ECG-HRM

Generated by Doxygen 1.9.7

1 Module Index	2
1.1 Modules	2
2 Data Structure Index	2
2.1 Data Structures	2
3 File Index	2
3.1 File List	2
4 Module Documentation	4
4.1 Device Drivers	4
4.1.1 Detailed Description	4
4.1.2 ADC br>	4
4.1.3 GPIO br>	5
4.1.4 ILI9341 	7
4.1.5 PLL 	14
4.1.6 SPI br>	14
4.1.7 SysTick 	16
4.1.8 Timer < br>	17
4.1.9 UART br>	21
4.2 Application Software	24
4.3 Program Threads	24
4.3.1 Detailed Description	25
4.3.2 Function Documentation	25
5 Data Structure Documentation	26
5.1 FIFO_buffer_t Struct Reference	26
5.1.1 Detailed Description	26
6 File Documentation	26
6.1 Debug.h File Reference	26
6.1.1 Detailed Description	27
6.2 Filter.h File Reference	27
6.2.1 Detailed Description	28
6.3 LCD.c File Reference	28
6.3.1 Detailed Description	29
6.3.2 Function Documentation	29
6.4 LCD.h File Reference	30
6.4.1 Detailed Description	31
6.4.2 Function Documentation	31
6.5 QRS.h File Reference	31
6.5.1 Detailed Description	32
6.6 UserCtrl.h File Reference	32
6.6.1 Detailed Description	33

6.6.2 Function Documentation	33
6.7 fifo_buff.c File Reference	33
6.7.1 Detailed Description	34
6.7.2 Function Documentation	34
6.8 fifo_buff.h File Reference	36
6.8.1 Detailed Description	37
6.8.2 Function Documentation	37
6.9 ADC.c File Reference	38
6.9.1 Detailed Description	39
6.10 ADC.h File Reference	39
6.10.1 Detailed Description	40
6.11 GPIO.h File Reference	40
6.11.1 Detailed Description	41
6.12 ILI9341.c File Reference	42
6.12.1 Detailed Description	43
6.13 ILI9341.h File Reference	43
6.13.1 Detailed Description	45
6.14 isr.c File Reference	46
6.14.1 Detailed Description	46
6.15 PLL.c File Reference	46
6.15.1 Detailed Description	47
6.16 PLL.h File Reference	47
6.16.1 Detailed Description	48
6.17 SPI.c File Reference	48
6.17.1 Detailed Description	49
6.18 SPI.h File Reference	49
6.18.1 Detailed Description	50
6.19 SysTick.c File Reference	51
6.19.1 Detailed Description	51
6.20 SysTick.h File Reference	51
6.20.1 Detailed Description	52
6.21 Timer.c File Reference	53
6.21.1 Detailed Description	53
6.22 Timer.h File Reference	54
6.22.1 Detailed Description	55
6.23 UART.c File Reference	55
6.23.1 Detailed Description	56
6.24 UART.h File Reference	56
6.24.1 Detailed Description	57
6.25 main.c File Reference	58
6.25.1 Detailed Description	58

Inde	ndex 59			
1	Module Index			
1.1	Modules			
Here	e is a list of all modules:			
I	Device Drivers	4		
	ADC br>	4		
	GPIO br>	5		
	ILI9341 <br< td=""><td>7</td></br<>	7		
	PLL br>	14		
	SPI br>	14		
	SysTick	16		
	Timer br>	17		
	UART	21		
1	Application Software	24		
I	Program Threads	24		
2	Data Structure Index			
2.1	Data Structures			
	e are the data structures with brief descriptions:			
	FIFO_buffer_t Array-based FIFO buffer type	26		
3	File Index			
3.1	File List			
Here	e is a list of all documented files with brief descriptions:			
ı	Debug.h Functions to output debugging information to a serial port via UART	26		

3.1 File List 3

Filter.h	
Functions to implement digital filters via linear constant coefficient difference equations (LC-CDEs)	27
LCD.c Source code for LCD module	28
LCD.h	
Module for outputting the ECG waveform and HR to a liquid crystal display (LCD)	30
QRS.h QRS detection algorithm functions	31
UserCtrl.h Interface for user control module	32
fifo_buff.c Source code file for FIFO buffer type	33
fifo_buff.h Header file for FIFO buffer type	36
ADC.c	38
ADC.h Driver module for analog-to-digital conversion (ADC)	39
GPIO.h Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts	40
ILI9341.c Source code for ILI9341 module	42
ILI9341.h Driver module for interfacing with an ILI9341 LCD driver	43
isr.c Source code for interrupt service routines (ISRs)	46
PLL.c Implementation details for phase-lock-loop (PLL) functions	46
PLL.h Driver module for activating the phase-locked-loop (PLL)	47
SPI.c Source code for SPI module	48
SPI.h Driver module for using the serial peripheral interface (SPI) protocol	49
SysTick.c Implementation details for SysTick functions	51
SysTick.h Driver module for using SysTick-based timing and/or interrupts	51
Timer.c Implementation for timer module	53

Timer.h	
Driver module for timing (Timer0) and interrupts (Timer1)	54
UART.c	
Source code for UART module	55
UART.h	
Driver module for serial communication via UART0 and UART 1	56
main.c	
Main program file for ECG-HRM	58

4 Module Documentation

4.1 Device Drivers

Device driver modules.

Modules

• ADC

Analog-to-digital conversion module.

• GPIO

GPIO Port F module.

ILI9341

Module for interfacing ILI9341-based RGB LCD via SPI.

• PLL

Phase-locked loop module.

• SPI

Serial peripheral interface module.

