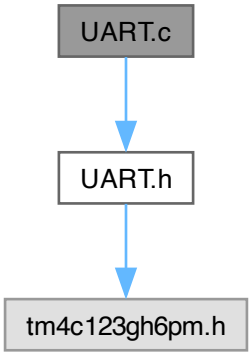
Parameters

time_ms	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

input_char	

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

UART0_WriteStr()

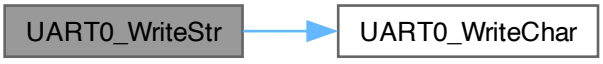
```
void UART0_WriteStr (
             unsigned char * str_ptr )
```

Write a C string to UART0.

Parameters

str_ptr	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:

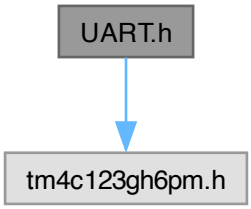


4.27 UART.h File Reference

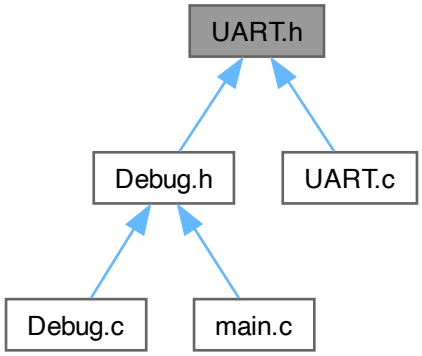
Driver module for UART1.

```
#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()

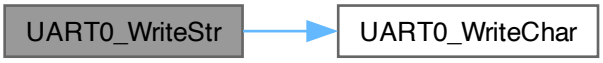
```
void UART0_WriteStr (
            unsigned char * str_ptr )
```

Write a C string to UART0.

Parameters

str_ptr	pointer to C string

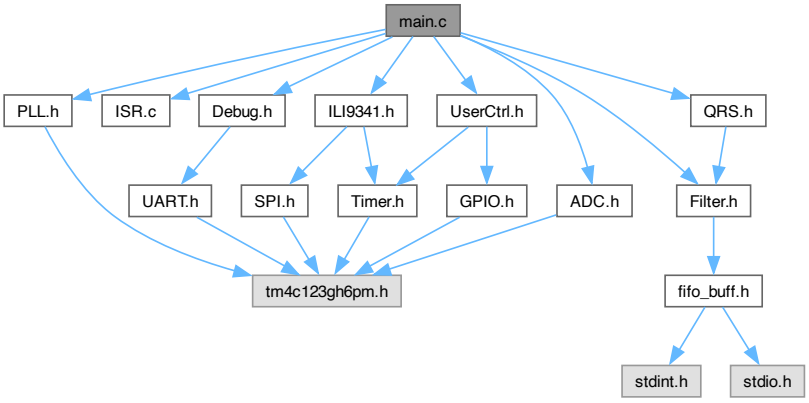
This function uses UART0_WriteChar() function to write a C string to UART0. The function writes until either the entire string has been written or a null-terminated character has been reached.Here is the call graph for this function:



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