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 br>	7
4.1.5 PLL 	17
4.1.6 SPI br>	17
4.1.7 SysTick 	19
4.1.8 Timer 	20
4.1.9 UART br>	23
4.2 Application Software	26
4.3 Program Threads	26
4.3.1 Detailed Description	27
4.3.2 Function Documentation	27
5 Data Structure Documentation	28
5.1 FIFO_buffer_t Struct Reference	28
5.1.1 Detailed Description	28
5.2 LCD_t Struct Reference	28
6 File Documentation	29
6.1 Debug.h File Reference	29
6.1.1 Detailed Description	29
6.2 Filter.h File Reference	30
6.2.1 Detailed Description	30
6.3 LCD.c File Reference	30
6.3.1 Detailed Description	32
6.3.2 Function Documentation	32
6.4 LCD.h File Reference	37
6.4.1 Detailed Description	39
·	39
	44
	45
	45

6.6.1 Detailed Description	46
6.6.2 Function Documentation	46
6.7 fifo_buff.c File Reference	46
6.7.1 Detailed Description	47
6.7.2 Function Documentation	47
6.8 fifo_buff.h File Reference	49
6.8.1 Detailed Description	50
6.8.2 Function Documentation	50
6.9 ADC.c File Reference	51
6.9.1 Detailed Description	52
6.10 ADC.h File Reference	52
6.10.1 Detailed Description	53
6.11 GPIO.h File Reference	53
6.11.1 Detailed Description	54
6.12 ILI9341.c File Reference	55
6.12.1 Detailed Description	57
6.13 ILI9341.h File Reference	57
6.13.1 Detailed Description	59
6.14 isr.c File Reference	59
6.14.1 Detailed Description	59
6.15 PLL.c File Reference	59
6.15.1 Detailed Description	60
6.16 PLL.h File Reference	60
6.16.1 Detailed Description	61
6.17 SPI.c File Reference	61
6.17.1 Detailed Description	62
6.18 SPI.h File Reference	62
6.18.1 Detailed Description	63
6.19 SysTick.c File Reference	63
6.19.1 Detailed Description	64
6.20 SysTick.h File Reference	64
6.20.1 Detailed Description	65
6.21 Timer.c File Reference	66
6.21.1 Detailed Description	66
6.22 Timer.h File Reference	67
6.22.1 Detailed Description	68
6.23 UART.c File Reference	68
6.23.1 Detailed Description	69
6.24 UART.h File Reference	69
6.24.1 Detailed Description	70
6.25 main.c File Reference	71
6.25.1 Detailed Description	71

Ind	ndex 73			
1	Module Index			
1.1	Modules			
Her	re is a list of all modules:			
	Device Drivers	4		
	ADC br>	4		
	GPIO br>	5		
	IL19341	7		
	PLL	17		
	SPI br>	17		
	SysTick <br< td=""><td>19</td></br<>	19		
	Timer Timer Timer Timer Timer Timer Timer Timer Timer	20		
	UART br>	23		
	Application Software	26		
	Program Threads	26		
2	Data Structure Index			
2.1	Data Structures			
Her	re are the data structures with brief descriptions:			
	FIFO_buffer_t Array-based FIFO buffer type	28		
	LCD_t	28		
3	File Index			
3.1	File List			
Her	re is a list of all documented files with brief descriptions:			
	Debug.h Functions to output debugging information to a serial port via UART	29		

3.1 File List 3

Filter.h	
Functions to implement digital filters via linear constant coefficient difference equations (LC-CDEs)	30
LCD.c Source code for LCD module	30
LCD.h	
Module for outputting the ECG waveform and HR to a liquid crystal display (LCD)	37
QRS.h QRS detection algorithm functions	44
UserCtrl.h Interface for user control module	45
fifo_buff.c Source code file for FIFO buffer type	46
fifo_buff.h Header file for FIFO buffer type	49
ADC.c	51
ADC.h Driver module for analog-to-digital conversion (ADC)	52
GPIO.h Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts	53
ILI9341.c Source code for ILI9341 module	55
ILI9341.h Driver module for interfacing with an ILI9341 LCD driver	57
isr.c Source code for interrupt service routines (ISRs)	59
PLL.c Implementation details for phase-lock-loop (PLL) functions	59
PLL.h Driver module for activating the phase-locked-loop (PLL)	60
SPI.c Source code for SPI module	61
SPI.h Driver module for using the serial peripheral interface (SPI) protocol	62
SysTick.c Implementation details for SysTick functions	63
SysTick.h Driver module for using SysTick-based timing and/or interrupts	64
Timer.c Implementation for timer module	66

Timer.h	
Driver module for timing (Timer0) and interrupts (Timer1)	67
UART.c	
Source code for UART module	68
UART.h	
Driver module for serial communication via UART0 and UART 1	69
main.c	
Main program file for ECG-HRM	71

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 <br

Analog-to-digital conversion module.

Files

• file ADC.h

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

4.1.2.1 Detailed Description

Analog-to-digital conversion module.

4.1.3 GPIO

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 NUM_COLS (uint16_t) 240
- #define NUM_ROWS (uint16_t) 320

Functions

void ILI9341_Init (void)

Initialize the LCD driver, the SPI module, and Timer2A.

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 setSleepMode (bool is sleeping)

Enter or exit sleep mode. The LCD driver is in sleep mode by default upon powering on or either kind of reset.

void ILI9341 setDispMode (bool is normal)

Set the display to normal mode or partial mode. The LCD driver starts out in normal mode. Calling with either possible value exits scrolling mode.

void ILI9341_setPartialArea (uint16_t rowStart, uint16_t rowEnd)

Set the partial display area for partial mode. Call before activating partial mode via ILI9341_setDisplayMode().

void ILI9341 setDispInversion (bool is ON)

Toggle display inversion. Turning ON causes colors to be inverted on the display.

void ILI9341 setDispOutput (bool is ON)

Turn display output ON or OFF. This function clears the display and stops outputting to the display area, but does not affect frame memory or power.