• SysTick

SysTick timing module.

• Timer

Timer0A module.

• UART

UART0 module.

4.1.1 Detailed Description

Device driver modules.

4.1.2 ADC

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

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()

Initialize GPIO Port F.

GPIO_PF_Interrupt_Init()

Initialize GPIO Port F interrupts via Sw1 and Sw2.

Here is the call graph for this function:



GPIO_PF_LED_Init()

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

Here is the call graph for this function:



GPIO_PF_LED_Toggle()

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()

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:



4.1.4 ILI9341

Module for interfacing ILI9341-based RGB LCD via SPI.

Files

• file ILI9341.h

Driver module for interfacing with an ILI9341 LCD driver.

Macros

- #define NOP (uint8 t) 0x00
- #define SWRESET (uint8_t) 0x01
- #define RDDST (uint8_t) 0x09
- #define RDDMADCTL (uint8 t) 0x0B
- #define RDDCOLMOD (uint8 t) 0x0C
- #define DINVOFF (uint8 t) 0x20
- #define DINVON (uint8_t) 0x21
- #define CASET (uint8 t) 0x2A
- #define PASET (uint8 t) 0x2B
- #define RAMWR (uint8 t) 0x2C
- #define RAMRD (uint8 t) 0x2E
- #define **DISPOFF** (uint8 t) 0x28
- #define **DISFOLL** (dinto_t) 0x20
- #define **DISPON** (uint8_t) 0x29
- #define **VSCRDEF** (uint8_t) 0x33
- #define MADCTL (uint8 t) 0x36
- #define VSCRSADD (uint8_t) 0x37
- #define PIXSET (uint8 t) 0x3A
- #define WRDISBV (uint8 t) 0x51
- #define RDDISBV (uint8 t) 0x52
- #define IFMODE (uint8_t) 0xB0
- #define FRMCTR1 (uint8 t) 0xB1
- #define PRCTR (uint8 t) 0xB5
- #define IFCTL (uint8 t) 0xF6
- #define NUM_COLS (uint8 t) 240
- #define NUM_ROWS (uint8_t) 320

Functions

void ILI9341 Init (void)

Initialize the LCD driver.

· void ILI9341 ResetHard (void)

Perform a hardware reset of the LCD driver.

void ILI9341_ResetSoft (void)

Perform a software reset of the LCD driver.

void ILI9341 NoOpCmd (void)

Send the "No Operation" command (NOP) to the LCD driver. Can be used to terminate the "Memory Write" and "Memory Read" commands (RAMWR and RAMRD, respectively), but does nothing otherwise.

- uint8 t * ILI9341_getDispStatus (void)
- uint8_t ILI9341_getMemAccessCtrl (void)
- void ILI9341_setRowAddress (uint16_t start_row, uint16_t end_row)

Sets the start/end rows to be written to.

void ILI9341_setColAddress (uint16_t start_col, uint16_t end_col)

Sets the start/end rows to be written to.

void ILI9341_writeMemCmd (void)

Sends the "Write Memory" command (RAMWR). Should be used before ILI9341_write1px().

void ILI9341_write1px (uint8_t data[3])

Write a single pixel to memory. Should be used after ILI9341_writeMemCmd().

void ILI9341_setDispInversion (uint8_t is_ON)

Send command to turn the display ON or OFF.

- void ILI9341_setDisplayStatus (uint8 t is ON)
- void ILI9341_setVertScrollArea (uint16_t top_fixed, uint16_t vert_scroll, uint16_t bottom_fixed)

- void ILI9341_setVertScrollStart (uint16_t start_address)
- void ILI9341_setMemAccessCtrl (uint8_t row_address_order, uint8_t col_address_order, uint8_t row_col
 —exchange, uint8_t vert_refresh_order, uint8_t rgb_order, uint8_t hor_refresh_order)
- void ILI9341_setPixelFormat (uint8 t is 16bit)
- void ILI9341_setDispBrightness (uint8 t brightness)
- uint8_t ILI9341_getDispBrightness (void)
- void ILI9341_setDispInterface (uint8_t param)

Send command to set operation status of display interface.

- void ILI9341_setFrameRate (uint8 t div ratio, uint8 t clocks per line)
- void **ILI9341_setBlankingPorch** (uint8_t vert_front_porch, uint8_t vert_back_porch, uint8_t hor_front_porch, uint8 t hor_back_porch)
- void ILI9341_setInterface (void)

Sets the interface for the ILI9341.

4.1.4.1 Detailed Description

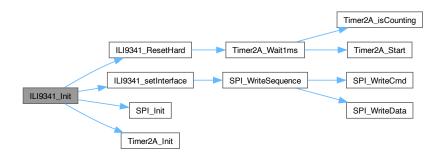
Module for interfacing ILI9341-based RGB LCD via SPI.

4.1.4.2 Function Documentation

ILI9341_Init()

Initialize the LCD driver.

This function initializes the SPI (i.e. SSI0) and Timer2A peripherals, initiates a hardware reset of the display, and tunes the display interface to allow blanking porch values to be manipulated via SPI commands. Here is the call graph for this function:

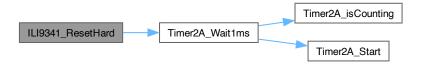


ILI9341_ResetHard()

Perform a hardware reset of the LCD driver.

The ILI9341's RESET signal requires a negative logic (i.e. active LOW) signal for \geq = 10 [us] and an additional 5 [ms] before further commands can be sent.

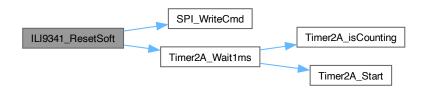
Here is the call graph for this function:



ILI9341_ResetSoft()

Perform a software reset of the LCD driver.

