ECG-HRM

Generated by Doxygen 1.9.8

1 Topic Index	1
1.1 Topics	1
2 Data Structure Index	2
2.1 Data Structures	2
3 File Index	2
3.1 File List	
6	
4 Topic Documentation	4
4.1 Device Drivers	
4.1.1 Detailed Description	5
4.1.2 Analog-to-Digital Conversion (ADC)	6
4.1.3 GPIO	7
4.1.4 Phase-Locked Loop (PLL)	7
4.1.5 Serial Peripheral Interface (SPI)	8
4.1.6 System Tick (SysTick)	10
4.1.7 Timer	11
4.1.8 Universal Asynchronous Receiver/Transmitter (UART)	14
4.2 Middleware	17
4.2.1 Detailed Description	18
4.2.2 ILI9341	18
4.2.3 LED	27
4.3 Application Software	
4.3.1 Detailed Description	
4.3.2 Data Acquisition (DAQ)	
4.3.3 Debug	
4.3.4 LCD	
4.3.5 OBS	40
4.4 Common	
4.4 Common	
4.4.1 Detailed Description	
4.4.3 FIFO	
4.4.4 NewAssert	47
5 Data Structure Documentation	47
5.1 FIFO_t Struct Reference	47
5.2 GPIO_Port_t Struct Reference	47
5.3 Led_t Struct Reference	48
5.4 UART_t Struct Reference	
6 File Documentation	48
6.1 DAQ.c File Reference	_
6.1.1 Detailed Description	

6.2 DAQ.h File Reference	49
6.2.1 Detailed Description	50
6.3 Debug.h File Reference	50
6.3.1 Detailed Description	50
6.3.2 Function Documentation	50
6.4 LCD.c File Reference	51
6.4.1 Detailed Description	53
6.5 LCD.h File Reference	53
6.5.1 Detailed Description	54
6.6 QRS.h File Reference	54
6.6.1 Detailed Description	54
6.7 FIFO.c File Reference	55
6.7.1 Detailed Description	56
6.8 FIFO.h File Reference	56
6.8.1 Detailed Description	57
6.9 lookup.c File Reference	57
6.9.1 Detailed Description	57
6.10 lookup.h File Reference	57
6.10.1 Detailed Description	58
6.11 NewAssert.c File Reference	58
6.11.1 Detailed Description	58
6.12 NewAssert.h File Reference	58
6.12.1 Detailed Description	58
6.13 ADC.c File Reference	59
6.13.1 Detailed Description	59
6.14 ADC.h File Reference	59
6.14.1 Detailed Description	60
6.15 GPIO.c File Reference	60
6.15.1 Detailed Description	61
6.15.2 Function Documentation	62
6.16 GPIO.h File Reference	67
6.16.1 Detailed Description	69
6.16.2 Function Documentation	69
6.17 PLL.c File Reference	75
6.17.1 Detailed Description	75
6.18 PLL.h File Reference	75
6.18.1 Detailed Description	76
6.19 SPI.c File Reference	76
6.19.1 Detailed Description	77
6.20 SPI.h File Reference	77
6.20.1 Detailed Description	77
6.21 SysTick.c File Reference	77

6.21.1 Detailed Description	78
6.22 SysTick.h File Reference	78
6.22.1 Detailed Description	78
6.23 Timer.c File Reference	78
6.23.1 Detailed Description	79
6.24 Timer.h File Reference	79
6.24.1 Detailed Description	80
6.25 UART.c File Reference	80
6.25.1 Detailed Description	81
6.26 UART.h File Reference	82
6.26.1 Detailed Description	82
6.27 main.c File Reference	82
6.27.1 Detailed Description	83
6.28 ILI9341.c File Reference	83
6.28.1 Detailed Description	84
6.29 ILI9341.h File Reference	85
6.29.1 Detailed Description	86
6.30 Led.c File Reference	86
6.30.1 Detailed Description	87
6.31 Led.h File Reference	87
6.31.1 Detailed Description	88
6.32 test_adc.c File Reference	88
6.32.1 Detailed Description	88
6.33 test_daq.c File Reference	89
6.33.1 Detailed Description	89
6.34 test_debug.c File Reference	89
6.34.1 Detailed Description	90
6.35 test_fifo.c File Reference	90
6.35.1 Detailed Description	90
6.36 test_lcd_image.c File Reference	91
6.36.1 Detailed Description	91
6.37 test_lcd_scroll.c File Reference	91
6.37.1 Detailed Description	92
6.38 test_pll.c File Reference	92
6.38.1 Detailed Description	92
6.39 test_spi.c File Reference	93
6.39.1 Detailed Description	93
6.40 test_systick_int.c File Reference	93
6.40.1 Detailed Description	93
6.41 test_timer1_int.c File Reference	94
6.41.1 Detailed Description	94
6.42 test uart interrupt c File Reference	94

1 Topic Index

6.42.1 Detailed Description	95
6.42.2 Variable Documentation	95
6.43 test_uart_la.c File Reference	95
6.43.1 Detailed Description	
6.44 test_uart_write.c File Reference	
6.44.1 Detailed Description	
6.45 test_userctrl.c File Reference	
0.40.1 Detailed Description	
Index	97
1 Topic Index	
1.1 Topics	
Here is a list of all topics with brief descriptions:	
Device Drivers	4
Analog-to-Digital Conversion (ADC)	6
GPIO	7
Phase-Locked Loop (PLL)	7
Serial Peripheral Interface (SPI)	8
System Tick (SysTick)	10
Timer	11
Universal Asynchronous Receiver/Transmitter (UART)	14
Middleware	17
IL19341	18
LED	27
Application Software	30
Data Acquisition (DAQ)	31
Debug	32
LCD	33
QRS	40
Common	40
FIFO	41
NewAssert	47

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

FIFO_t	4
GPIO_Port_t	4
Led_t	4
UART_t	4

3 File Index

3.1 File List

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

Source code for DAQ module	48
DAQ.h Application software for handling data acquision (DAQ) functions	49
Debug.h Functions to output debugging information to a serial port via UART	50
LCD.c Source code for LCD module	51
LCD.h Module for outputting the ECG waveform and HR to a liquid crystal display (LCD)	53
QRS.h QRS detection algorithm functions	54
FIFO.c Source code for FIFO buffer module	55
FIFO.h FIFO buffer data structure	56
lookup.c Lookup table source code	57
lookup.h Lookup table API	57
NewAssert.c Source code for custom assert implementation	58
NewAssert.h Header file for custom assert implementation	58

3.1 File List 3

ADC.c Source code for ADC module	59
ADC.h Driver module for analog-to-digital conversion (ADC)	59
GPIO.c Source code for GPIO module	60
GPIO.h Header file for general-purpose input/output (GPIO) device driver	67
PLL.c Implementation details for phase-lock-loop (PLL) functions	75
PLL.h Driver module for activating the phase-locked-loop (PLL)	75
SPI.c Source code for SPI module	76
SPI.h Driver module for using the serial peripheral interface (SPI) protocol	77
SysTick.c Implementation details for SysTick functions	77
SysTick.h Driver module for using SysTick-based timing and/or interrupts	78
Timer.c Implementation for timer module	78
Timer.h Driver module for general-purpose timer modules	79
UART.c Source code for UART module	80
UART.h Driver module for serial communication via UART0 and UART 1	82
main.c Main program file for ECG-HRM	82
ILI9341.c Source code for ILI9341 module	83
ILI9341.h Driver module for interfacing with an ILI9341 LCD driver	85
Led.c Source code for LED module	86
Led.h Interface for LED module	87
test_adc.c Test script for analog-to-digital conversion (ADC) module	88
test_daq.c Test script for the data acquisition (DAQ) module	89

test_debug.c	
Test script for Debug module	89
test_fifo.c	
Test script for FIFO buffer	90
test_lcd_image.c	
Test script for writing images onto the display	91
test_lcd_scroll.c	
Test script for writing different colors on the LCD	91
test pll.c	
Test script for the PLL module	92
test spi.c	
Test script for initializing SSI0 and writing data/commands via SPI	93
test systick int.c	
Test script for SysTick interrupts	93
test timer1 int.c	
Test script for Timer1A interrupts	94
test uart interrupt.c	
(DISABLED) Test script for writing to serial port via UART0	94
test uart la.c	
Test script for using a USB logic analyzer to decode UART signals	95
test_uart_write.c	
Test script for writing to serial port via UART0	96
test userctrl.c	
Test file for GPIO/UserCtrl modules and GPIO interrupts	96

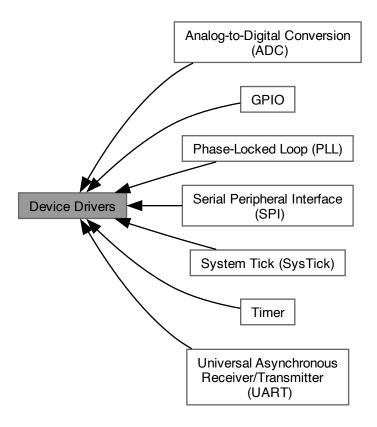
4 Topic Documentation

4.1 Device Drivers

Low level device driver modules.

4.1 Device Drivers 5

Collaboration diagram for Device Drivers:



Modules

- Analog-to-Digital Conversion (ADC)
- GPIO
- Phase-Locked Loop (PLL)
- Serial Peripheral Interface (SPI)
- System Tick (SysTick)
- Timer
- Universal Asynchronous Receiver/Transmitter (UART)

4.1.1 Detailed Description

Low level device driver modules.

These modules contain functions for interfacing with peripherals available on the TM4C123GH6PM microcontroller.

4.1.2 Analog-to-Digital Conversion (ADC)

Collaboration diagram for Analog-to-Digital Conversion (ADC):



Files

• file ADC.c

Source code for ADC module.

• file ADC.h

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

Functions

void ADC_Init (void)

Initialize ADC0 as a single-input analog-to-digital converter.

• void ADC_InterruptEnable (void)

Enable the ADC interrupt.

void ADC_InterruptDisable (void)

Disable the ADC interrupt.