• void ILI9341_setVertScrollArea (uint16_t top_fixed, uint16_t vert_scroll, uint16_t bottom_fixed)

TODO: Write.

void ILI9341_setVertScrollStart (uint16 t start address)

TODO: Write.

 void ILI9341_setMemAccessCtrl (bool areRowsFlipped, bool areColsFlipped, bool areRowsColsSwitched, bool isVertRefreshFlipped, bool isColorOrderFlipped, bool isHorRefreshFlipped)

Set how data is converted from memory to display.

void ILI9341 setColorDepth (bool is 16bit)

Set the pixel format to be 16-bit (65K colors) or 18-bit (262K colors).

void ILI9341_NoOpCmd (void)

Send the "No Operation" command (NOP = 0×00) to the LCD driver. Can be used to terminate the "Memory Write" (RAMWR) and "Memory Read" (RAMRD) commands, but does nothing otherwise.

• void ILI9341 setFrameRate (uint8 t div ratio, uint8 t clocks per line)

TODO: Write.

void ILI9341_setBlankingPorch (uint8_t vpf, uint8_t vbp, uint8_t hfp, uint8_t hbp)

TODO: Write.

void ILI9341_setInterface (void)

Sets the interface for the ILl9341. The parameters for this command are hard-coded, so it only needs to be called once upon initialization.

void ILI9341_setRowAddress (uint16_t start_row, uint16_t end_row)

not using backlight, so these aren't necessary

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" (RAMWR) command to the LCD driver, signalling that incoming data should be written to memory.

• void ILI9341 write1px (uint8 t red, uint8 t green, uint8 t blue, bool is 16bit)

Write a single pixel to frame memory.

4.1.4.1 Detailed Description

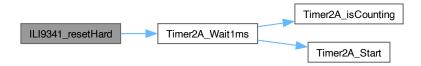
Module for interfacing ILI9341-based RGB LCD via SPI.

4.1.4.2 Function Documentation

ILI9341_resetHard()

Perform a hardware reset of the LCD driver.

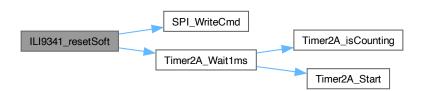
The LCD driver's RESET pin requires a negative logic (i.e. active LOW) signal for >= 10 [us] and an additional 5 [ms] before further commands can be sent. Here is the call graph for this function:



ILI9341_resetSoft()

Perform a software reset of the LCD driver.

the driver needs 5 $\left[\text{ms}\right]$ before another commandHere is the call graph for this function:



ILI9341_setBlankingPorch()

```
void ILI9341_setBlankingPorch (
          uint8_t vpf,
          uint8_t vbp,
          uint8_t hfp,
          uint8_t hbp )
```

TODO: Write.

TODO: Write

ILI9341_setColAddress()

Sets the start/end rows to be written to.

```
Should be called along with 'ILI9341_setRowAddress()' and before 'ILI9341_writeMemCmd()'.
```

Parameters

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

This function is simply an interface to ILI9341_setAddress(). To work correctly, start_col must be no greater than end_col, and end_col cannot be greater than the max column number (default 240).

ILI9341_setColorDepth()

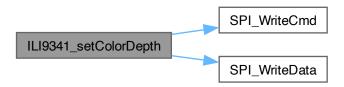
```
void ILI9341_setColorDepth ( bool is\_16bit )
```

Set the pixel format to be 16-bit (65K colors) or 18-bit (262K colors).

Parameters

```
is_16bit
```

16-bit requires 2 transfers and allows for 65K colors. 18-bit requires 3 transfers and allows for 262K colors. Here is the call graph for this function:



ILI9341_setDispInversion()

```
void ILI9341_setDispInversion ( bool is\_ON )
```

Toggle display inversion. Turning ON causes colors to be inverted on the display.

Parameters

```
is_ON true to turn ON, false to turn OFF
```

TODO: Write descriptionHere is the call graph for this function:



ILI9341_setDispMode()

```
void ILI9341_setDispMode (
                bool is_normal )
```

Set the display to normal mode or partial mode. The LCD driver starts out in normal mode. Calling with either possible value exits scrolling mode.

Parameters

is_normal	'true' for normal mode, 'false' for partial mode
-----------	--

Here is the call graph for this function:



ILI9341_setDispOutput()

```
void ILI9341_setDispOutput ( bool is\_ON )
```

Turn display output ON or OFF. This function clears the display and stops outputting to the display area, but does not affect frame memory or power.

Parameters

```
is_ON true to turn ON, false to turn OFF
```

TODO: Write descriptionHere is the call graph for this function:



ILI9341_setFrameRate()

TODO: Write.

TODO: Write

ILI9341_setInterface()

Sets the interface for the ILI9341. The parameters for this command are hard-coded, so it only needs to be called once upon initialization.

This function implements the "Interface Control" IFCTL command from p. 192-194 of the ILI9341 datasheet, which controls how the LCD driver handles 16-bit data and what interfaces (internal or external) are used.

Name	Bit #	Param #	Effect when set = 1
MY_EOR	7		flips value of corresponding MADCTL bit
MX_EOR	6		flips value of corresponding MADCTL bit
MV_EOR	5	0	flips value of corresponding MADCTL bit
BGR_EOR	3	- 1 -	flips value of corresponding MADCTL bit
WEMODE	0		overflowing pixel data is not ignored
EPF[1:0]	5:4		controls 16 to 18-bit pixel data conversion
MDT[1:0]	1:0		controls display data transfer method
ENDIAN	5		host sends LSB first
DM[1:0]	3:2	2	selects display operation mode
RM	1		selects GRAM interface mode
RIM	0		specifies RGB interface-specific details

The first param's bits are cleared so that the corresponding MADCTL bits (ILI9341_setMemoryAccessCtrl()) are unaffected and overflowing pixel data is ignored. The EPF bits are cleared so that the LSB of the R and B values is copied from the MSB when using 16-bit color depth. The TM4C123 sends the MSB first, so the ENDIAN bit is cleared. The other bits are cleared and/or irrelevant since the RGB and VSYNC interfaces aren't used.Here is the call graph for this function:



ILI9341_setMemAccessCtrl()

Set how data is converted from memory to display.

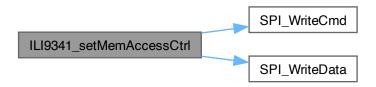
Parameters

areRowsFlipped	
areColsFlipped	
areRowsColsSwitched	
isVertRefreshFlipped	
isColorOrderFlipped	
isHorRefreshFlipped	

This function implements the "Memory Access Control" (MADCTL) command from p. 127-128 of the ILI9341 datasheet, which controls how the LCD driver displays data upon writing to memory.