The ILI9341 requires an additional 5 [ms] before further commands can be sent after a reset. Here is the call graph for this function:



ILI9341_setColAddress()

Sets the start/end rows to be written to.

Parameters

start_col	<pre>0 <= start_col <= end_col</pre>
end_col	start_col <= end_col < 240

This function implements the "Column Address Set" command from p. 110 of the ILI9341 datasheet.

The input parameters represent the first and last columns to be written to when ILI9341_WriteMem() is called.

To work correctly, start_col must be no greater than end_col, and end_col cannot be greater than 239.Here is the call graph for this function:



ILI9341_setDispInterface()

Send command to set operation status of display interface.

Parameters



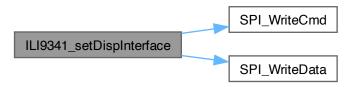
This function sets the display interface according to the following table, adapted from pg. 154 of the ILI9341 datasheet.

Bit	7	6	5	4	3	2	1	0
Value	ByPass_MODE	RCM[1]	RCM[0]	0	VSPL	HSPL	DPL	EPL
Default	0	1	0	0	0	0	0	1

Name	Description	0	1	Notes
ByPass_MODE	display data path	shift register	memory	N/A
RCM[1]	RGB interface select	N/A	N/A	Always 1
RCM[0]	RGB interface select	DE	SYNC	use SYNC to set blanking porch w/o DE bit/signal
VPSL	VSYNC polarity	low level	high level	N/A
HSPL	HSYNC polarity	low level	high level	N/A
DPL	DOTCLK polarity	rising edge	falling edge	when to fetch data relative to the dot clock

Name	Description	0	1	Notes
EPL	DE polarity	high enable	low enable	irrelevant in SYNC mode

Here is the call graph for this function:



ILI9341_setDispInversion()

```
void ILI9341_setDispInversion ( \label{eq:linear_setDispInversion} \text{ uint8\_t } is\_\textit{ON} \; )
```

Send command to turn the display ON or OFF.

Parameters

is ON	1 to turn ON, 0 to turn OFF
13_011	

Here is the call graph for this function:



ILI9341_setInterface()

Sets the interface for the ILI9341.

RGB Interface, 6-bit data transfer (3 transfer/pixel)Here is the call graph for this function:



ILI9341_setRowAddress()

Sets the start/end rows to be written to.

Parameters

start_row	<pre>0 <= start_row <= end_row</pre>
end_row	start_row <= end_row < 320

This function implements the "Page Address Set" command from p. 112 of the ILI9341 datasheet.

The input parameters represent the first and last rows to be written to when ILI9341_WriteMem() is called.

To work correctly, start_row must be no greater than end_row, and end_row cannot be greater than 319.Here is the call graph for this function:



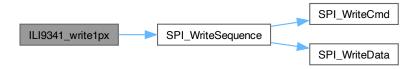
ILI9341_write1px()

Write a single pixel to memory. Should be used after ILI9341_writeMemCmd().

Parameters



Here is the call graph for this function:



4.1.5 PLL

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.

Functions

void SPI Init (void)

Initialize SSI0 to act as an SPI Controller (AKA Master) in mode 0.

uint8_t SPI_Read (void)

Read data from peripheral.

void SPI_WriteCmd (uint8_t cmd)

Write an 8-bit command to the peripheral.

void SPI_WriteData (uint8_t data)

Write 8-bit data to the peripheral.

void SPI_WriteSequence (uint8_t cmd, uint8_t param_sequence[], uint8_t num_params)

Write a sequence of data to the peripheral, with or without a preceding command.

4.1.6.1 Detailed Description

Serial peripheral interface module.

4.1.6.2 Function Documentation

SPI Init()

```
void SPI_Init (
     void )
```

Initialize SSI0 to act as an SPI Controller (AKA Master) in mode 0.

4.1.6.2.1 Pin | Function | ILI9341 Pin

PA2 | SSI0Clk | CLK | Serial clock signal PA3 | SSI0Fss | CS | Chip select signal PA4 | SSI0Rx | MISO | TM4C (M) input, LCD (S) output PA5 | SSI0Tx | MOSI | TM4C (M) output, LCD (S) input PA6 | GPIO | D/C | Data = 1, Command = 0 PA7 | GPIO | RESET | Reset the display (negative logic/active LOW)

Clk. Polarity = steady state low (0) Clk. Phase = rising clock edge (0) The bit rate BR is set using the clock prescale divisor CPSDVSR and SCR field in the SSI Control 0 (CR0) register:

```
fBR = f_{bus} / (CPSDVSR * (1 + SCR))
```

The ILI9341 driver has a minimum write cycle of 100 [ns], corresponding to a maximum serial clock frequency of 10 [MHz]. Thus, this function sets the bit rate BR to be the bus frequency ($f_{bus} = 80 \, [MHz]$) divided by 10 ($f_{bus} = 80 \, [MHz]$).

SPI_Read()

Read data from peripheral.

Returns

uint8_t

SPI_WriteCmd()

Write an 8-bit command to the peripheral.

Parameters

cmd	command for peripheral
-----	------------------------

SPI_WriteData()

Write 8-bit data to the peripheral.

Parameters

data input data for peripheral	
--------------------------------	--

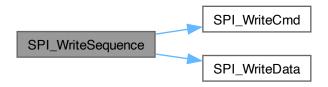
SPI_WriteSequence()

Write a sequence of data to the peripheral, with or without a preceding command.