• float32_t ADC_ConvertToVolts (uint16_t raw_sample)

Convert a raw ADC sample to voltage in [mV].

4.1.2.1 Detailed Description

Functions for differential-input analog-to-digital conversion.

4.1.2.2 Function Documentation

ADC_ConvertToVolts()

Convert a raw ADC sample to voltage in [mV].

Parameters

raw_sample	12-bit unsigned ADC value.	sample	=	[0,	0xFFF]
------------	----------------------------	--------	---	-----	--------

4.1 Device Drivers 7

Returns

double Voltage value in range [-5.5, 5.5) [mV].

4.1.3 GPIO

Collaboration diagram for GPIO:



Functions for using general-purpose input/output (GPIO) ports.

4.1.4 Phase-Locked Loop (PLL)

Collaboration diagram for Phase-Locked Loop (PLL):



Files

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

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

Functions

• void PLL_Init (void)

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

4.1.4.1 Detailed Description

Function for initializing the phase-locked loop.

4.1.5 Serial Peripheral Interface (SPI)

Collaboration diagram for Serial Peripheral Interface (SPI):



Files

• file SPI.c

Source code for SPI module.

• file SPI.h

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

Macros

- #define SPI SET DC() (GPIO PORTA DATA R |= 0x40)
- #define **SPI_CLEAR_DC**() (GPIO_PORTA_DATA_R &= \sim (0x40))
- #define SPI_IS_BUSY (SSI0_SR_R & 0x10)
- #define SPI_TX_ISNOTFULL (SSI0_SR_R & 0x02)

Enumerations

enum {

```
 \begin{split} & \textbf{SPI\_CLK\_PIN} = \texttt{GPIO\_PIN2} \;, \; \textbf{SPI\_CS\_PIN} = \texttt{GPIO\_PIN3} \;, \; \textbf{SPI\_RX\_PIN} = \texttt{GPIO\_PIN4} \;, \; \textbf{SPI\_TX\_PIN} = \texttt{GPIO\_PIN5} \;, \\ & \textbf{SPI\_DC\_PIN} = \texttt{GPIO\_PIN6} \;, \; \textbf{SPI\_RESET\_PIN} = \texttt{GPIO\_PIN7} \;, \; \textbf{SPI\_SSI0\_PINS} = (\texttt{SPI\_CLK\_PIN} \mid \texttt{SPI\_CLK\_PIN} \mid \texttt{SPI\_CLK\_PIN} \mid \texttt{SPI\_TX\_PIN} \;, \\ & \textbf{SPI\_ALL\_PINS} = (\texttt{SPI\_SSI0\_PINS} \mid \texttt{SPI\_GPIO\_PINS} \;) \; \end{split}
```

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 the 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.

4.1 Device Drivers 9

4.1.5.1 Detailed Description

Functions for SPI-based communication via SSI0 peripheral.

4.1.5.2 Macro Definition Documentation

SPI_SET_DC

#define SPI_SET_DC() (GPIO_PORTA_DATA_R \mid = 0x40)

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)
```

4.1.5.3 Function Documentation

SPI_Init()

```
void SPI_Init (
     void )
```

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

The bit rate BR is set using the (positive, even-numbered) clock prescale divisor CPSDVSR and the SCR field in the SSI Control 0 (CR0) register:

$$BR = f_{bus}/(CPSDVSR * (1 + SCR))$$

The ILI9341 driver has a min. read cycle of 150 [ns] and a min. write cycle of 100 [ns], so the bit rate BR is set to be equal to the bus frequency ($f_{bus}=80[MHz]$) divided by 8, allowing a bit rate of 10 [MHz], or a period of 100 [ns].

SPI_Read()

Read data from the peripheral.

Returns

uint8_t

SPI_WriteCmd()

Write an 8-bit command to the peripheral.

Parameters

```
cmd command for peripheral
```

SPI_WriteData()

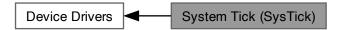
Write 8-bit data to the peripheral.

Parameters

data input data for peripheral

4.1.6 System Tick (SysTick)

Collaboration diagram for System Tick (SysTick):



Files

• file SysTick.c

Implementation details for SysTick functions.

file SysTick.h

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

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 Device Drivers 11

4.1.6.1 Detailed Description

Functions for timing and periodic interrupts via SysTick.

4.1.6.2 Function Documentation

SysTick_Interrupt_Init()

Initialize SysTick for interrupts.

Parameters

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

4.1.7 Timer

Collaboration diagram for Timer:



Files

• file Timer.c

Implementation for timer module.

• file Timer.h

Driver module for general-purpose timer modules.

Timer0A

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].

Timer1A

void Timer1A_Init (uint32_t time_ms)

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

Timer2A

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].

void Timer3A_Init (uint32_t time_ms)

Initialize Timer3A as a 32-bit, periodic, countdown timer that triggers ADC sample capture.

4.1.7.1 Detailed Description

Functions for timing and periodic interrupts via general-purpose timer modules (GPTM).

4.1.7.2 Function Documentation

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.

4.1 Device Drivers 13

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.
```

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 seconds.
```

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.
```

Timer3A_Init()

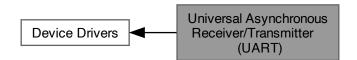
Initialize Timer3A as a 32-bit, periodic, countdown timer that triggers ADC sample capture.

Parameters

time_ms	Time in [ms] to load into Timer3A. Must be <= 53 seconds.
---------	---

4.1.8 Universal Asynchronous Receiver/Transmitter (UART)

Collaboration diagram for Universal Asynchronous Receiver/Transmitter (UART):



Files

• file UART.c

Source code for UART module.

file UART.h

Driver module for serial communication via UART0 and UART 1.

Data Structures

struct UART_t

4.1 Device Drivers 15

Macros

• #define ASCII CONVERSION 0x30

Typedefs

typedef volatile uint32_t * register_t

Enumerations

```
    enum GPIO BASE ADDRESSES {

 GPIO PORTA BASE = (uint32 t) 0x40004000 , GPIO PORTB BASE = (uint32 t) 0x40005000 , GPIO ←
 PORTC BASE = (uint32 t) 0x40006000 , GPIO PORTD BASE = (uint32 t) 0x40007000 ,
 GPIO_PORTE_BASE = (uint32_t) 0x40024000 , GPIO_PORTF_BASE = (uint32_t) 0x40025000 }
enum UART_BASE_ADDRESSES {
 UART0_BASE = (uint32_t) 0x4000C000 , UART1_BASE = (uint32_t) 0x4000D000 , UART2_BASE =
 (uint32 t) 0x4000E000 , UART3 BASE = (uint32 t) 0x4000F000 ,
 UART4_BASE = (uint32_t) 0x40010000 , UART5_BASE = (uint32_t) 0x40011000 , UART6_BASE =
 (uint32 t) 0x40012000, UART7_BASE = (uint32 t) 0x40013000 }

    enum UART_REG_OFFSETS {

 UART FR R OFFSET = (uint32 t) 0x18 , IBRD R OFFSET = (uint32 t) 0x24 , FBRD R OFFSET =
 (uint32_t) 0x28, LCRH_R_OFFSET = (uint32_t) 0x2C,
 CTL_R_OFFSET = (uint32_t) 0x30 , CC_R_OFFSET = (uint32_t) 0xFC8 }
enum UART_Num_t {
 UARTO, UART1, UART2, UART3,
 UART4, UART5, UART6, UART7}
```

Functions

```
    UART_t * UART_Init (GPIO_Port_t *port, UART_Num_t uartNum)
```

Initialize the specified UART peripheral.

unsigned char UART ReadChar (UART t *uart)

Read a single ASCII character from the UART.

• void UART_WriteChar (UART_t *uart, unsigned char input_char)

Write a single character to the UART.

void UART_WriteStr (UART_t *uart, void *input_str)

Write a C string to the UART.

void UART_WriteInt (UART_t *uart, int32_t n)

Write a 32-bit unsigned integer the UART.

void UART_WriteFloat (UART_t *uart, double n, uint8_t num_decimals)

Write a floating-point number the UART.

4.1.8.1 Detailed Description

Functions for UART-based communication.

4.1.8.2 Function Documentation

UART_Init()

Initialize the specified UART peripheral.

Parameters

in	port	GPIO port to use.
in	uartNum	UART number. Should be either one of the enumerated constants or an int in range [0, 7].
out	UART⊷	(Pointer to) initialized UART peripheral.
	_ <i>t</i> *	

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)

UART_ReadChar()

Read a single ASCII character from the UART.

Parameters

in <i>uart</i>		UART to read from.
out	unsigned	char ASCII character from sender.

UART_WriteChar()

Write a single character to the UART.

Parameters

in	uart	UART to read from.
in	input_char	ASCII character to send.

UART_WriteFloat()

Write a floating-point number the UART.

Parameters

in	uart	UART to read from.
in	n	Floating-point number to be converted and transmitted.
in	num_decimals	Number of digits after the decimal point to include.

UART_WriteInt()

Write a 32-bit unsigned integer the UART.

Parameters

in	uart	UART to read from.	
in	n	Unsigned 32-bit int to be converted and transmitted.	

UART_WriteStr()

Write a C string to the UART.

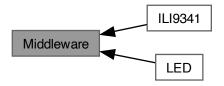
Parameters

in	uart	UART to read from.
in	input_str	Array of ASCII characters.

4.2 Middleware

High-level device driver modules.

Collaboration diagram for Middleware:



Modules

- ILI9341
- LED

4.2.1 Detailed Description

High-level device driver modules.

These modules contain functions for interfacing with external devices/peripherals via the use of low-level drivers.

4.2.2 ILI9341

Collaboration diagram for ILI9341:



Files

• file ILI9341.c

Source code for ILI9341 module.