Name	Bit #	Effect when set = 1	
MY	7 flip row (AKA "page") addresses		
MX	6	6 flip column addresses	
MV	5	exchange rows and column addresses	
ML	4	reverse horizontal refresh order	
BGR	3	reverse color input order (RGB -> BGR)	
MH	2 reverse vertical refresh order		

All bits are clear after powering on or HWRESET. Here is the call graph for this function:



ILI9341_setPartialArea()

Set the partial display area for partial mode. Call before activating partial mode via ILI9341_setDisplayMode().

Parameters

rowStart	
rowEnd	

TODO: Implement

ILI9341_setRowAddress()

not using backlight, so these aren't necessary

Sets the start/end rows to be written to.

```
Should be called along with 'ILI9341_setColAddress()' and before 'ILI9341_writeMemCmd()'.
```

Parameters

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

This function is simply an interface to ILI9341_setAddress(). To work correctly, start_row must be no greater

than end_row, and end_row cannot be greater than the max row number (default 320).

ILI9341_setSleepMode()

```
void ILI9341_setSleepMode ( bool \ is\_sleeping \ )
```

Enter or exit sleep mode. The LCD driver is in sleep mode by default upon powering on or either kind of reset.

Parameters

is_sleeping	true to enter sleep mode, false to exit
-------------	---

This function turns sleep mode ON or OFF depending on the value of $is_sleeping$. Either way, the MCU must wait >= 5 [ms] before sending further commands.

It's also necessary to wait 120 [ms] before sending SPLOUT after sending SPLIN or a reset, so this function waits 120 [ms] regardless of the preceding event. Here is the call graph for this function:



ILI9341_write1px()

Write a single pixel to frame memory.

```
Call 'ILI9341_writeMemCmd()' before this one.
```

Parameters

red	5 or 6-bit R value
green	5 or 6-bit G value
blue	5 or 6-bit B value
is_16bit	true for 16-bit (65K colors, 2 transfers) color depth, false for 18-bit (262K colors, 3 transfer) color
	depth NOTE: set color depth via ILI9341_setColorDepth()

This function sends one pixel to the display. Because the serial interface (SPI) is used, each pixel requires 2 transfers in 16-bit mode and 3 transfers in 18-bit mode.

The following table (adapted from p. 63 of the datasheet) visualizes how the RGB data is sent to the display when using 16-bit color depth.

Transfer		1										2	2			
Bit #	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Value	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	В3	B2	B1	B0

The following table (adapted from p. 64 of the datasheet) visualizes how the RGB data is sent to the display when using 18-bit color depth.

Transfer		1								2	
Bit #	7	6	5	4	3	2	1	0	7	6	
Value	R5	R4	R3	R2	R1	R0	0/1	0/1	G5	G4	

Here is the call graph for this function:



ILI9341_writeMemCmd()

```
void ILI9341_writeMemCmd ( void \ \ )
```

Sends the "Write Memory" (RAMWR) command to the LCD driver, signalling that incoming data should be written to memory.

Should be called after setting the row (ILI9341_setRowAddress()) and/or and/or column (ILI9341_setRowAddress()) addresses, but before writing image data (ILI9341_write1px()). 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.

TM4C Pin	Function	ILI9341 Pin	Description
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 min. write cycle of 100 [ns], and a min. write cycle of 150 [ns]. Thus, this function sets the bit rate BR to be the bus frequency ($f_{bus} = 80 \, [MHz] \, dvided by 12$, allowing a bit rate of 6.67 [MHz], or a period of 150 [ns].

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
------	---------------------------

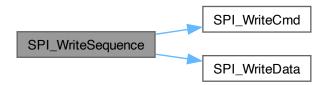
SPI_WriteSequence()

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()

```
void Timer0A_Wait1ms (
```

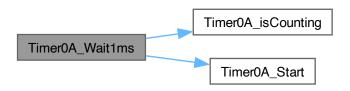
```
uint32_t time_ms )
```

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:



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

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

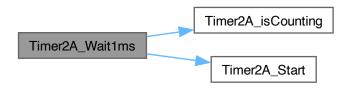
Timer2A_Wait1ms()

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

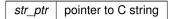
input_char

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

5.2 LCD_t Struct Reference

Data Fields

- uint16_t x1
- uint16_t **x2**
- uint16_t **y1**
- uint16_t **y2**
- uint32_t numPixels
- uint8_t R_val
- uint8_t G_val
- uint8_t **B_val**
- bool is_ON
- bool is_inverted
- · bool is 16bit
- bool is_init

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

• LCD.c

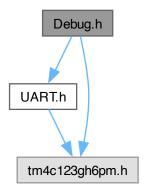
6 File Documentation 29

6 File Documentation

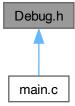
6.1 Debug.h File Reference

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

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

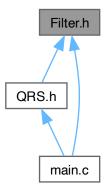
Author

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

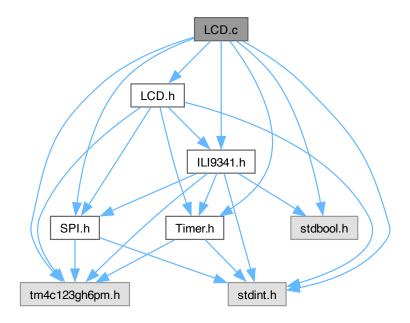
6.3 LCD.c File Reference

Source code for LCD module.

```
#include "LCD.h"
#include "ILI9341.h"
#include "SPI.h"
#include "Timer.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
```

6.3 LCD.c File Reference 31

#include <stdbool.h>
Include dependency graph for LCD.c:



Data Structures

• struct LCD_t

Functions

- void LCD_Init (void)
- void LCD_toggleStatus (void)

Toggle the display ON or OFF (ON by default)

void LCD_toggleInversion (void)

Toggle display inversion ON or OFF (OFF by default)

void LCD_toggleColorDepth (void)

Toggle 16-bit or 18-bit color depth (16-bit by default)

void LCD_setArea (uint16_t x1New, uint16_t x2New, uint16_t y1New, uint16_t y2New)

Set the area of the display to be written to.

void LCD_setRow (uint16_t x1New, uint16_t x2New)

Set only the rows to be written to.

void LCD_setCol (uint16_t y1New, uint16_t y2New)

Set only the columns to be written to.