Parameters

cmd	8-bit command (using cmd = 0 omits the command)	
param_sequence	sequence of parameters to send after cmd	
num_params	number of parameters to send; should be <= size of param_sequence	

Here is the call graph for this function:



4.1.7 SysTick

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()

Initialize SysTick for interrupts.

Parameters

time_ms Time in [ms] between interrupts. Cannot be more than 200[ms].

SysTick_Timer_Init()

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].

void Timer2A_Init (void)

Initialize timer 2 as 32-bit, one-shot, countdown timer.

void Timer2A_Start (uint32_t time_ms)

Count down starting from the inputted value.

uint8_t Timer2A_isCounting (void)

Returns 1 if Timer2 is still counting and 0 if not.

void Timer2A_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()

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

Returns

uint8_t status

Timer0A_Start()

Count down starting from the inputted value.

Parameters

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

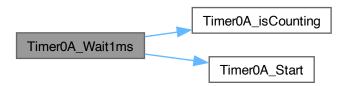
Timer0A_Wait1ms()

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

Parameters

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

Here is the call graph for this function:



Timer2A_Init()

```
void Timer2A_Init (
     void )
```

Initialize timer 2 as 32-bit, one-shot, countdown timer.

Timer2A_isCounting()

Returns 1 if Timer2 is still counting and 0 if not.

Returns

uint8_t status

Timer2A_Start()

Count down starting from the inputted value.

Parameters

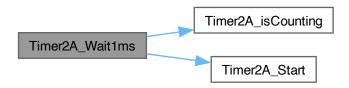
Timer2A_Wait1ms()

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

Parameters

time_ms	Time in [ms] to load into Timer 2. Must be \leq 53 seconds.
---------	---

Here is the call graph for this function:



4.1.9 UART

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 UARTO.

void UART0_WriteChar (unsigned char input_char)

Write a single character to UARTO.

void UART0_WriteStr (unsigned char *str_ptr)

Write a C string to UARTO.

void UART1_Init (void)

Initialize UART1 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.

• unsigned char UART1_ReadChar (void)

Read a single character from UART1.

void UART1_WriteChar (unsigned char input_char)

Write a single character to UART1.

void UART1_WriteStr (unsigned char *str_ptr)

Write a C string to UART1.

4.1.9.1 Detailed Description

UART0 module.

4.1.9.2 Function Documentation

UART0_Init()

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()

Read a single character from UART0.

Returns

input_char

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

UART0_WriteChar()

Write a single character to UART0.

Parameters

input_char

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

UART0_WriteStr()

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:



UART1_Init()

Initialize UART1 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 register

UART1_ReadChar()

```
unsigned char UART1_ReadChar ( \mbox{void} \ \ \mbox{)}
```

Read a single character from UART1.

Returns

input_char

This function uses busy-wait synchronization to read a character from UART1.

UART1_WriteChar()

Write a single character to UART1.

Parameters

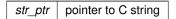
input_char

This function uses busy-wait synchronization to write a character to UART1.

UART1_WriteStr()

Write a C string to UART1.

Parameters



This function uses UART1_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 (void)

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()

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

Parameters

time ms	Time in [ms] between interrupts. Must be <= 53	S seconds
1 11110 1113		J SCOUIUS.

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

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.

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

• fifo_buff.c

6 File Documentation

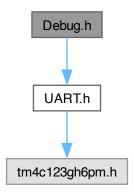
6.1 Debug.h File Reference

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

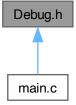
6.2 Filter.h File Reference 27

#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.

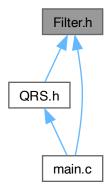
Author

Bryan McElvy

6.2 Filter.h File Reference

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

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

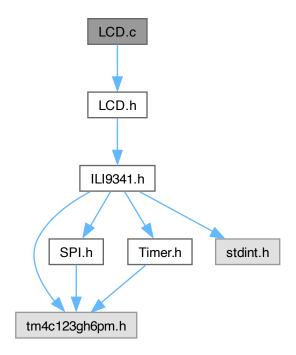
6.3 LCD.c File Reference

Source code for LCD module.

6.3 LCD.c File Reference 29

```
#include "LCD.h"
```

Include dependency graph for LCD.c:



Functions

void LCD_Init (void)

Initializes and configures the ILI9341 LCD driver.

6.3.1 Detailed Description

Source code for LCD module.

Author

Bryan McElvy

6.3.2 Function Documentation

LCD_Init()

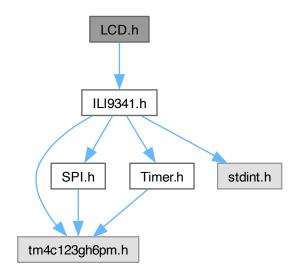
```
void LCD_Init (
     void )
```

Initializes and configures the ILI9341 LCD driver.

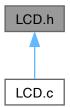
6.4 LCD.h File Reference

Module for outputting the ECG waveform and HR to a liquid crystal display (LCD).

#include "ILI9341.h"
Include dependency graph for LCD.h:



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



Functions

• void LCD_Init (void)

Initializes and configures the ILI9341 LCD driver.

6.5 QRS.h File Reference 31

6.4.1 Detailed Description

Module for outputting the ECG waveform and HR to a liquid crystal display (LCD).

Author

Bryan McElvy

6.4.2 Function Documentation

LCD_Init()

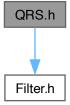
```
void LCD_Init (
     void )
```

Initializes and configures the ILI9341 LCD driver.

6.5 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:



6.5.1 Detailed Description

QRS detection algorithm functions.