• 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

Enumerations

```
    enum Cmd_t {
    NOP = 0x00 , SWRESET = 0x01 , SPLIN = 0x10 , SPLOUT = 0x11 ,
    PTLON = 0x12 , NORON = 0x13 , DINVOFF = 0x20 , DINVON = 0x21 ,
    CASET = 0x2A , PASET = 0x2B , RAMWR = 0x2C , DISPOFF = 0x28 ,
    DISPON = 0x29 , PLTAR = 0x30 , VSCRDEF = 0x33 , MADCTL = 0x36 ,
    VSCRSADD = 0x37 , IDMOFF = 0x38 , IDMON = 0x39 , PIXSET = 0x3A ,
    FRMCTR1 = 0xB1 , FRMCTR2 = 0xB2 , FRMCTR3 = 0xB3 , PRCTR = 0xB5 ,
    IFCTL = 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 isSleeping)

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 isNormal, bool isFullColors)

Set the display area and color expression.

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_setScrollArea (uint16_t topFixedArea, uint16_t vertScrollArea, uint16_t bottFixedArea)

Set the vertical scrolling area of the display. The sum of the three parameters should be equal to the max number of rows $NUM_ROWS = 320$.

void ILI9341_setScrollStart (uint16_t startRow)

Set the start row for vertical scrolling.

void ILI9341_setMemAccessCtrl (bool areRowsFlipped, bool areColsFlipped, bool areRowsAndCols
 — Switched, 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_setFrameRateNorm (uint8_t divisionRatio, uint8_t clocksPerLine)

TODO: Write brief.

• void ILI9341_setFrameRateIdle (uint8_t divisionRatio, uint8_t clocksPerLine)

TODO: Write brief.

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 startRow, uint16 t endRow)

not using backlight, so these aren't necessary

• void ILI9341_setColAddress (uint16_t startCol, uint16_t endCol)

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_writePixel (uint8_t red, uint8_t green, uint8_t blue, bool is_16bit)

Write a single pixel to frame memory.

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

TODO: Write.

4.2.2.1 Detailed Description

Functions for interfacing an ILI9341-based 240RGBx320 LCD via Serial Peripheral Interface (SPI).

4.2.2.2 Enumeration Type Documentation

Cmd_t

enum Cmd_t

Enumerator

SWRESET	No Operation.
SPLIN	Software Reset.
SPLOUT	Enter Sleep Mode.
PTLON	Sleep Out (i.e. Exit Sleep Mode)
NORON	Partial Display Mode ON.
DINVOFF	Normal Display Mode ON.
DINVON	Display Inversion OFF.
CASET	Display Inversion ON.
PASET	Column Address Set.
RAMWR	Page Address Set.
DISPOFF	Memory Write.
DISPON	Display OFF.
PLTAR	Display ON.
VSCRDEF	Partial Area.
MADCTL	Vertical Scrolling Definition.
VSCRSADD	Memory Access Control.
IDMOFF	Vertical Scrolling Start Address.
IDMON	Idle Mode OFF.
PIXSET	Idle Mode ON.
FRMCTR1	Pixel Format Set.
FRMCTR2	Frame Rate Control Set (Normal Mode)
FRMCTR3	Frame Rate Control Set (Idle Mode)
PRCTR	Frame Rate Control Set (Partial Mode)
IFCTL	Blanking Porch Control.

4.2.2.3 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.

ILI9341_resetSoft()

Perform a software reset of the LCD driver.

the driver needs 5 [ms] before another command

ILI9341_setColAddress()

Sets the start/end rows to be written to.

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

Parameters

startCol	0 <= startCol <= endCol
endCol	startCol <= endCol < 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.

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 description

ILI9341_setDispMode()

```
void ILI9341_setDispMode (
                bool isNormal,
                bool isFullColors )
```

Set the display area and color expression.

```
Normal mode is the default and allows output to the full display area. Partial mode should be activated after calling 'ILI9341_setPartialArea()'.

Setting 'isFullColors' to 'false' restricts the color expression to 8 colors, determined by the MSB of the R/G/B values.
```

Parameters

isNormal	true for normal mode, false for partial mode
isFullColors	true for full colors, false for 8 colors

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, f	alse to turn OFF
----------------------------	-------------------------

TODO: Write description

ILI9341_setFrameRateIdle()

TODO: Write brief.

TODO: Write description

ILI9341_setFrameRateNorm()

TODO: Write brief.

TODO: Write description

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		flips value of corresponding MADCTL bit
WEMODE	0		overflowing pixel data is not ignored
EPF[1:0]	5:4	4	controls 16 to 18-bit pixel data conversion
MDT[1:0]	1:0	1	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.

ILI9341_setMemAccessCtrl()

Set how data is converted from memory to display.

Parameters

in	areRowsFlipped	
in	areColsFlipped	
in	areRowsAndColsSwitched	
in	isVertRefreshFlipped	
in	isColorOrderFlipped	
in	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	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.

ILI9341_setPartialArea()

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

Parameters

rowStart	
rowEnd	

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

startRow	0 <= startRow <= endRow
endRow	startRow <= endRow < 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_setScrollArea()

Set the vertical scrolling area of the display. The sum of the three parameters should be equal to the max number of rows $NUM_ROWS = 320$.

Parameters

topFixedArea	Number of rows fixed at the top of the screen.
vertScrollArea	Number of rows that scroll.
bottFixedArea	Number of rows fixed at the bottom of the screen.

ILI9341_setScrollStart()

Set the start row for vertical scrolling.

Parameters

startRow	Start row for scrolling. Should be >= topFixedArea	- 1

ILI9341_setSleepMode()

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

Parameters

isSleeping	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.

ILI9341_writeMemCmd()

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_writePixel()).

ILI9341_writePixel()

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	

4.2.3 LED

Collaboration diagram for LED:



Files

• file Led.c

Source code for LED module.

• file Led.h

Interface for LED module.

Data Structures

• struct Led_t

Macros

• #define LED_POOL_SIZE 3

Functions

4.2.3.1 Detailed Description

Functions for driving light-emitting diodes (LEDs) via GPIO.

4.2.3.2 Function Documentation

Led_GetPin()

Get the GPIO pin associated with the LED.

Parameters

in	led	Pointer to LED data structure.
out	<i>GPIO</i> _⇔	GPIO pin associated with the LED.
	Pin_t	

Led_GetPort()

Get the GPIO port associated with the LED.

Parameters

in	led	Pointer to LED data structure.
out	GPIO_Port⊷	Pointer to a GPIO port data structure.
	_t*	

Led_Init()

Initialize a light-emitting diode (LED) as an Led_t.

Parameters

in	gpioPort	Pointer to a struct representing a GPIO port.
in	pin	GPIO pin to use.
out	Led_t*	Pointer to LED data structure.

Led_isOn()

```
bool Led_isOn (
          Led_t * led )
```

Check the LED's status.

Parameters

in	led	Pointer to LED data structure.
out	true	the LED is ON.
out	false	the LED is OFF.

Led_Toggle()

Toggle the LED (i.e. $\texttt{OFF} \mathrel{->} \texttt{ON}$ or $\texttt{ON} \mathrel{->} \texttt{OFF}).$

Parameters

in	led	Pointer to LED data structure.

Led_TurnOff()

```
void Led_TurnOff (
    Led_t * led )
```

Turn the LED ${\tt OFF}.$

Parameters

in	led	Pointer to LED data structure.

Led_TurnOn()

Turn the LED ON.

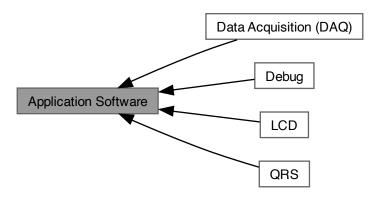
Parameters

in led Pointer to LED data structure

4.3 Application Software

Application-specific software modules.

Collaboration diagram for Application Software:



Modules

- Data Acquisition (DAQ)
- Debug
- LCD
- QRS

4.3.1 Detailed Description

Application-specific software modules.

These modules contain functions specifically built for this project's purposes.

4.3.2 Data Acquisition (DAQ)

Collaboration diagram for Data Acquisition (DAQ):



Files

• file DAQ.c

Source code for DAQ module.

· file DAQ.h

Application software for handling data acquision (DAQ) functions.

· file lookup.c

Lookup table source code.

file lookup.h

Lookup table API.

Macros

#define SAMPLING_PERIOD_MS 5

sampling period in ms ($T_s=1/f_s$)

- #define LOOKUP_ADC_MAX (float32_t) 5.5
- #define LOOKUP_ADC_MIN (float32_t)(-5.5)

Typedefs

typedef arm_biquad_casd_df1_inst_f32 filt_t

Enumerations

• enum { NUM_FILT_STAGES = 10 , NUM_FILT_COEFFS = NUM_FILT_STAGES * 5 , STATE_BUFF_SIZE = NUM_FILT_STAGES * 4 }

Functions

void DAQ_Init (void)

Initialize the data acquisition module, including the input filter and timer interrupt-based analog-to-digital conversion (ADC) @ $f_s = 200[Hz]$.

float32_t DAQ_Filter (volatile float32_t inputSample)

Filter an input sample using a 40 [Hz] low pass filter and a 60 [Hz] notch filter.

const float32_t * Lookup_GetPtr_ADC (void)

Return a pointer to the ADC lookup table.

4.3.2.1 Detailed Description

Module for managing data acquisition (DAQ) functions.

4.3.2.2 Function Documentation

DAQ_Filter()

Filter an input sample using a 40 [Hz] low pass filter and a 60 [Hz] notch filter.

Parameters

	in	inputSample	Raw input sample in range $[-5.5, 5.5]$.5)	[V].
ſ	out	float32_t	Filtered output sample.		

Lookup_GetPtr_ADC()

Return a pointer to the ADC lookup table.

Returns

const float32_t*

4.3.3 Debug

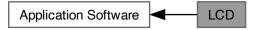
Collaboration diagram for Debug:



Module for debugging functions, including serial output and assertion.

4.3.4 LCD

Collaboration diagram for LCD:



Files

file LCD.c

Source code for LCD module.

· file LCD.h

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

Enumerations

enum { X_MAX = NUM_ROWS , Y_MAX = NUM_COLS }

Color Setting Functions

enum {