• void LCD_setColor (uint8_t R_val, uint8_t G_val, uint8_t B_val)

Set the current color value for the display. Only the first 5-6 bits are used.

void LCD setColor 3bit (uint8 t color code)

Set the color value via a 3-bit code.

void LCD_draw (void)

Draw on the LCD display. This function should be called after setting the drawable area via $LCD_setArea()$, or after individually setting the row and column areas using LCD_setRow and LCD_setCol , respectively.

void LCD_drawHLine (uint16_t yCenter, uint16_t lineWidth)

Draw a horizontal line onto the display.

• void LCD_drawVLine (uint16_t xCenter, uint16_t lineWidth)

Draw a vertical line onto the display.

void LCD_drawRectangle (uint16_t x1, uint16_t y1, uint16_t dx, uint16_t dy, bool is_filled)

Draw an $1 \times h$ rectangle on the display. The bottom-left corner will be located at (x1, y1).

6.3.1 Detailed Description

Source code for LCD module.

Author

Bryan McElvy

6.3.2 Function Documentation

LCD_drawHLine()

Draw a horizontal line onto the display.

Parameters

yCenter	y-coordinate to center the line on
lineWidth	width of the line; should be a positive, odd number

LCD_drawRectangle()

Draw an 1 x h rectangle on the display. The bottom-left corner will be located at (x1, y1).

Parameters

x1	x-coordinate of bottom-left corner
y1	y-coordinate of bottom-left corner
dx	AKA 1; length (horizontal distance) of the rectangle
dy	AKA h; height (vertical distance) of the rectangle
_is_filled	true to fill the rectangle, false to leave it unfilled

6.3 LCD.c File Reference 33

Here is the call graph for this function:



LCD_drawVLine()

Draw a vertical line onto the display.

Parameters

xCenter	x-coordinate to center the line on
lineWidt	width of the line; should be a positive, odd number

LCD_setArea()

Set the area of the display to be written to.

Parameters

rowStart	<pre>index of top-most row; 0 <= rowStart <= rowEnd</pre>	
rowEnd	<pre>index of bottom-most row; rowStart <= rowEnd < NUM_ROWS</pre>	
colStart	index of left-most column; 0 <= colStart <= colEnd	
colEnd	<pre>index of right-most column; colStart <= colEnd < NUM_COLS</pre>	

LCD_setCol()

Set only the columns to be written to.

6.3 LCD.c File Reference 35

Parameters

colStart	<pre>index of left-most column; 0 <= colStart <= colEnd</pre>
colEnd	<pre>index of right-most column; colStart <= colEnd < NUM_COLS</pre>

LCD_setColor()

Set the current color value for the display. Only the first 5-6 bits are used.

Parameters

R_val 5-bit (0-31) R		5-bit (0-31) R value; 6-bit (0-63) if color depth is 18-bit
G_val 6-bit (0-63) G value		6-bit (0-63) G value
	B_val	5-bit (0-31) B value; 6-bit (0-63) if color depth is 18-bit

LCD_setColor_3bit()

Set the color value via a 3-bit code.

Parameters

color_code 3-bit color value to use. Bits 2, 1, 0 corresp	oond to R, G, and B values, respectively.
---	---

This is simply a convenience function for ${\tt LCD_setColor}$ (). The following table shows what the output color will be:

hex	binary	pixel color
0x04	100	red
0x06	110	yellow
0x02	010	green
0x03	011	cyan
0x01	001	blue
0x05	101	purple
0x07	111	white

Here is the call graph for this function:



LCD_setRow()

Set only the rows to be written to.

Parameters

rowStart	<pre>index of top-most row; 0 <= rowStart <= rowEnd</pre>
rowEnd	index of bottom-most row; rowStart <= rowEnd < NUM_ROWS

LCD_toggleColorDepth()

Toggle 16-bit or 18-bit color depth (16-bit by default)

Here is the call graph for this function:



LCD_toggleInversion()

6.4 LCD.h File Reference 37

Toggle display inversion ON or OFF (OFF by default)

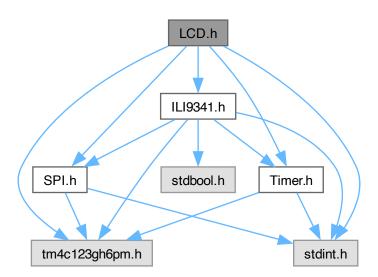
Here is the call graph for this function:



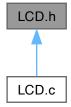
6.4 LCD.h File Reference

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

```
#include "ILI9341.h"
#include "SPI.h"
#include "Timer.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
Include dependency graph for LCD.h:
```



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



Macros

- #define LCD_RED (uint8_t) 0x04
- #define LCD_GREEN (uint8_t) 0x02
- #define LCD_BLUE (uint8_t) 0x01
- #define LCD_YELLOW (LCD_RED + LCD_GREEN)
- #define LCD_CYAN (LCD BLUE + LCD GREEN)
- #define **LCD_PURPLE** (LCD_RED + LCD_BLUE)
- #define LCD_WHITE (LCD_RED + LCD_BLUE + LCD_GREEN)

Functions

- void LCD_Init (void)
- void LCD_toggleStatus (void)

Toggle the display ON or OFF (ON by default)

void LCD_toggleInversion (void)

Toggle display inversion ON or OFF (OFF by default)

void LCD_toggleColorDepth (void)

Toggle 16-bit or 18-bit color depth (16-bit by default)

void LCD_setArea (uint16_t rowStart, uint16_t rowEnd, uint16_t colStart, uint16_t colEnd)

Set the area of the display to be written to.

void LCD_setRow (uint16_t rowStart, uint16_t rowEnd)

Set only the rows to be written to.

void LCD_setCol (uint16_t colStart, uint16_t colEnd)

Set only the columns to be written to.

• void LCD_setColor (uint8_t R_val, uint8_t G_val, uint8_t B_val)

Set the current color value for the display. Only the first 5-6 bits are used.