Author

Bryan McElvy

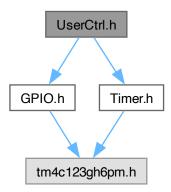
This module contains functions for detecting heart rate (HR) using a simplified version of the Pan-Tompkins algorithm.

6.6 UserCtrl.h File Reference

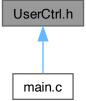
Interface for user control module.

```
#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)

6.6.1 Detailed Description

Interface for user control module.

Author

Bryan McElvy

6.6.2 Function Documentation

UserCtrl_Init()

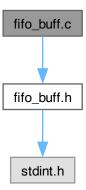
```
void UserCtrl_Init ( )
```

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

6.7 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.

6.7.1 Detailed Description

Source code file for FIFO buffer type.

Author

Bryan McElvy

6.7.2 Function Documentation

FIFO_add_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()

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

Parameters

FIFO_ptr	pointer to FIFO buffer
----------	------------------------

Returns

curr_size

FIFO_init()

Initializes a FIFO buffer with the specified size.

Parameters

buffer size desired buffer size

Returns

FIFO_buffer

FIFO_rem_sample()

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()

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

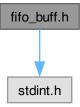
Parameters

FIFO_ptr pointer to FIFO bu

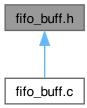
6.8 fifo_buff.h File Reference

Header file for FIFO buffer type.

#include <stdint.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.

6.8.1 Detailed Description

Header file for FIFO buffer type.

Author

Bryan McElvy

6.8.2 Function Documentation

FIFO_add_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()

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

Parameters

```
FIFO_ptr | pointer to FIFO buffer
```

Returns

curr_size

FIFO_init()

Initializes a FIFO buffer with the specified size.

Parameters

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

Returns

FIFO_buffer

FIFO_rem_sample()

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()

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

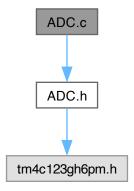
Parameters

```
FIFO_ptr | pointer to FIFO buffer
```

6.9 ADC.c File Reference

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

Include dependency graph for ADC.c:



6.9.1 Detailed Description

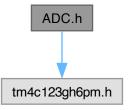
Author

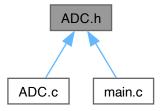
Bryan McElvy

6.10 ADC.h File Reference

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

#include "tm4c123gh6pm.h"
Include dependency graph for ADC.h:





6.10.1 Detailed Description

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

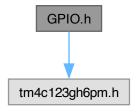
Author

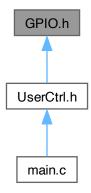
Bryan McElvy

6.11 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:





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.

6.11.1 Detailed Description

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

Author

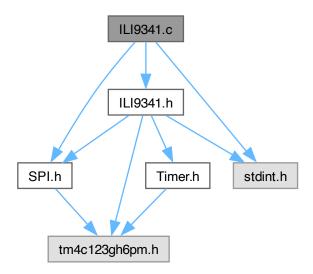
Bryan McElvy

6.12 ILI9341.c File Reference

Source code for ILI9341 module.

```
#include "ILI9341.h"
#include "SPI.h"
#include <stdint.h>
```

Include dependency graph for ILI9341.c:



Functions

void ILI9341_Init (void)

Initialize the LCD driver.

void ILI9341_ResetHard (void)

Perform a hardware reset of the LCD driver.

void ILI9341 ResetSoft (void)

Perform a software reset of the LCD driver.

• void ILI9341_NoOpCmd (void)

Send the "No Operation" command (NOP) to the LCD driver. Can be used to terminate the "Memory Write" and "Memory Read" commands (RAMWR and RAMRD, respectively), but does nothing otherwise.

- uint8 t * ILI9341 getDispStatus (void)
- uint8_t ILI9341_getMemAccessCtrl (void)
- void ILI9341_setRowAddress (uint16_t start_row, uint16_t end_row)

Sets the start/end rows to be written to.

• void ILI9341 setColAddress (uint16 t start col, uint16 t end col)

Sets the start/end rows to be written to.

void ILI9341_writeMemCmd (void)

Sends the "Write Memory" command (RAMWR). Should be used before ${\tt ILI9341_write1px}$ ().

void ILI9341_write1px (uint8_t data[3])

Write a single pixel to memory. Should be used after ILI9341_writeMemCmd().

- void ILI9341_setDispInversion (uint8_t is_ON)
 - Send command to turn the display ON or OFF.
- void ILI9341_setDisplayStatus (uint8_t is_ON)
- void ILI9341 setVertScrollArea (uint16 t top fixed, uint16 t vert scroll, uint16 t bottom fixed)
- void ILI9341_setVertScrollStart (uint16_t start_address)
- void ILI9341_setMemAccessCtrl (uint8_t row_address_order, uint8_t col_address_order, uint8_t row_col
 —exchange, uint8_t vert_refresh_order, uint8_t rgb_order, uint8_t hor_refresh_order)
- void ILI9341_setPixelFormat (uint8_t is_16bit)
- void ILI9341_setDispBrightness (uint8_t brightness)
- uint8 t ILI9341_getDispBrightness (void)
- void ILI9341_setDispInterface (uint8_t param)

Send command to set operation status of display interface.

- void ILI9341_setFrameRate (uint8_t div_ratio, uint8_t clocks_per_line)
- void **ILI9341_setBlankingPorch** (uint8_t vert_front_porch, uint8_t vert_back_porch, uint8_t hor_front_porch, uint8_t hor_back_porch)
- void ILI9341 setInterface (void)

Sets the interface for the ILI9341.

6.12.1 Detailed Description

Source code for ILI9341 module.

Author

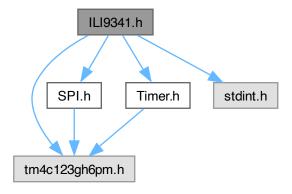
Bryan McElvy

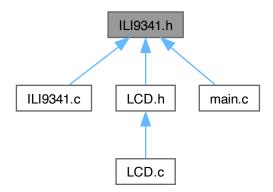
6.13 ILI9341.h File Reference

Driver module for interfacing with an ILI9341 LCD driver.