```
 \begin{array}{l} \textbf{LCD\_BLACK} = 0x00 \text{ , } \textbf{LCD\_RED} = 0x04 \text{ , } \textbf{LCD\_GREEN} = 0x02 \text{ , } \textbf{LCD\_BLUE} = 0x01 \text{ , } \\ \textbf{LCD\_YELLOW} = 0x06 \text{ , } \textbf{LCD\_CYAN} = 0x03 \text{ , } \textbf{LCD\_PURPLE} = 0x05 \text{ , } \textbf{LCD\_WHITE} = 0x07 \text{ , } \\ \textbf{LCD\_BLACK\_INV} = \textbf{LCD\_WHITE} \text{ , } \textbf{LCD\_RED\_INV} = \textbf{LCD\_CYAN} \text{ , } \textbf{LCD\_GREEN\_INV} = \textbf{LCD\_PURPLE} \text{ , } \\ \textbf{LCD\_BLUE\_INV} = \textbf{LCD\_YELLOW} \text{ , } \\ \textbf{LCD\_YELLOW\_INV} = \textbf{LCD\_BLUE} \text{ , } \textbf{LCD\_CYAN\_INV} = \textbf{LCD\_RED} \text{ , } \textbf{LCD\_PURPLE\_INV} = \textbf{LCD\_GREEN} \text{ , } \\ \textbf{LCD\_WHITE\_INV} = \textbf{LCD\_BLACK} \end{array} \right.
```

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 of each inputted value are used.

void LCD_setColor_3bit (uint8_t color_code)

Set the color value via a 3-bit code.

Init./Config. Functions

void LCD_Init (void)

Initialize the LCD driver and its internal independencies.

void LCD_setOutputMode (bool isOn)

Toggle display output ON or OFF (OFF by default). Turning output OFF stops the LCD driver chip from writing to the display, and also blanks out the display completely.

void LCD_toggleOutput (void)

Toggle display output ON or OFF (OFF by default).

void LCD setColorInversionMode (bool isOn)

Turn color inversion ON or OFF (OFF by default).

void LCD_toggleColorInversion (void)

Toggle color inversion ON or OFF (OFF by default).

void LCD setColorDepth (bool is 16bit)

Set the color depth to 16-bit or 18-bit. 16-bit color depth allows for only \sim 65K colors, but only needs 2 data transfers. 18-bit color depth allows for \sim 262K colors, but requires 3 transfers.

void LCD_toggleColorDepth (void)

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

Drawing Area Definition Functions

```
    void LCD_setArea (uint16_t x1_new, uint16_t x2_new, uint16_t y1_new, uint16_t y2_new)
    Set the area of the display to be written to. 0 <= x1 <= x2 < X_MAX 0 <= y1 <= y2 < Y_MAX</li>
```

void LCD_setX (uint16_t x1_new, uint16_t x2_new)

Set only new x-coordinates to be written to. $0 <= x1 <= x2 < X_MAX$

void LCD_setY (uint16_t y1_new, uint16_t y2_new)

Set only new y-coordinates to be written to. $0 <= y1 <= y2 < Y_MAX$

Drawing Functions

• void LCD Draw (void)

Draw on the LCD display. Call this function after setting the drawable area via LCD_setArea(), or after individually calling LCD_setX() and/or LCD_setY().

• void LCD_Fill (void)

Fill the display with a single color.

void LCD drawHoriLine (uint16 t yCenter, uint16 t lineWidth)

Draw a horizontal line across the entire display.

void LCD drawVertLine (uint16 t xCenter, uint16 t lineWidth)

Draw a vertical line across the entire display.

void LCD drawRectangle (uint16 t x1, uint16 t dx, uint16 t y1, uint16 t dy, bool isFilled)

Draw a rectangle of size $dx \times dy$ onto the display. The bottom-left corner will be located at (x1, y1).

void LCD_graphSample (uint16_t x1, uint16_t dx, uint16_t y1, uint16_t dy, uint16_t y_min, uint16_t y_max, uint16_t color_code)

Draw a rectangle of size dx x dy and blank out all other pixels between y_min and y_max.

4.3.4.1 Detailed Description

Module for displaying graphs on an LCD via the ILI9341 module.

4.3.4.2 Function Documentation

LCD_Draw()

```
void LCD_Draw (
     void )
```

Draw on the LCD display. Call this function after setting the drawable area via LCD_setArea(), or after individually calling LCD_setX() and/or LCD_setY().

LCD drawHoriLine()

Draw a horizontal line across the entire display.

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

See also

LCD_drawVertLine, LCD_drawRectangle()

LCD_drawRectangle()

Draw a rectangle of size dx x dy onto the display. The bottom-left corner will be located at (x1, y1).

Parameters

x1	lowest (left-most) x-coordinate
dx	length (horizontal distance) of the rectangle
y1	lowest (bottom-most) y-coordinate
dy	height (vertical distance) of the rectangle
isFilled	true to fill the rectangle, false to leave it unfilled

LCD_drawVertLine()

Draw a vertical line across the entire display.

Parameters

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

See also

LCD_drawHoriLine, LCD_drawRectangle()

LCD_graphSample()

```
void LCD_graphSample (
```

```
uint16_t x1,
uint16_t dx,
uint16_t y1,
uint16_t dy,
uint16_t y_min,
uint16_t y_max,
uint16_t color_code )
```

Draw a rectangle of size dx x dy and blank out all other pixels between y_min and y_max.

Parameters

x1	lowest (left-most) x-coordinate
dx	length (horizontal distance) of the column
y1	y-coordinate of the pixel's bottom side
dy	height (vertical distance) of the pixel
y_min	lowest (bottom-most) y-coordinate
y_max	highest (top-most) y-coordinate
color_code	3-bit color code

TODO: Write description

LCD_setArea()

Set the area of the display to be written to. 0 <= x1 <= x2 < X_MAX 0 <= y1 <= y2 < Y_MAX

Parameters

x1_new	left-most x-coordinate
x2_new	right-most x-coordinate
y1_new	lowest y-coordinate
y2_new	highest y-coordinate

See also

```
LCD_setX(), LCD_setY()
```

LCD_setColor()

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

R_val	5-bit ([0-31]) R value; 6-bit ([0-63]) if color depth is 18-	
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	

See also

```
LCD_setColorDepth(), LCD_toggleColorDepth(), LCD_setColor_3bit()
```

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 correspond to R, G, and B values, respectively.	
--	------------	--	--

See also

```
LCD_setColorDepth(), LCD_toggleColorDepth(), LCD_setColor()
```

This is simply a convenience function for setting the color using the enum values defined in the header file. The ones with the $_{{\tt INV}}$ suffix should be used when the display colors are inverted.

hex	binary	macro
0x00	000	LCD_BLACK
0x01	001	LCD_BLUE
0x02	010	LCD_GREEN
0x03	011	LCD_CYAN
0x04	100	LCD_RED
0x05	101	LCD_PURPLE
0x06	110	LCD_YELLOW
0x07	111	LCD_WHITE

LCD_setColorDepth()

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

Set the color depth to 16-bit or 18-bit. 16-bit color depth allows for only \sim 65K colors, but only needs 2 data transfers. 18-bit color depth allows for \sim 262K colors, but requires 3 transfers.

See also

```
LCD_toggleColorDepth(), LCD_setColor(), LCD_setColor_3bit()
```

LCD_setColorInversionMode()

Turn color inversion ON or OFF (OFF by default).

Parameters

	in	isOn	true to invert colors, false to use regular colors	
--	----	------	--	--

See also

```
LCD_toggleColorInversion(), LCD_setColor(), LCD_setColor_3bit()
```

LCD_setOutputMode()

```
void LCD_setOutputMode (
          bool isOn )
```

Toggle display output ON or OFF (OFF by default). Turning output OFF stops the LCD driver chip from writing to the display, and also blanks out the display completely.

Parameters

```
in isOn true to turn display output ON, false to turn OFF
```

See also

LCD_toggleOutput()

LCD_setX()

Set only new x-coordinates to be written to. 0 $<= x1 <= x2 < X_MAX$

x1_new	left-most x-coordinate
x2_new	right-most x-coordinate

See also

```
LCD_setY(), LCD_setArea()
```

LCD_setY()

```
void LCD_setY ( \label{lcd_var} \mbox{uint16\_t } y1\_new, \\ \mbox{uint16\_t } y2\_new \mbox{)}
```

Set only new y-coordinates to be written to. 0 <= y1 <= y2 < Y_MAX

Parameters

y1_new	lowest y-coordinate
y2_new	highest y-coordinate

See also

```
LCD_setX(), LCD_setArea()
```

LCD_toggleColorDepth()

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

See also

```
LCD_setColorDepth(), LCD_setColor(), LCD_setColor_3bit()
```

LCD_toggleColorInversion()

```
\begin{tabular}{ll} \beg
```

Toggle color inversion ${\tt ON}$ or ${\tt OFF}$ (OFF by default).

See also

```
LCD\_setColorInversionMode(),\ LCD\_setColor(),\ LCD\_setColor\_3bit()
```

LCD_toggleOutput()

Toggle display output \mathtt{ON} or \mathtt{OFF} (\mathtt{OFF} by default).

See also

LCD_setOutputMode()

4.3.5 QRS

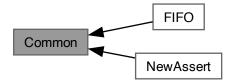
Collaboration diagram for QRS:



Module for analyzing ECG data to determine heart rate.

4.4 Common

Collaboration diagram for Common:



Modules

- FIFO
- NewAssert

4.4 Common 41

Files

· file NewAssert.c

Source code for custom assert implementation.

· file NewAssert.h

Header file for custom assert implementation.

Functions

· void Assert (bool condition)

Custom assert implementation that is more lightweight than the one from newlib.

4.4.1 Detailed Description

Modules that are used by multiple layers and/or don't fit into any one layer.

4.4.2 Function Documentation

Assert()

```
void Assert (
          bool condition )
```

Custom assert implementation that is more lightweight than the one from newlib.

Parameters

in	condition	Conditional to test. Causes an infinite loop if false.

4.4.3 FIFO

Collaboration diagram for FIFO:



Files

• file FIFO.c

Source code for FIFO buffer module.

• file FIFO.h

FIFO buffer data structure.

Data Structures

struct FIFO_t

Macros

• #define FIFO_POOL_SIZE 5

Functions

volatile FIFO_t * FIFO_Init (volatile uint32_t buffer[], const uint32_t N)
 Initialize a FIFO buffer of length N.

Basic Operations