• void LCD_setColor_3bit (uint8_t color_code)

Set the color value via a 3-bit code.

• void LCD draw (void)

Draw on the LCD display. This function should be called after setting the drawable area via LCD_setArea(), or after individually setting the row and column areas using LCD_setRow and LCD_setCol, respectively.

void LCD_drawHLine (uint16_t yCenter, uint16_t lineWidth)

Draw a horizontal line onto the display.

void LCD_drawVLine (uint16_t xCenter, uint16_t lineWidth)

Draw a vertical line onto the display.

void LCD_drawRectangle (uint16_t x1, uint16_t y1, uint16_t dx, uint16_t dy, bool is_filled)

Draw an $1 \times h$ rectangle on the display. The bottom-left corner will be located at (x1, y1).

6.4 LCD.h File Reference 39

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_drawHLine()

Draw a horizontal line onto the display.

Parameters

yCe	nter	y-coordinate to center the line on
line	Nidth	width of the line; should be a positive, odd number

LCD_drawRectangle()

Draw an $1 \times h$ rectangle on the display. The bottom-left corner will be located at (x1, y1).

Parameters

x1	x-coordinate of bottom-left corner
y1 y-coordinate of bottom-left corner	
dx	AKA 1; length (horizontal distance) of the rectangle
dy	AKA h; height (vertical distance) of the rectangle
is_filled	true to fill the rectangle, false to leave it unfilled

Here is the call graph for this function:



LCD_drawVLine()

Draw a vertical line onto the display.

Parameters

xCenter	x-coordinate to center the line on
lineWidth	width of the line; should be a positive, odd number

LCD_setArea()

Set the area of the display to be written to.

Parameters

rowStart	<pre>index of top-most row; 0 <= rowStart <= rowEnd</pre>		
rowEnd	<pre>index of bottom-most row; rowStart <= rowEnd < NUM_ROWS</pre>		
colStart	index of left-most column; 0 <= colStart <= colEnd		
colEnd	index of right-most column; colStart <= colEnd < NUM_COLS		

LCD_setCol()

6.4 LCD.h File Reference 41

Set only the columns to be written to.

Parameters

colStart	<pre>index of left-most column; 0 <= colStart <= colEnd</pre>		
colEnd	<pre>index of right-most column; colStart <= colEnd < NUM_COLS</pre>		

LCD_setColor()

Set the current color value for the display. Only the first 5-6 bits are used.

Parameters

R_val	5-bit (0-31) R value; 6-bit (0-63) if color depth is 18-bit
G_val 6-bit (0-63) G value	
B_val	5-bit (0-31) B value; 6-bit (0-63) if color depth is 18-bit

LCD_setColor_3bit()

Set the color value via a 3-bit code.

Parameters

color_code 3-bit color value to use. Bits 2, 1, 0 corre	espond to R, G, and B values, respectively.
---	---

This is simply a convenience function for ${\tt LCD_setColor}$ (). The following table shows what the output color will be:

hex	binary	pixel color
0x04	100	red
0x06	110	yellow
0x02	010	green
0x03	011	cyan
0x01	001	blue
0x05	101	purple
0x07	111	white

6.4 LCD.h File Reference 43

Here is the call graph for this function:



LCD_setRow()

Set only the rows to be written to.

Parameters

rowStart	<pre>index of top-most row; 0 <= rowStart <= rowEnd</pre>
rowEnd	index of bottom-most row; rowStart <= rowEnd < NUM_ROWS

LCD_toggleColorDepth()

Toggle 16-bit or 18-bit color depth (16-bit by default)

Here is the call graph for this function:



LCD_toggleInversion()

Toggle display inversion ON or OFF (OFF by default)

Here is the call graph for this function:



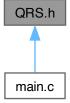
6.5 QRS.h File Reference

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

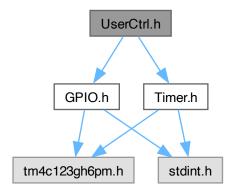
This module contains functions for detecting heart rate (HR) using a simplified version of the Pan-Tompkins algorithm.

6.6 UserCtrl.h File Reference

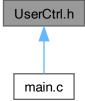
Interface for user control module.

```
#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()

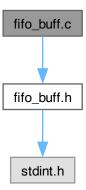
```
void UserCtrl_Init ( )
```

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.

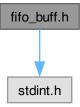
Parameters

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

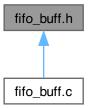
6.8 fifo_buff.h File Reference

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.

6.9 ADC.c File Reference 51

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.

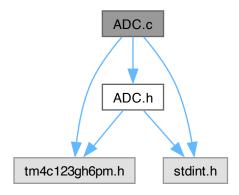
Parameters

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

6.9 ADC.c File Reference

```
#include "ADC.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
```

Include dependency graph for ADC.c:



6.9.1 Detailed Description

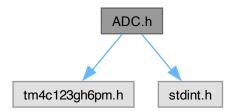
Author

Bryan McElvy

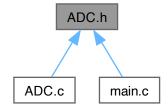
6.10 ADC.h File Reference

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

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



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



6.10.1 Detailed Description

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

Author

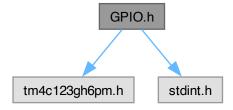
Bryan McElvy

6.11 GPIO.h File Reference

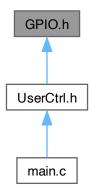
Driver module for using the LaunchPad's onboard switches and RGB LEDs for GPIO and interrupts.

```
#include "tm4c123gh6pm.h"
#include <stdint.h>
Include dependency graph for GPIO h
```

Include dependency graph for GPIO.h:



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



Functions

```
• void GPIO_PF_Init (void)
```

Initialize GPIO Port F.

• void GPIO_PF_LED_Init (void)

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

• void GPIO_PF_LED_Write (uint8_t color_mask, uint8_t on_or_off)

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

void GPIO_PF_LED_Toggle (uint8_t color_mask)

Toggle the selected LED(s).

• void GPIO_PF_Sw_Init (void)

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

void GPIO_PF_Interrupt_Init (void)

Initialize GPIO Port F interrupts via Sw1 and Sw2.

6.11.1 Detailed Description

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

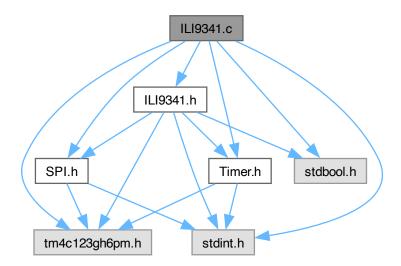
Author

Bryan McElvy

6.12 ILI9341.c File Reference

Source code for ILI9341 module.