```
#include "tm4c123gh6pm.h"
#include "SPI.h"
#include "Timer.h"
#include <stdint.h>
```

Include dependency graph for ILI9341.h:





Macros

- #define NOP (uint8_t) 0x00
- #define SWRESET (uint8 t) 0x01
- #define RDDST (uint8_t) 0x09
- #define RDDMADCTL (uint8 t) 0x0B
- #define RDDCOLMOD (uint8_t) 0x0C
- #define **DINVOFF** (uint8_t) 0x20
- #define DINVON (uint8 t) 0x21
- #define CASET (uint8_t) 0x2A
- #define PASET (uint8_t) 0x2B
- #define RAMWR (uint8_t) 0x2C
- #define RAMRD (uint8_t) 0x2E
- #define **DISPOFF** (uint8_t) 0x28
- #define **DISPON** (uint8_t) 0x29
- #define VSCRDEF (uint8_t) 0x33
- #define MADCTL (uint8_t) 0x36
- #define VSCRSADD (uint8_t) 0x37
- #define PIXSET (uint8_t) 0x3A
- #define WRDISBV (uint8 t) 0x51
- #define **RDDISBV** (uint8_t) 0x52
- #define IFMODE (uint8 t) 0xB0
- #define FRMCTR1 (uint8_t) 0xB1
- #define PRCTR (uint8_t) 0xB5
- #define IFCTL (uint8_t) 0xF6
- #define NUM_COLS (uint8_t) 240
- #define NUM_ROWS (uint8 t) 320

Functions

void ILI9341_Init (void)

Initialize the LCD driver.

· void ILI9341_ResetHard (void)

Perform a hardware reset of the LCD driver.

void ILI9341_ResetSoft (void)

Perform a software reset of the LCD driver.

void ILI9341_NoOpCmd (void)

Send the "No Operation" command (NOP) to the LCD driver. Can be used to terminate the "Memory Write" and "Memory Read" commands (RAMWR and RAMRD, respectively), but does nothing otherwise.

- uint8 t * ILI9341_getDispStatus (void)
- uint8 t ILI9341_getMemAccessCtrl (void)
- void ILI9341_setRowAddress (uint16_t start_row, uint16_t end_row)

Sets the start/end rows to be written to.

void ILI9341_setColAddress (uint16_t start_col, uint16_t end_col)

Sets the start/end rows to be written to.

void ILI9341_writeMemCmd (void)

Sends the "Write Memory" command (RAMWR). Should be used before ILI9341_write1px().

void ILI9341 write1px (uint8 t data[3])

Write a single pixel to memory. Should be used after ILI9341_writeMemCmd().

void ILI9341_setDispInversion (uint8_t is_ON)

Send command to turn the display ON or OFF.

- void ILI9341 setDisplayStatus (uint8 t is ON)
- · void ILI9341 setVertScrollArea (uint16 t top fixed, uint16 t vert scroll, uint16 t bottom fixed)
- void ILI9341_setVertScrollStart (uint16_t start_address)
- void ILI9341_setMemAccessCtrl (uint8_t row_address_order, uint8_t col_address_order, uint8_t row_col
 —exchange, uint8_t vert_refresh_order, uint8_t rgb_order, uint8_t hor_refresh_order)
- void ILI9341_setPixelFormat (uint8_t is_16bit)
- void ILI9341_setDispBrightness (uint8 t brightness)
- uint8 t ILI9341_getDispBrightness (void)
- void ILI9341_setDispInterface (uint8_t param)

Send command to set operation status of display interface.

- void ILI9341 setFrameRate (uint8 t div ratio, uint8 t clocks per line)
- void ILI9341_setBlankingPorch (uint8_t vert_front_porch, uint8_t vert_back_porch, uint8_t hor_front_porch, uint8_t hor_back_porch)
- void ILI9341_setInterface (void)

Sets the interface for the ILI9341.

6.13.1 Detailed Description

Driver module for interfacing with an ILI9341 LCD driver.

Author

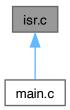
Bryan McElvy

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.

6.14 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].

6.14.1 Detailed Description

Source code for interrupt service routines (ISRs)

Author

Bryan McElvy

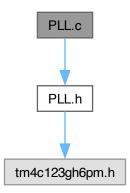
6.15 PLL.c File Reference

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

6.16 PLL.h File Reference 47

#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].

6.15.1 Detailed Description

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

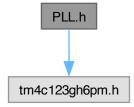
Author

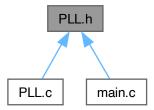
Bryan McElvy

6.16 PLL.h File Reference

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

#include "tm4c123gh6pm.h"
Include dependency graph for PLL.h:





Functions

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

6.16.1 Detailed Description

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

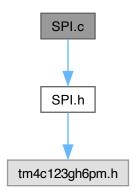
Author

Bryan McElvy

6.17 SPI.c File Reference

Source code for SPI module.

#include "SPI.h"