```
• void FIFO_Put (volatile FIFO_t *fifo, const uint32_t val)
```

Add a value to the end of the buffer.

• uint32_t FIFO_Get (volatile FIFO_t *fifo)

Remove the first value of the buffer.

void FIFO TransferOne (volatile FIFO t *srcFifo, volatile FIFO t *destFifo)

Transfer a value from one FIFO buffer to another.

Bulk Removal

```
    void FIFO_Flush (volatile FIFO_t *fifo, uint32_t outputBuffer[])
```

Empty the FIFO buffer's contents into an array.

• void FIFO_Reset (volatile FIFO_t *fifo)

Reset the FIFO buffer.

• void FIFO_TransferAll (volatile FIFO_t *srcFifo, volatile FIFO_t *destFifo)

Transfer the contents of one FIFO buffer to another.

Peeking

• uint32_t FIFO_PeekOne (volatile FIFO_t *fifo)

See the first element in the FIFO without removing it.

void FIFO_PeekAll (volatile FIFO_t *fifo, uint32_t outputBuffer[])

See the FIFO buffer's contents without removing them.

Status Checks

• bool FIFO_isFull (volatile FIFO_t *fifo)

Check if the FIFO buffer is full.

• bool FIFO_isEmpty (volatile FIFO_t *fifo)

Check if the FIFO buffer is empty.

uint32_t FIFO_getCurrSize (volatile FIFO_t *fifo)

Get the current size of the FIFO buffer.

4.4 Common 43

4.4.3.1 Detailed Description

Module for using the "first-in first-out (FIFO) buffer" data structure.

4.4.3.2 Function Documentation

FIFO_Flush()

Empty the FIFO buffer's contents into an array.

Parameters

fifo	Pointer to source FIFO buffer.
outputBuffer	Array to output values to. Should be the same length as the FIFO buffer.

FIFO_Get()

Remove the first value of the buffer.

Parameters

fifo	Pointer to FIFO object
1110	Fointer to FIFO object

Returns

First sample in the FIFO.

FIFO_getCurrSize()

Get the current size of the FIFO buffer.

Parameters

fifo Pointer to the FIFO buffer.

FIFO_Init()

Initialize a FIFO buffer of length ${\tt N}.$

Parameters

buffer	Array of size ${\tt N}$ to be used as FIFO buffer
N	Length of buffer. Usable length is ${\tt N}-1$.

Returns

pointer to the FIFO buffer

TODO: Add details

FIFO_isEmpty()

Check if the FIFO buffer is empty.

Parameters

Return values

true	The buffer is empty.
false	The buffer is not empty.

FIFO_isFull()

Check if the FIFO buffer is full.

4.4 Common 45

Return values

true	The buffer is full.
false	The buffer is not full.

FIFO_PeekAll()

See the FIFO buffer's contents without removing them.

Parameters

fifo	Pointer to FIFO object
outputBuffer	Array to output values to. Should be the same length as the FIFO buffer.

FIFO_PeekOne()

See the first element in the FIFO without removing it.

Parameters

fifo	Pointer to FIFO object
------	------------------------

Returns

First sample in the FIFO.

FIFO_Put()

Add a value to the end of the buffer.

fifo	Pointer to FIFO object
val	last value in the buffer

FIFO_Reset()

```
void FIFO_Reset ( \label{eq:volatile} \mbox{volatile FIFO\_t * } fifo\ )
```

Reset the FIFO buffer.

Parameters

in fifo Pointer to FI	FO buffer.
-----------------------	------------

FIFO_TransferAll()

Transfer the contents of one FIFO buffer to another.

Parameters

srcFifo	Pointer to source FIFO buffer.
destFifo	Pointer to destination FIFO buffer.

FIFO_TransferOne()

Transfer a value from one FIFO buffer to another.

srcFifo	Pointer to source FIFO buffer.
destFifo	Pointer to destination FIFO buffer.

4.4.4 NewAssert

Collaboration diagram for NewAssert:



Module for using a custom assert implementation.

5 Data Structure Documentation

5.1 FIFO_t Struct Reference

Data Fields

- volatile uint32_t * buffer
 - (pointer to) array to use as FIFO buffer
- volatile uint32_t N
 - length of buffer
- volatile uint32_t front_idx
 - idx of front of FIFO
- volatile uint32_t back_idx

idx of back of FIFO

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

• FIFO.c

5.2 GPIO_Port_t Struct Reference

Data Fields

- const uint32_t BASE_ADDRESS
- const uint32_t DATA_REGISTER
- · bool islnit

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

• GPIO.c

5.3 Led_t Struct Reference

Data Fields

```
• GPIO_Port_t * GPIO_PORT_PTR
     pointer to GPIO port data structure
• GPIO_Pin_t GPIO_PIN
     GPIO pin number.

    bool is_ON

     state indicator
```

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

· Led.c

5.4 UART_t Struct Reference

Data Fields

- · const uint32 t BASE ADDRESS · register_t const FLAG_R_ADDRESS GPIO_Port_t * GPIO_PORT pointer to GPIO port data structure · GPIO Pin tRX PIN NUM

GPIO pin number.

- GPIO_Pin_t TX_PIN_NUM GPIO pin number.
- · bool islnit

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

• UART.c

File Documentation

6.1 DAQ.c File Reference

Source code for DAQ module.

```
#include "DAQ.h"
#include "ADC.h"
#include "Timer.h"
#include "FIFO.h"
#include "NewAssert.h"
#include "arm_math_types.h"
#include "dsp/filtering_functions.h"
#include "lookup.h"
#include "tm4c123gh6pm.h"
#include <math.h>
#include <stdbool.h>
#include <stdint.h>
```

6.2 DAQ.h File Reference 49

Macros

#define SAMPLING PERIOD MS 5

```
sampling period in ms ( T_s = 1/f_s )
```

Typedefs

typedef arm_biquad_casd_df1_inst_f32 filt_t

Enumerations

```
    enum { NUM_FILT_STAGES = 10 , NUM_FILT_COEFFS = NUM_FILT_STAGES * 5 , STATE_BUFF_SIZE = NUM_FILT_STAGES * 4 }
```

Functions

· void DAQ_Init (void)

Initialize the data acquisition module, including the input filter and timer interrupt-based analog-to-digital conversion (ADC) @ $f_s = 200[Hz]$.

• float32_t DAQ_Filter (volatile float32_t inputSample)

Filter an input sample using a 40 [Hz] low pass filter and a 60 [Hz] notch filter.

6.1.1 Detailed Description

Source code for DAQ module.

Author

Bryan McElvy

6.2 DAQ.h File Reference

Application software for handling data acquision (DAQ) functions.

```
#include "ADC.h"
#include "Timer.h"
#include "FIFO.h"
#include "arm_math_types.h"
#include "dsp/filtering_functions.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
```

Functions

• void DAQ_Init (void)

Initialize the data acquisition module, including the input filter and timer interrupt-based analog-to-digital conversion (ADC) @ $f_s = 200[Hz]$.

• float32_t DAQ_Filter (volatile float32_t inputSample)

Filter an input sample using a 40 [Hz] low pass filter and a 60 [Hz] notch filter.

6.2.1 Detailed Description

Application software for handling data acquision (DAQ) functions.

Author

Bryan McElvy

6.3 Debug.h File Reference

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

```
#include "UART.h"
#include "NewAssert.h"
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Enumerations

```
enum msg_t {
    START_MSG, DAQ_INIT, QRS_INIT, LCD_INIT,
    ASSERT_FALSE }
```

Functions

void Debug_Init (void)

Init. the Debug module and send a start message to the port.

void Debug_SendMsg (void *message)

Send a message to the serial port.

void Debug_SendFromList (msg_t msg)

Send a message from the message list.

void Debug_WriteFloat (double value)

Write a floating-point value to the serial port.

void Debug_Assert (bool condition)

Stops program if condition is true. Useful for bug detection during debugging.

6.3.1 Detailed Description

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

Author

Bryan McElvy

6.3.2 Function Documentation

Debug_Assert()

```
void Debug_Assert (
          bool condition )
```

Stops program if ${\tt condition}$ is ${\tt true}.$ Useful for bug detection during debugging.

6.4 LCD.c File Reference 51

Parameters

condition

Debug_SendFromList()

Send a message from the message list.

Parameters

in	msg	Message to send.
----	-----	------------------

Debug_SendMsg()

Send a message to the serial port.

Parameters

```
message (Pointer to) array of ASCII characters.
```

Debug_WriteFloat()

Write a floating-point value to the serial port.

Parameters

in value Floating-point value.

6.4 LCD.c File Reference

Source code for LCD module.