```
#include "ILI9341.h"
#include "SPI.h"
#include "Timer.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
#include <stdbool.h>
Include dependency graph for ILI9341.c:
```



Macros

- #define NOP (uint8_t) 0x00
- #define **SWRESET** (uint8_t) 0x01
- #define SPLIN (uint8_t) 0x10
- #define SPLOUT (uint8 t) 0x11
- #define **PTLON** (uint8_t) 0x12
- #define NORON (uint8_t) 0x13
- #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 **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 FRMCTR1 (uint8 t) 0xB1
- #define PRCTR (uint8_t) 0xB5
- #define IFCTL (uint8_t) 0xF6

Functions

void ILI9341_Init (void)

Initialize the LCD driver, the SPI module, and Timer2A.

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 setSleepMode (bool is sleeping)

Enter or exit sleep mode. The LCD driver is in sleep mode by default upon powering on or either kind of reset.

void ILI9341 setDispMode (bool is normal)

Set the display to normal mode or partial mode. The LCD driver starts out in normal mode. Calling with either possible value exits scrolling mode.

void ILI9341 setPartialArea (uint16 t rowStart, uint16 t rowEnd)

Set the partial display area for partial mode. Call before activating partial mode via ILI9341_setDisplayMode().

void ILI9341 setDispInversion (bool is ON)

Toggle display inversion. Turning ON causes colors to be inverted on the display.

void ILI9341_setDispOutput (bool is_ON)

Turn display output ON or OFF. This function clears the display and stops outputting to the display area, but does not affect frame memory or power.

void ILI9341 setVertScrollArea (uint16 t top fixed, uint16 t vert scroll, uint16 t bottom fixed)

TODO: Write.

void ILI9341_setVertScrollStart (uint16_t start_address)

TODO: Write.

 void ILI9341_setMemAccessCtrl (bool areRowsFlipped, bool areColsFlipped, bool areRowsColsSwitched, bool isVertRefreshFlipped, bool isColorOrderFlipped, bool isHorRefreshFlipped)

Set how data is converted from memory to display.

• void ILI9341_setColorDepth (bool is_16bit)

Set the pixel format to be 16-bit (65K colors) or 18-bit (262K colors).

void ILI9341_NoOpCmd (void)

Send the "No Operation" command ($NOP = 0 \times 00$) to the LCD driver. Can be used to terminate the "Memory Write" (RAMWR) and "Memory Read" (RAMRD) commands, but does nothing otherwise.

• void ILI9341 setFrameRate (uint8 t div ratio, uint8 t clocks per line)

TODO: Write.

• void ILI9341 setBlankingPorch (uint8 t vpf, uint8 t vbp, uint8 t hfp, uint8 t hbp)

TODO: Write.

void ILI9341 setInterface (void)

Sets the interface for the ILI9341. The parameters for this command are hard-coded, so it only needs to be called once upon initialization.

void ILI9341_setRowAddress (uint16_t start_row, uint16_t end_row)

not using backlight, so these aren't necessary

• 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" (RAMWR) command to the LCD driver, signalling that incoming data should be written to memory.

void ILI9341_write1px (uint8_t red, uint8_t green, uint8_t blue, bool is_16bit)

Write a single pixel to frame memory.

6.12.1 Detailed Description

Source code for ILI9341 module.

Author

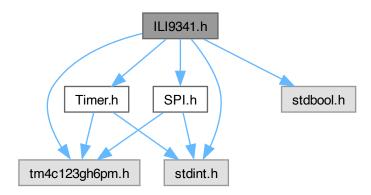
Bryan McElvy

6.13 ILI9341.h File Reference

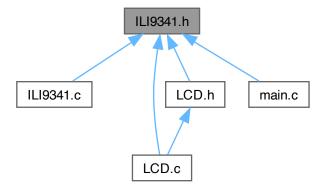
Driver module for interfacing with an ILI9341 LCD driver.

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

Include dependency graph for ILI9341.h:



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



Macros

- #define NUM_COLS (uint16_t) 240
- #define NUM_ROWS (uint16 t) 320

Functions

void ILI9341_Init (void)

Initialize the LCD driver, the SPI module, and Timer2A.

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_setSleepMode (bool is_sleeping)

Enter or exit sleep mode. The LCD driver is in sleep mode by default upon powering on or either kind of reset.

void ILI9341_setDispMode (bool is_normal)

Set the display to normal mode or partial mode. The LCD driver starts out in normal mode. Calling with either possible value exits scrolling mode.

void ILI9341 setPartialArea (uint16 t rowStart, uint16 t rowEnd)

Set the partial display area for partial mode. Call before activating partial mode via ILI9341_setDisplayMode().

void ILI9341 setDispInversion (bool is ON)

Toggle display inversion. Turning ON causes colors to be inverted on the display.

void ILI9341_setDispOutput (bool is_ON)

Turn display output ON or OFF. This function clears the display and stops outputting to the display area, but does not affect frame memory or power.

· void ILI9341 setVertScrollArea (uint16 t top fixed, uint16 t vert scroll, uint16 t bottom fixed)

TODO: Write.

• void ILI9341_setVertScrollStart (uint16_t start_address)

TODO: Write.

 void ILI9341_setMemAccessCtrl (bool areRowsFlipped, bool areColsFlipped, bool areRowsColsSwitched, bool isVertRefreshFlipped, bool isColorOrderFlipped, bool isHorRefreshFlipped)

Set how data is converted from memory to display.

void ILI9341_setColorDepth (bool is_16bit)

Set the pixel format to be 16-bit (65K colors) or 18-bit (262K colors).

void ILI9341_NoOpCmd (void)

Send the "No Operation" command ($NOP = 0 \times 00$) to the LCD driver. Can be used to terminate the "Memory Write" (RAMWR) and "Memory Read" (RAMRD) commands, but does nothing otherwise.

void ILI9341_setFrameRate (uint8_t div_ratio, uint8_t clocks_per_line)

TODO: Write.

• void ILI9341_setBlankingPorch (uint8_t vpf, uint8_t vbp, uint8_t hfp, uint8_t hbp)

TODO: Write.

· void ILI9341 setInterface (void)

Sets the interface for the ILI9341. The parameters for this command are hard-coded, so it only needs to be called once upon initialization.

void ILI9341_setRowAddress (uint16_t start_row, uint16_t end_row)

not using backlight, so these aren't necessary

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" (RAMWR) command to the LCD driver, signalling that incoming data should be written to memory.

void ILI9341_write1px (uint8_t red, uint8_t green, uint8_t blue, bool is_16bit)

Write a single pixel to frame memory.

6.14 isr.c File Reference 59

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)

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

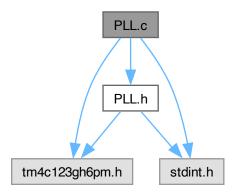
Bryan McElvy

6.15 PLL.c File Reference

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

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

#include <stdint.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.

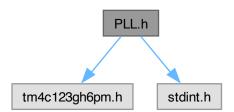
Author

Bryan McElvy

6.16 PLL.h File Reference

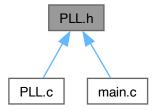
Driver module for activating the phase-locked-loop (PLL).

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



6.17 SPI.c File Reference 61

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



Functions

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

6.16.1 Detailed Description

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

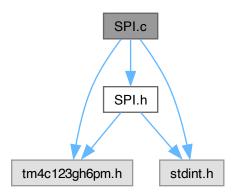
Author

Bryan McElvy

6.17 SPI.c File Reference

Source code for SPI module.