Include dependency graph for SPI.c:



6.18 SPI.h File Reference 49

Functions

• void SPI_Init (void)

Initialize SSI0 to act as an SPI Controller (AKA Master) in mode 0.

• uint8_t SPI_Read (void)

Read data from peripheral.

• void SPI_WriteCmd (uint8_t cmd)

Write an 8-bit command to the peripheral.

• void SPI_WriteData (uint8_t data)

Write 8-bit data to the peripheral.

• void SPI_WriteSequence (uint8_t cmd, uint8_t param_sequence[], uint8_t num_params)

Write a sequence of data to the peripheral, with or without a preceding command.

6.17.1 Detailed Description

Source code for SPI module.

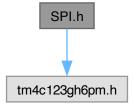
Author

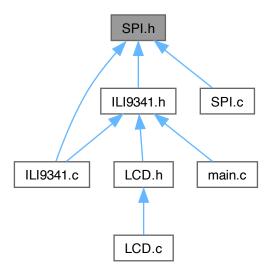
Bryan McElvy

6.18 SPI.h File Reference

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

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





Functions

• void SPI_Init (void)

Initialize SSI0 to act as an SPI Controller (AKA Master) in mode 0.

• uint8_t SPI_Read (void)

Read data from peripheral.

void SPI_WriteCmd (uint8_t cmd)

Write an 8-bit command to the peripheral.

void SPI_WriteData (uint8_t data)

Write 8-bit data to the peripheral.

• void SPI_WriteSequence (uint8_t cmd, uint8_t param_sequence[], uint8_t num_params)

Write a sequence of data to the peripheral, with or without a preceding command.

6.18.1 Detailed Description

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

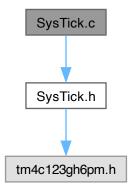
Author

Bryan McElvy

6.19 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.

6.19.1 Detailed Description

Implementation details for SysTick functions.

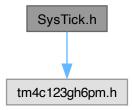
Author

Bryan McElvy

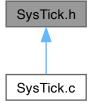
6.20 SysTick.h File Reference

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.

6.20.1 Detailed Description

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

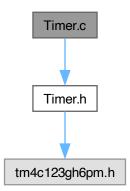
Author

Bryan McElvy

6.21 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.

void Timer2A_Init (void)

Initialize timer 2 as 32-bit, one-shot, countdown timer.

void Timer2A_Start (uint32_t time_ms)

Count down starting from the inputted value.

uint8_t Timer2A_isCounting (void)

Returns 1 if Timer2 is still counting and 0 if not.

void Timer2A_Wait1ms (uint32_t time_ms)

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

6.21.1 Detailed Description

Implementation for timer module.

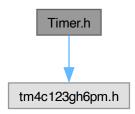
Author

Bryan McElvy

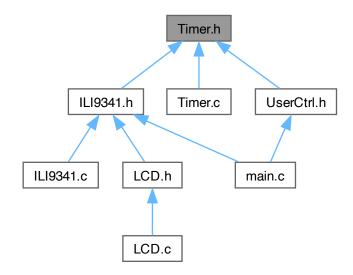
6.22 Timer.h File Reference

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.

void Timer2A_Init (void)

Initialize timer 2 as 32-bit, one-shot, countdown timer.

void Timer2A_Start (uint32_t time_ms)

Count down starting from the inputted value.

• uint8_t Timer2A_isCounting (void)

Returns 1 if Timer2 is still counting and 0 if not.

void Timer2A_Wait1ms (uint32_t time_ms)

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

6.22.1 Detailed Description

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

Author

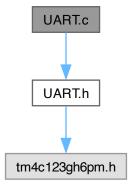
Bryan McElvy

6.23 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 UARTO.

• void UART0_WriteChar (unsigned char input_char)

Write a single character to UARTO.

void UART0_WriteStr (unsigned char *str_ptr)

Write a C string to UARTO.

void UART1_Init (void)

Initialize UART1 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.

• unsigned char UART1_ReadChar (void)

Read a single character from UART1.

void UART1_WriteChar (unsigned char input_char)

Write a single character to UART1.

• void UART1_WriteStr (unsigned char *str_ptr)

Write a C string to UART1.

6.23.1 Detailed Description

Source code for UART module.

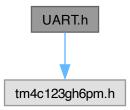
Author

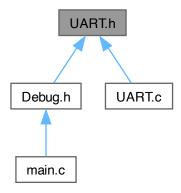
Bryan McElvy

6.24 UART.h File Reference

Driver module for serial communication via UART0 and UART 1.

#include "tm4c123gh6pm.h"
Include dependency graph for UART.h:





Functions

• void UARTO_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 UARTO.

void UART0_WriteChar (unsigned char input_char)

Write a single character to UARTO.

void UART0_WriteStr (unsigned char *str_ptr)