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

```
#include "NewAssert.h"
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Functions

Init./Config. Functions

• void LCD_Init (void)

Initialize the LCD driver and its internal independencies.

void LCD_setOutputMode (bool isOn)

Toggle display output ON or OFF (OFF by default). Turning output OFF stops the LCD driver chip from writing to the display, and also blanks out the display completely.

void LCD_toggleOutput (void)

Toggle display output ON or OFF (OFF by default).

void LCD_setColorInversionMode (bool isOn)

Turn color inversion ON or OFF (OFF by default).

void LCD toggleColorInversion (void)

Toggle color inversion ON or OFF (OFF by default).

void LCD_setColorDepth (bool is_16bit)

Set the color depth to 16-bit or 18-bit. 16-bit color depth allows for only \sim 65K colors, but only needs 2 data transfers. 18-bit color depth allows for \sim 262K colors, but requires 3 transfers.

void LCD_toggleColorDepth (void)

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

Drawing Area Definition Functions

```
    void LCD_setArea (uint16_t x1_new, uint16_t x2_new, uint16_t y1_new, uint16_t y2_new)
```

Set the area of the display to be written to. $0 <= x1 <= x2 < X_MAX 0 <= y1 <= y2 < Y_MAX$

void LCD_setX (uint16_t x1_new, uint16_t x2_new)

Set only new x-coordinates to be written to. $0 <= x1 <= x2 < X_MAX$

void LCD_setY (uint16_t y1_new, uint16_t y2_new)

Set only new y-coordinates to be written to. $0 <= y1 <= y2 < Y_MAX$

Color Setting Functions

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 of each inputted value are used.

void LCD setColor 3bit (uint8 t color code)

Set the color value via a 3-bit code.

Drawing Functions

void LCD_Draw (void)

Draw on the LCD display. Call this function after setting the drawable area via LCD_setArea(), or after individually calling LCD_setX() and/or LCD_setY().

void LCD_Fill (void)

Fill the display with a single color.

void LCD drawHoriLine (uint16 t yCenter, uint16 t lineWidth)

Draw a horizontal line across the entire display.

void LCD drawVertLine (uint16 t xCenter, uint16 t lineWidth)

Draw a vertical line across the entire display.

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

Draw a rectangle of size $dx \times dy$ onto the display. The bottom-left corner will be located at (x1, y1).

void LCD_graphSample (uint16_t x1, uint16_t dx, uint16_t y1, uint16_t dy, uint16_t y_min, uint16_t y_max, uint16_t color code)

Draw a rectangle of size dx x dy and blank out all other pixels between y_min and y_max .

6.5 LCD.h File Reference 53

6.4.1 Detailed Description

Source code for LCD module.

Author

Bryan McElvy

6.5 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 "NewAssert.h"
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Enumerations

enum { X_MAX = NUM_ROWS , Y_MAX = NUM_COLS }

Functions

Init./Config. Functions

void LCD Init (void)

Initialize the LCD driver and its internal independencies.

void LCD_setOutputMode (bool isOn)

Toggle display output ON or OFF (OFF by default). Turning output OFF stops the LCD driver chip from writing to the display, and also blanks out the display completely.

void LCD_toggleOutput (void)

Toggle display output ON or OFF (OFF by default).

void LCD_setColorInversionMode (bool isOn)

Turn color inversion ON or OFF (OFF by default).

• void LCD_toggleColorInversion (void)

Toggle color inversion ON or OFF (OFF by default).

• void LCD_setColorDepth (bool is_16bit)

Set the color depth to 16-bit or 18-bit. 16-bit color depth allows for only \sim 65K colors, but only needs 2 data transfers. 18-bit color depth allows for \sim 262K colors, but requires 3 transfers.

void LCD_toggleColorDepth (void)

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

Drawing Area Definition Functions

```
• void LCD_setArea (uint16_t x1_new, uint16_t x2_new, uint16_t y1_new, uint16_t y2_new)
```

Set the area of the display to be written to. $0 <= x1 <= x2 < X_MAX 0 <= y1 <= y2 < Y_MAX$

void LCD_setX (uint16_t x1_new, uint16_t x2_new)

Set only new x-coordinates to be written to. $0 <= x1 <= x2 < X_MAX$

void LCD_setY (uint16_t y1_new, uint16_t y2_new)

Set only new y-coordinates to be written to. $0 <= y1 <= y2 < Y_MAX$

Drawing Functions

void LCD Draw (void)

Draw on the LCD display. Call this function after setting the drawable area via LCD_setArea(), or after individually calling LCD_setX() and/or LCD_setY().

void LCD_Fill (void)

Fill the display with a single color.

void LCD drawHoriLine (uint16 t yCenter, uint16 t lineWidth)

Draw a horizontal line across the entire display.

void LCD_drawVertLine (uint16_t xCenter, uint16_t lineWidth)

Draw a vertical line across the entire display.

• void LCD_drawRectangle (uint16_t x1, uint16_t dx, uint16_t y1, uint16_t dy, bool isFilled)

Draw a rectangle of size $dx \times dy$ onto the display. The bottom-left corner will be located at (x1, y1).

• void LCD_graphSample (uint16_t x1, uint16_t dx, uint16_t y1, uint16_t dy, uint16_t y_min, uint16_t y_max, uint16_t color_code)

Draw a rectangle of size dx x dy and blank out all other pixels between y_min and y_max.

Color Setting Functions

• enum {

```
 \begin{split} & \textbf{LCD\_BLACK} = 0x00 \text{ , } \textbf{LCD\_RED} = 0x04 \text{ , } \textbf{LCD\_GREEN} = 0x02 \text{ , } \textbf{LCD\_BLUE} = 0x01 \text{ , } \\ & \textbf{LCD\_YELLOW} = 0x06 \text{ , } \textbf{LCD\_CYAN} = 0x03 \text{ , } \textbf{LCD\_PURPLE} = 0x05 \text{ , } \textbf{LCD\_WHITE} = 0x07 \text{ , } \\ & \textbf{LCD\_BLACK\_INV} = \textbf{LCD\_WHITE} \text{ , } \textbf{LCD\_RED\_INV} = \textbf{LCD\_CYAN} \text{ , } \textbf{LCD\_GREEN\_INV} = \textbf{LCD\_PURPLE} \text{ , } \\ & \textbf{LCD\_BLUE\_INV} = \textbf{LCD\_YELLOW} \text{ , } \\ & \textbf{LCD\_YELLOW\_INV} = \textbf{LCD\_BLUE} \text{ , } \textbf{LCD\_CYAN\_INV} = \textbf{LCD\_RED} \text{ , } \textbf{LCD\_PURPLE\_INV} = \textbf{LCD\_GREEN} \text{ , } \\ & \textbf{LCD\_WHITE\_INV} = \textbf{LCD\_BLACK} \text{ } \end{split}
```

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 of each inputted value are used.

void LCD_setColor_3bit (uint8_t color_code)

Set the color value via a 3-bit code.

6.5.1 Detailed Description

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

Author

Bryan McElvy

6.6 QRS.h File Reference

QRS detection algorithm functions.

```
#include "dsp/filtering_functions_f16.h"
```

6.6.1 Detailed Description

QRS detection algorithm functions.

Author

Bryan McElvy

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

6.7 FIFO.c File Reference 55

6.7 FIFO.c File Reference

Source code for FIFO buffer module.

```
#include "FIFO.h"
#include "NewAssert.h"
#include <stdint.h>
#include <stdbool.h>
```

Data Structures

• struct FIFO_t

Functions

volatile FIFO_t * FIFO_Init (volatile uint32_t buffer[], const uint32_t N)
 Initialize a FIFO buffer of length N.

Basic Operations

```
    void FIFO_Put (volatile FIFO_t *fifo, const uint32_t val)
```

Add a value to the end of the buffer.

uint32_t FIFO_Get (volatile FIFO_t *fifo)

Remove the first value of the buffer.

• void FIFO_TransferOne (volatile FIFO_t *srcFifo, volatile FIFO_t *destFifo)

Transfer a value from one FIFO buffer to another.

Bulk Removal

```
• void FIFO_Flush (volatile FIFO_t *fifo, uint32_t outputBuffer[])
```

Empty the FIFO buffer's contents into an array.

void FIFO_Reset (volatile FIFO_t *fifo)

Reset the FIFO buffer.

void FIFO_TransferAll (volatile FIFO_t *srcFifo, volatile FIFO_t *destFifo)

Transfer the contents of one FIFO buffer to another.

Peeking

```
    uint32_t FIFO_PeekOne (volatile FIFO_t *fifo)
```

See the first element in the FIFO without removing it.

void FIFO_PeekAll (volatile FIFO_t *fifo, uint32_t outputBuffer[])

See the FIFO buffer's contents without removing them.

Status Checks

```
    bool FIFO isFull (volatile FIFO t *fifo)
```

Check if the FIFO buffer is full.

• bool FIFO_isEmpty (volatile FIFO_t *fifo)

Check if the FIFO buffer is empty.

uint32_t FIFO_getCurrSize (volatile FIFO_t *fifo)

Get the current size of the FIFO buffer.

6.7.1 Detailed Description

Source code for FIFO buffer module.

Author

Bryan McElvy

6.8 FIFO.h File Reference

FIFO buffer data structure.

```
#include "NewAssert.h"
#include <stdbool.h>
#include <stdint.h>
```

Macros

• #define FIFO_POOL_SIZE 5

Functions

volatile FIFO_t * FIFO_Init (volatile uint32_t buffer[], const uint32_t N)
 Initialize a FIFO buffer of length N.

Basic Operations

- void FIFO Put (volatile FIFO t *fifo, const uint32 t val)
 - Add a value to the end of the buffer.
- uint32_t FIFO_Get (volatile FIFO_t *fifo)

Remove the first value of the buffer.

void FIFO_TransferOne (volatile FIFO_t *srcFifo, volatile FIFO_t *destFifo)

Transfer a value from one FIFO buffer to another.

Bulk Removal

- void FIFO Flush (volatile FIFO t *fifo, uint32 t outputBuffer[])
 - Empty the FIFO buffer's contents into an array.
- void FIFO_Reset (volatile FIFO_t *fifo)

Reset the FIFO buffer.

void FIFO TransferAll (volatile FIFO t *srcFifo, volatile FIFO t *destFifo)

Transfer the contents of one FIFO buffer to another.

Peeking

- uint32 t FIFO PeekOne (volatile FIFO t *fifo)
 - See the first element in the FIFO without removing it.
- void FIFO_PeekAll (volatile FIFO_t *fifo, uint32_t outputBuffer[])

See the FIFO buffer's contents without removing them.

Status Checks

bool FIFO_isFull (volatile FIFO_t *fifo)

Check if the FIFO buffer is full.

bool FIFO_isEmpty (volatile FIFO_t *fifo)

Check if the FIFO buffer is empty.

uint32_t FIFO_getCurrSize (volatile FIFO_t *fifo)

Get the current size of the FIFO buffer.

6.8.1 Detailed Description

FIFO buffer data structure.

Author

Bryan McElvy

6.9 lookup.c File Reference

Lookup table source code.