```
#include "SPI.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
Include dependency graph for SPI.c:
```



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.

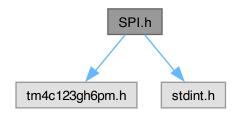
Author

Bryan McElvy

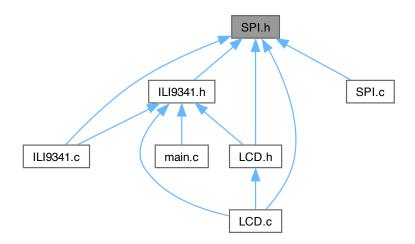
6.18 SPI.h File Reference

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

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



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



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

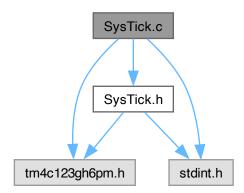
6.19 SysTick.c File Reference

Implementation details for SysTick functions.

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

#include <stdint.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.

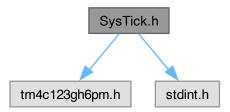
Author

Bryan McElvy

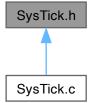
6.20 SysTick.h File Reference

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

```
#include "tm4c123gh6pm.h"
#include <stdint.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.

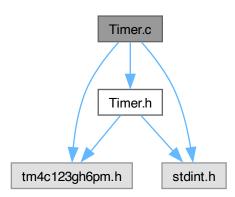
Author

Bryan McElvy

6.21 Timer.c File Reference

Implementation for timer module.

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

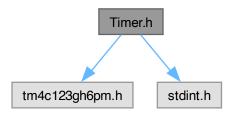
Author

Bryan McElvy

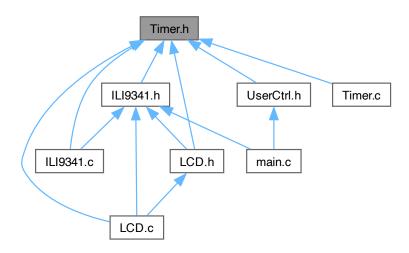
6.22 Timer.h File Reference

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

```
#include "tm4c123gh6pm.h"
#include <stdint.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).

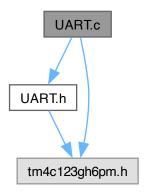
Author

Bryan McElvy

6.23 UART.c File Reference

Source code for UART module.

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

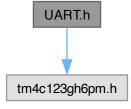
Author

Bryan McElvy

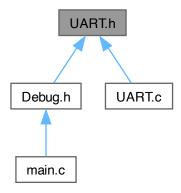
6.24 UART.h File Reference

Driver module for serial communication via UART0 and UART 1.

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



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



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.

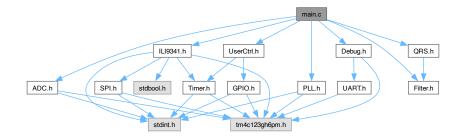
UART1 uses PB0 (Rx) and PB1 (Tx), which are not broken out but do not connect to a serial port.

6.25 main.c File Reference

Main program file for ECG-HRM.