Write a C string to UARTO.

void UART1_Init (void)

Initialize UART1 to a baud rate of 115200, 8-bit data length, 1 start bit, and 1 stop bit.

unsigned char UART1_ReadChar (void)

Read a single character from UART1.

· void UART1_WriteChar (unsigned char input_char)

Write a single character to UART1.

void UART1_WriteStr (unsigned char *str_ptr)

Write a C string to UART1.

6.24.1 Detailed Description

Driver module for serial communication via UART0 and UART 1.

Author

Bryan McElvy

UARTO uses PAO and PAI, which are not broken out but do connect to a PC's serial port via USB.

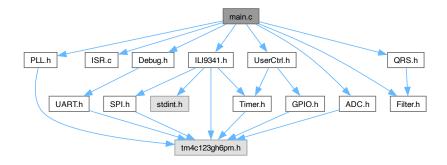
UART1 uses PB0 (Rx) and PB1 (Tx), which are not broken out but do not connect to a serial port.

6.25 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 (void)

6.25.1 Detailed Description

Main program file for ECG-HRM.

Author

Bryan McElvy

Index

ADC br>, 4	GPIO_PF_Sw_Init
ADC.c, 38	GPIO , 7
ADC.h, 39	GPIO PortF Handler
Application Software, 24	Program Threads, 25
, pp	
Debug.h, 26	ILI9341 , 7
Device Drivers, 4	ILI9341_Init, 9
	ILI9341_ResetHard, 9
FIFO_add_sample	ILI9341_ResetSoft, 10
fifo_buff.c, 34	ILI9341_setColAddress, 10
fifo_buff.h, 37	ILI9341_setDispInterface, 11
fifo_buff.c, 33	ILI9341 setDispInversion, 12
FIFO_add_sample, 34	ILI9341_setInterface, 12
FIFO_get_size, 34	ILI9341_setRowAddress, 13
FIFO_init, 35	ILI9341_write1px, 13
FIFO_rem_sample, 35	IL19341_write rpx, 13
FIFO_show_data, 35	
fifo_buff.h, 36	ILI9341.h, 43
FIFO_add_sample, 37	ILI9341_Init
·	ILI9341 br>, 9
FIFO_get_size, 37	ILI9341_ResetHard
FIFO_init, 37	ILI9341 br>, 9
FIFO_rem_sample, 38	ILI9341_ResetSoft
FIFO_show_data, 38	ILI9341 , 10
FIFO_buffer_t, 26	ILI9341_setColAddress
FIFO_get_size	ILI9341 , 10
fifo_buff.c, 34	ILI9341_setDispInterface
fifo_buff.h, 37	ILI9341 , 11
FIFO_init	ILI9341_setDispInversion
fifo_buff.c, 35	ILI9341 br>, 12
fifo_buff.h, 37	ILI9341_setInterface
FIFO_rem_sample	
fifo_buff.c, 35	ILI9341_setRowAddress
fifo_buff.h, 38	
FIFO_show_data	ILI9341_write1px
fifo_buff.c, 35	ILI9341 br>, 13
fifo buff.h, 38	isr.c, 46
Filter.h, 27	101.0, 10
,	LCD.c, 28
GPIO br>, 5	LCD_Init, 29
GPIO_PF_Init, 5	LCD.h, 30
GPIO_PF_Interrupt_Init, 5	LCD_Init, 31
GPIO PF LED Init, 6	LCD_Init
GPIO_PF_LED_Toggle, 6	LCD.c, 29
GPIO PF LED Write, 6	LCD.h, 31
GPIO_PF_Sw_Init, 7	LOD.II, 31
GPIO.h, 40	main.c, 58
GPIO PF Init	mam.e, 90
GPIO Spront 5 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	PLL , 14
	PLL Init, 14
GPIO_PF_Interrupt_Init	PLL.c, 46
GPIO OBE LED Init	PLL.h, 47
GPIO_PF_LED_Init	PLL Init
GPIO GPIO ADIO ADIO 	
GPIO_PF_LED_Toggle	PLL br>, 14
GPIO dr>, 6	Program Threads, 24
GPIO_PF_LED_Write	GPIO_PortF_Handler, 25
GPIO < br >, 6	SysTick_Handler, 25

60 INDEX

Timer1A_Handler, 25 Timer1A_Init, 25	Timer , 19 Timer2A_isCounting
QRS.h, 31	Timer br>, 19 Timer2A_Start
CDI chrs 14	Timer br>, 19
SPI SPI Init 15	Timer2A_Wait1ms
SPI_Init, 15	Timer br>, 21
SPI_Read, 15	
SPI_WriteCmd, 15	UART , 21
SPI_WriteData, 16	UART0_Init, 22
SPI_WriteSequence, 16	UART0_ReadChar, 22
SPI.c, 48	UART0_WriteChar, 22
SPI.h, 49	UART0_WriteStr, 22
SPI_Init	UART1 Init, 23
SPI br>, 15	UART1_ReadChar, 23
SPI_Read	UART1_WriteChar, 23
SPI br>, 15	UART1 WriteStr, 24
SPI WriteCmd	UART.c, 55
	UART.h, 56
SPI WriteData	UART0_Init
SPI br>, 16	
SPI_WriteSequence	UART br>, 22
SPI SPI	UARTO_ReadChar
	UART br>, 22
SysTick SysTick laterwest lait 17	UART0_WriteChar
SysTick_Interrupt_Init, 17	UART , 22
SysTick_Timer_Init, 17	UART0_WriteStr
SysTick.c, 51	UART , 22
SysTick.h, 51	UART1_Init
SysTick_Handler	UART , 23
Program Threads, 25	UART1_ReadChar
SysTick_Interrupt_Init	UART , 23
SysTick , 17	UART1_WriteChar
SysTick_Timer_Init	UART , 23
SysTick , 17	UART1 WriteStr
Timer br>, 17	UserCtrl.h, 32
Timer0A_Init, 18	UserCtrl Init, 33
Timer0A_isCounting, 18	UserCtrl_Init
Timer0A_Start, 18	UserCtrl.h, 33
Timer0A_Wait1ms, 19	3301311111, 33
Timer2A_Init, 19	
Timer2A_isCounting, 19	
Timer2A_Start, 19	
Timer2A Wait1ms, 21	
Timer.c, 53	
Timer.h, 54	
Timer0A Init	
Timer br>, 18	
Timer0A_isCounting	
Timer try, 18	
Timer0A_Start	
Timer br>, 18	
TimerOA_Wait1ms	
Timer < br > , 19	
Timer1A_Handler	
Program Threads, 25	
Timer1A_Init	
Program Threads, 25	
Timer2A_Init	