```
#include "lookup.h"
#include "arm_math_types.h"
```

Functions

const float32_t * Lookup_GetPtr_ADC (void)
 Return a pointer to the ADC lookup table.

6.9.1 Detailed Description

Lookup table source code.

Author

Bryan McElvy

6.10 lookup.h File Reference

```
Lookup table API.
```

```
#include "arm_math_types.h"
```

Macros

- #define LOOKUP_ADC_MAX (float32_t) 5.5
- #define LOOKUP_ADC_MIN (float32_t)(-5.5)

Functions

const float32_t * Lookup_GetPtr_ADC (void)
 Return a pointer to the ADC lookup table.

6.10.1 Detailed Description

Lookup table API.

Author

Bryan McElvy

6.11 NewAssert.c File Reference

Source code for custom assert implementation.

```
#include "NewAssert.h"
#include <stdbool.h>
```

Functions

· void Assert (bool condition)

Custom assert implementation that is more lightweight than the one from newlib.

6.11.1 Detailed Description

Source code for custom assert implementation.

Author

Bryan McElvy

6.12 NewAssert.h File Reference

Header file for custom assert implementation.

```
#include <stdbool.h>
```

Functions

void Assert (bool condition)

Custom assert implementation that is more lightweight than the one from newlib.

6.12.1 Detailed Description

Header file for custom assert implementation.

Author

Bryan McElvy

6.13 ADC.c File Reference

Source code for ADC module.

```
#include "ADC.h"
#include "GPIO.h"
#include "Timer.h"
#include "lookup.h"
#include "arm_math_types.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
```

Functions

· void ADC_Init (void)

Initialize ADC0 as a single-input analog-to-digital converter.

void ADC_InterruptEnable (void)

Enable the ADC interrupt.

• void ADC_InterruptDisable (void)

Disable the ADC interrupt.

float32_t ADC_ConvertToVolts (uint16_t raw_sample)

Convert a raw ADC sample to voltage in [mV].

6.13.1 Detailed Description

Source code for ADC module.

Author

Bryan McElvy

6.14 ADC.h File Reference

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

```
#include "GPIO.h"
#include "Timer.h"
#include "lookup.h"
#include "arm_math_types.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
```

Functions

void ADC_Init (void)

Initialize ADC0 as a single-input analog-to-digital converter.

void ADC_InterruptEnable (void)

Enable the ADC interrupt.

void ADC_InterruptDisable (void)

Disable the ADC interrupt.

float32_t ADC_ConvertToVolts (uint16_t raw_sample)

Convert a raw ADC sample to voltage in [mV].

6.14.1 Detailed Description

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

Author

Bryan McElvy

6.15 GPIO.c File Reference

Source code for GPIO module.

```
#include "GPIO.h"
#include <NewAssert.h>
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Data Structures

struct GPIO Port t

Macros

• #define GPIO_NUM_PORTS 6

Typedefs

typedef volatile uint32 t * register_t

Enumerations

```
    enum {
    GPIO_PORTA_BASE_ADDRESS = (uint32_t) 0x40004000 , GPIO_PORTB_BASE_ADDRESS = (uint32 ←
    _t) 0x40005000 , GPIO_PORTC_BASE_ADDRESS = (uint32_t) 0x40006000 , GPIO_PORTD_BASE_←
    ADDRESS = (uint32_t) 0x40007000 ,
    GPIO_PORTE_BASE_ADDRESS = (uint32_t) 0x40024000 , GPIO_PORTF_BASE_ADDRESS = (uint32_t) 0x40025000 }
```

enum {GPIO |

```
 \begin{aligned} & \textbf{GPIO\_DATA\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x03FC \ , \ & \textbf{GPIO\_DIR\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0400 \ , \ & \textbf{GPIO\_IS\_R} \hookrightarrow \\ & \textbf{\_OFFSET} = (\textbf{uint}32\_t) \ 0x0404 \ , \ & \textbf{GPIO\_IBE\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0408 \ , \\ & \textbf{GPIO\_IEV\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x041C \ , \ & \textbf{GPIO\_IM\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0410 \ , \ & \textbf{GPIO\_ICR\_R\_} \hookrightarrow \\ & \textbf{OFFSET} = (\textbf{uint}32\_t) \ 0x041C \ , \ & \textbf{GPIO\_AFSEL\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0420 \ , \\ & \textbf{GPIO\_DR2R\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0500 \ , \ & \textbf{GPIO\_DR4R\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0504 \ , \ & \textbf{GPIO\_PUR\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0510 \ , \\ & \textbf{GPIO\_PDR\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0520 \ , \ & \textbf{GPIO\_DEN\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0524 \ , \\ & \textbf{GPIO\_AMSEL\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x0528 \ , \ & \textbf{GPIO\_PCTL\_R\_OFFSET} = (\textbf{uint}32\_t) \ 0x052C \ \} \end{aligned}
```

Functions

GPIO Port t * GPIO InitPort (GPIO PortName t portName)

Initialize a GPIO Port and return a pointer to its struct.

bool GPIO_isPortInit (GPIO_Port_t *gpioPort)

Check if the GPIO port is initialized.

- uint32 t GPIO_getBaseAddr (GPIO_Port_t *gpioPort)
- void GPIO_ConfigDirOutput (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Configure the direction of the specified GPIO pins. All pins are configured to INPUT by default, so this function should only be called to specify OUTPUT pins.

void GPIO_ConfigDirInput (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Configure the specified GPIO pins as INPUT pins. All pins are configured to INPUT by default, so this function is technically unnecessary, but useful for code readability.

void GPIO_ConfigPullUp (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate the specified pins' internal pull-up resistors.

void GPIO_ConfigPullDown (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate the specified pins' internal pull-down resistors.

• void GPIO ConfigDriveStrength (GPIO Port t *gpioPort, GPIO Pin t pinMask, uint8 t drive mA)

Configure the specified pins' drive strength. Pins are initialized with 2[mA] drive strength, so this is only needed for a drive strength of 4[mA] or 8[mA].

void GPIO_EnableDigital (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Enable digital I/O for the specified pins.

void GPIO DisableDigital (GPIO Port t *gpioPort, GPIO Pin t pinMask)

Disable digital I/O for the specified pins.

void GPIO_ConfigInterrupts_Edge (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask, bool risingEdge)

Configure the specified GPIO pins to trigger an interrupt on the rising or falling edge of an input.

• void GPIO_ConfigInterrupts_BothEdges (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Configure the specified GPIO pins to trigger an interrupt on both edges of an input.

void GPIO ConfigInterrupts LevelTrig (GPIO Port t *gpioPort, GPIO Pin t pinMask, bool highLevel)

Configure the specified GPIO pins to trigger an interrupt on a high level or low level pulse.

• void GPIO_ConfigNVIC (GPIO_Port_t *gpioPort, uint8_t priority)

Configure interrupts for the selected port in the NVIC.

uint8 t GPIO ReadPins (GPIO Port t *gpioPort, GPIO Pin t pinMask)

Read from the specified GPIO pin.

void GPIO_WriteHigh (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Write a 1 to the specified GPIO pins.

void GPIO_WriteLow (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Write a 0 to the specified GPIO pins.

void GPIO_Toggle (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Toggle the specified GPIO pins.

• void GPIO_ConfigAltMode (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate the alternate mode for the specified pins.

void GPIO_ConfigPortCtrl (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask, uint8_t fieldEncoding)

Specify the alternate mode to use for the specified pins.

void GPIO_ConfigAnalog (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate analog mode for the specified GPIO pins.

6.15.1 Detailed Description

Source code for GPIO module.

Author

Bryan McElvy

6.15.2 Function Documentation

GPIO_ConfigAltMode()

Activate the alternate mode for the specified pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigAnalog()

Activate analog mode for the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigDirInput()

Configure the specified GPIO pins as INPUT pins. All pins are configured to INPUT by default, so this function is technically unnecessary, but useful for code readability.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	bitMask	Bit mask corresponding to the intended ${ t INPUT}$ pin(s).

GPIO_ConfigDirOutput()

Configure the direction of the specified GPIO pins. All pins are configured to INPUT by default, so this function should only be called to specify OUTPUT pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	bitMask	Bit mask corresponding to the intended OUTPUT pin(s).

GPIO_ConfigDriveStrength()

Configure the specified pins' drive strength. Pins are initialized with 2[mA] drive strength, so this is only needed for a drive strength of 4[mA] or 8[mA].

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).
in	drive_mA	Drive strength in [mA]. Should be 2, 4, or 8 [mA].

GPIO_ConfigInterrupts_BothEdges()

Configure the specified GPIO pins to trigger an interrupt on both edges of an input.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigInterrupts_Edge()

Configure the specified GPIO pins to trigger an interrupt on the rising or falling edge of an input.

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).
in	risingEdge	true for rising edge, false for falling edge

GPIO_ConfigInterrupts_LevelTrig()

Configure the specified GPIO pins to trigger an interrupt on a high level or low level pulse.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).
in	highLevel	true for high level, false for low level

GPIO_ConfigNVIC()

Configure interrupts for the selected port in the NVIC.

Parameters

	in	gpioPort	Pointer to the specified GPIO port.
ĺ	in	priority	Priority number between 0 (highest) and 7 (lowest).

GPIO_ConfigPortCtrl()

Specify the alternate mode to use for the specified pins.

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).
in	fieldEncoding	Number corresponding to intended alternate mode.

GPIO_ConfigPullDown()

Activate the specified pins' internal pull-down resistors.

Parameters

ir	gpioPort	Pointer to the specified GPIO port.
ir	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigPullUp()

Activate the specified pins' internal pull-up resistors.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_DisableDigital()

Disable digital I/O for the specified pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_EnableDigital()

Enable digital I/O for the specified pins.

	in	gpioPort	Pointer to the specified GPIO port.
ſ	in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_InitPort()

Initialize a GPIO Port and return a pointer to its struct.

Parameters

in	portName	Name of the chosen port.
----	----------	--------------------------

Returns

 $\label{eq:GPIO_Port_t*} \textbf{GPIO_Port_t*} \ \textbf{Pointer to the GPIO port's struct}.$

GPIO_isPortInit()

Check if the GPIO port is initialized.

Parameters

	in	gpioPort	Pointer to the specified GPIO port.
	out	true	The GPIO port is initialized.
ſ	out	false	The GPIO port has not been initialized.

GPIO_ReadPins()

Read from the specified GPIO pin.

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_Toggle()

Toggle the specified GPIO pins.