```
#include "ADC.h"
#include "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, 51	GPIO br>, 7
ADC.h, 52	GPIO_PortF_Handler
Application Software, 26	Program Threads, 27
Debug.h, 29	ILI9341 >, 7
Device Drivers, 4	ILI9341_resetHard, 9
	ILI9341_resetSoft, 9
FIFO_add_sample	ILI9341_setBlankingPorch, 9
fifo_buff.c, 47	ILI9341_setColAddress, 9
fifo_buff.h, 50	ILI9341_setColorDepth, 10
fifo_buff.c, 46	ILI9341_setDispInversion, 10
FIFO_add_sample, 47	ILI9341_setDispMode, 11
FIFO_get_size, 47	ILI9341_setDispOutput, 11
FIFO_init, 48	ILI9341_setFrameRate, 12
FIFO_rem_sample, 48	ILI9341_setInterface, 12
FIFO_show_data, 48	ILI9341_setMemAccessCtrl, 13
fifo_buff.h, 49 FIFO_add_sample, 50	ILI9341_setPartialArea, 14
FIFO_get_size, 50	ILI9341_setRowAddress, 14
FIFO_init, 50	ILI9341_setSleepMode, 15
FIFO_rem_sample, 51	ILI9341_write1px, 15
FIFO_show_data, 51	ILI9341_writeMemCmd, 16
FIFO_buffer_t, 28	ILI9341.c, 55 ILI9341.h, 57
FIFO_get_size	ILI9341.resetHard
fifo_buff.c, 47	ILI9341 br>, 9
fifo_buff.h, 50	ILI9341 resetSoft
FIFO init	ILI9341 br>, 9
fifo_buff.c, 48	ILI9341_setBlankingPorch
fifo_buff.h, 50	ILI9341 br>, 9
FIFO_rem_sample	ILI9341_setColAddress
fifo_buff.c, 48	ILI9341 , 9
fifo_buff.h, 51	ILI9341_setColorDepth
FIFO_show_data	ILI9341 , 10
fifo_buff.c, 48	ILI9341_setDispInversion
fifo_buff.h, 51	ILI9341 , 10
Filter.h, 30	ILI9341_setDispMode
GPIO 5	ILI9341 , 11
GPIO_PF_Init, 5	ILI9341_setDispOutput
GPIO_PF_Interrupt_Init, 5	ILI9341 br>, 11
GPIO_PF_LED_Init, 6	ILI9341_setFrameRate
GPIO_PF_LED_Toggle, 6	ILI9341 br>, 12
GPIO_PF_LED_Write, 6	ILI9341_setInterface
GPIO_PF_Sw_Init, 7	ILI9341 ILI9341
GPIO.h, 53	ILI9341_setMemAccessCtrl
GPIO_PF_Init	ILI9341 br>, 13 ILI9341 setPartialArea
GPIO br>, 5	ILI9341 br>, 14
GPIO_PF_Interrupt_Init	ILI9341 setRowAddress
GPIO < br >, 5	ILI9341 br>, 14
GPIO_PF_LED_Init	ILI9341_setSleepMode
GPIO br>, 6	ILI9341 br>, 15
GPIO_PF_LED_Toggle	ILI9341_write1px
GPIO or>, 6	ILI9341 , 15
GPIO_PF_LED_Write	ILI9341_writeMemCmd
GPIO br>, 6	

74 INDEX

11.10044 - 4.10	DI 1 1 1 1 7
ILI9341 br>, 16	PLL PLL 17
isr.c, 59	PLL_Init, 17
LCD.c, 30	PLL.c, 59
	PLL.h, 60
LCD_drawHLine, 32	PLL_Init
LCD_drawRectangle, 32	PLL , 17
LCD_drawVLine, 33	Program Threads, 26
LCD_setArea, 33	GPIO_PortF_Handler, 27
LCD_setCol, 33	SysTick_Handler, 27
LCD_setColor, 35	Timer1A_Handler, 27
LCD_setColor_3bit, 35	Timer1A_Init, 27
LCD_setRow, 36	
LCD_toggleColorDepth, 36	QRS.h, 44
LCD_toggleInversion, 36	CDI chas 47
LCD.h, 37	SPI SPI 17
LCD_drawHLine, 39	SPI_Init, 17
LCD_drawRectangle, 39	SPI_Read, 18
LCD_drawVLine, 40	SPI_WriteCmd, 18
LCD_setArea, 40	SPI_WriteData, 18
LCD_setCol, 40	SPI_WriteSequence, 19
LCD_setColor, 42	SPI.c, 61
LCD_setColor_3bit, 42	SPI.h, 62
LCD_setRow, 43	SPI_Init
LCD_toggleColorDepth, 43	SPI br>, 17
LCD_toggleInversion, 43	SPI_Read
LCD_drawHLine	SPI br>, 18
LCD.c, 32	SPI_WriteCmd
LCD.h, 39	SPI br>, 18
LCD_drawRectangle	SPI_WriteData
LCD.c, 32	SPI br>, 18
LCD.h, 39	SPI_WriteSequence
LCD_drawVLine	SPI br>, 19
LCD.c, 33	SysTick , 19
LCD.h, 40	SysTick_Interrupt_Init, 20
LCD_setArea	SysTick_Timer_Init, 20
LCD.c, 33	SysTick.c, 63
LCD.h, 40	SysTick.h, 64
LCD_setCol	SysTick_Handler
LCD.c, 33	Program Threads, 27
LCD.h, 40	SysTick_Interrupt_Init
LCD_setColor	SysTick , 20
LCD.c, 35	SysTick_Timer_Init
LCD.h, 42	SysTick , 20
LCD_setColor_3bit	
LCD.c, 35	Timer br>, 20
LCD.h, 42	Timer0A_Init, 21
LCD setRow	Timer0A_isCounting, 21
 LCD.c, 36	Timer0A_Start, 21
LCD.h, 43	Timer0A_Wait1ms, 21
LCD_t, 28	Timer2A_Init, 22
LCD_toggleColorDepth	Timer2A_isCounting, 22
LCD.c, 36	Timer2A_Start, 22
LCD.h, 43	Timer2A_Wait1ms, 23
LCD_toggleInversion	Timer.c, 66
LCD.c, 36	Timer.h, 67
LCD.h, 43	Timer0A_Init
,	Timer br>, 21
main.c, 71	Timer0A_isCounting
	Timer br>, 21

INDEX 75

```
Timer0A_Start
     Timer <br/>br>, 21
Timer0A_Wait1ms
    Timer <br/>br>, 21
Timer1A_Handler
     Program Threads, 27
Timer1A_Init
     Program Threads, 27
Timer2A Init
     Timer <br/>br>, 22
Timer2A_isCounting
     Timer \langle br \rangle, 22
Timer2A_Start
     Timer <br/>br>, 22
Timer2A_Wait1ms
    Timer \langle br \rangle, 23
UART <br/>br>, 23
     UART0_Init, 24
     UARTO ReadChar, 24
     UART0_WriteChar, 24
     UART0_WriteStr, 24
     UART1_Init, 25
     UART1_ReadChar, 25
     UART1_WriteChar, 25
     UART1_WriteStr, 26
UART.c, 68
UART.h, 69
UART0_Init
     UART <br/>br>, 24
UART0_ReadChar
     UART <br/>br>, 24
UART0_WriteChar
     UART < br >, 24
UART0 WriteStr
     UART <br/>br>, 24
UART1_Init
     UART <br>, 25
UART1_ReadChar
     UART <br/>br>, 25
UART1_WriteChar
     UART <br/>br>, 25
UART1 WriteStr
     UART <br/>br>, 26
UserCtrl.h, 45
     UserCtrl_Init, 46
UserCtrl Init
     UserCtrl.h, 46
```