Parameters

i	n	gpioPort	Pointer to the specified GPIO port.
i	n	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_WriteHigh()

Write a $\ensuremath{\mathbb{1}}$ to the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_WriteLow()

Write a 0 to the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

6.16 GPIO.h File Reference

Header file for general-purpose input/output (GPIO) device driver.

```
#include <NewAssert.h>
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Enumerations

```
enum GPIO_Pin_t {
    GPIO_PIN0 = ((uint8_t) 1) , GPIO_PIN1 = ((uint8_t) (1 << 1)) , GPIO_PIN2 = ((uint8_t) (1 << 2)) , GPIO ←
    _PIN3 = ((uint8_t) (1 << 3)) ,
    GPIO_PIN4 = ((uint8_t) (1 << 4)) , GPIO_PIN5 = ((uint8_t) (1 << 5)) , GPIO_PIN6 = ((uint8_t) (1 << 6)) ,
    GPIO_PIN7 = ((uint8_t) (1 << 7)) ,
    GPIO_ALL_PINS = ((uint8_t) (0xFF)) }
</li>
enum {
    LED_RED = GPIO_PIN1 , LED_GREEN = GPIO_PIN3 , LED_BLUE = GPIO_PIN2 , LED_YELLOW =
    (LED_RED + LED_GREEN) ,
    LED_CYAN = (LED_BLUE + LED_GREEN) , LED_PURPLE = (LED_RED + LED_BLUE) , LED_WHITE =
    (LED_RED + LED_BLUE + LED_GREEN) }

enum GPIO_PortName_t {
        A , B , C , D ,
        E , F }
```

Functions

• GPIO Port t * GPIO InitPort (GPIO PortName t portName)

Initialize a GPIO Port and return a pointer to its struct.

- uint32 t GPIO_getBaseAddr (GPIO_Port_t *gpioPort)
- bool GPIO_isPortInit (GPIO_Port_t *gpioPort)

Check if the GPIO port is initialized.

void GPIO_ConfigDirOutput (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Configure the direction of the specified GPIO pins. All pins are configured to INPUT by default, so this function should only be called to specify OUTPUT pins.

void GPIO_ConfigDirInput (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Configure the specified GPIO pins as INPUT pins. All pins are configured to INPUT by default, so this function is technically unnecessary, but useful for code readability.

• void GPIO_ConfigPullUp (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate the specified pins' internal pull-up resistors.

void GPIO_ConfigPullDown (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate the specified pins' internal pull-down resistors.

• void GPIO_ConfigDriveStrength (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask, uint8_t drive_mA)

Configure the specified pins' drive strength. Pins are initialized with 2[mA] drive strength, so this is only needed for a drive strength of 4[mA] or 8[mA].

void GPIO_EnableDigital (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Enable digital I/O for the specified pins.

void GPIO DisableDigital (GPIO Port t*gpioPort, GPIO Pin t pinMask)

Disable digital I/O for the specified pins.

• void GPIO ConfigInterrupts Edge (GPIO Port t *gpioPort, GPIO Pin t pinMask, bool risingEdge)

Configure the specified GPIO pins to trigger an interrupt on the rising or falling edge of an input.

void GPIO ConfigInterrupts BothEdges (GPIO Port t *gpioPort, GPIO Pin t pinMask)

Configure the specified GPIO pins to trigger an interrupt on both edges of an input.

• void GPIO_ConfigInterrupts_LevelTrig (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask, bool highLevel)

Configure the specified GPIO pins to trigger an interrupt on a high level or low level pulse.

• void GPIO_ConfigNVIC (GPIO_Port_t *gpioPort, uint8_t priority)

Configure interrupts for the selected port in the NVIC.

uint8 t GPIO ReadPins (GPIO Port t *gpioPort, GPIO Pin t pinMask)

Read from the specified GPIO pin.

void GPIO_WriteHigh (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Write a 1 to the specified GPIO pins.

```
    void GPIO_WriteLow (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)
```

Write a 0 to the specified GPIO pins.

• void GPIO_Toggle (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Toggle the specified GPIO pins.

void GPIO_ConfigAltMode (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate the alternate mode for the specified pins.

• void GPIO_ConfigPortCtrl (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask, uint8_t fieldEncoding)

Specify the alternate mode to use for the specified pins.

void GPIO_ConfigAnalog (GPIO_Port_t *gpioPort, GPIO_Pin_t pinMask)

Activate analog mode for the specified GPIO pins.

6.16.1 Detailed Description

Header file for general-purpose input/output (GPIO) device driver.

Author

Bryan McElvy

6.16.2 Function Documentation

GPIO_ConfigAltMode()

Activate the alternate mode for the specified pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigAnalog()

Activate analog mode for the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigDirInput()

Configure the specified GPIO pins as INPUT pins. All pins are configured to INPUT by default, so this function is technically unnecessary, but useful for code readability.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	bitMask	Bit mask corresponding to the intended INPUT pin(s).

GPIO_ConfigDirOutput()

Configure the direction of the specified GPIO pins. All pins are configured to INPUT by default, so this function should only be called to specify OUTPUT pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	bitMask	Bit mask corresponding to the intended OUTPUT pin(s).

GPIO_ConfigDriveStrength()

Configure the specified pins' drive strength. Pins are initialized with 2[mA] drive strength, so this is only needed for a drive strength of 4[mA] or 8[mA].

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).
in	drive_mA	Drive strength in [mA]. Should be 2, 4, or 8 [mA].

GPIO_ConfigInterrupts_BothEdges()

```
{\tt void \ GPIO\_ConfigInterrupts\_BothEdges \ (}
```

```
GPIO_Port_t * gpioPort,
GPIO_Pin_t pinMask )
```

Configure the specified GPIO pins to trigger an interrupt on both edges of an input.

Parameters

iı	n	gpioPort	Pointer to the specified GPIO port.
iı	n	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigInterrupts_Edge()

Configure the specified GPIO pins to trigger an interrupt on the rising or falling edge of an input.

Parameters

	in	gpioPort	Pointer to the specified GPIO port.
	in	pinMask	Bit mask corresponding to the intended pin(s).
ĺ	in	risingEdge	true for rising edge, false for falling edge

GPIO_ConfigInterrupts_LevelTrig()

Configure the specified GPIO pins to trigger an interrupt on a high level or low level pulse.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).
in	highLevel	true for high level, false for low level

GPIO_ConfigNVIC()

Configure interrupts for the selected port in the NVIC.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	priority	Priority number between 0 (highest) and 7 (lowest).

GPIO_ConfigPortCtrl()

Specify the alternate mode to use for the specified pins.

Parameters

	in	gpioPort	Pointer to the specified GPIO port.
ſ	in	pinMask	Bit mask corresponding to the intended pin(s).
Ī	in	fieldEncoding	Number corresponding to intended alternate mode.

GPIO_ConfigPullDown()

Activate the specified pins' internal pull-down resistors.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_ConfigPullUp()

Activate the specified pins' internal pull-up resistors.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_DisableDigital()

Disable digital I/O for the specified pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_EnableDigital()

Enable digital I/O for the specified pins.

Parameters

ſ	in	gpioPort	Pointer to the specified GPIO port.
ſ	in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO InitPort()

Initialize a GPIO Port and return a pointer to its struct.

Parameters

_			
	in	portName	Name of the chosen port.

Returns

GPIO_Port_t* Pointer to the GPIO port's struct.

GPIO_isPortInit()

Check if the GPIO port is initialized.

Parameters

in	gpioPort	Pointer to the specified GPIO port.	
out	true	The GPIO port is initialized.	
out	false	The GPIO port has not been initialized.	

GPIO_ReadPins()

Read from the specified GPIO pin.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_Toggle()

Toggle the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

GPIO_WriteHigh()

Write a $\ensuremath{\mathbb{1}}$ to the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

6.17 PLL.c File Reference 75

GPIO_WriteLow()

Write a 0 to the specified GPIO pins.

Parameters

in	gpioPort	Pointer to the specified GPIO port.
in	pinMask	Bit mask corresponding to the intended pin(s).

6.17 PLL.c File Reference

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

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

Functions

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

6.17.1 Detailed Description

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

Author

Bryan McElvy

6.18 PLL.h File Reference

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

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

Functions

void PLL_Init (void)

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

6.18.1 Detailed Description

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

Author

Bryan McElvy

6.19 SPI.c File Reference

Source code for SPI module.

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

Macros

- #define SPI_SET_DC() (GPIO_PORTA_DATA_R |= 0x40)
- #define **SPI_CLEAR_DC**() (GPIO_PORTA_DATA_R &= \sim (0x40))
- #define SPI_IS_BUSY (SSI0_SR_R & 0x10)
- #define SPI_TX_ISNOTFULL (SSI0_SR_R & 0x02)

Enumerations

```
    enum {
    SPI_CLK_PIN = GPIO_PIN2 , SPI_CS_PIN = GPIO_PIN3 , SPI_RX_PIN = GPIO_PIN4 , SPI_TX_PIN = GPIO_PIN5 ,
    SPI_DC_PIN = GPIO_PIN6 , SPI_RESET_PIN = GPIO_PIN7 , SPI_SSIO_PINS = (SPI_CLK_PIN | SPI_CS_PIN | SPI_RX_PIN | SPI_TX_PIN) , SPI_GPIO_PINS = (SPI_DC_PIN | SPI_RESET_PIN) ,
    SPI_ALL_PINS = (SPI_SSIO_PINS | SPI_GPIO_PINS) }
```

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 the 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.

6.20 SPI.h File Reference 77

6.19.1 Detailed Description

Source code for SPI module.

Author

Bryan McElvy

6.20 SPI.h File Reference

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

```
#include "GPIO.h"
#include "FIFO.h"
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.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 the 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.

6.20.1 Detailed Description

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

Author

Bryan McElvy

6.21 SysTick.c File Reference

Implementation details for SysTick functions.

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

```
    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.

6.21.1 Detailed Description

Implementation details for SysTick functions.

Author

Bryan McElvy

6.22 SysTick.h File Reference

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

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

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.

6.22.1 Detailed Description

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

Author

Bryan McElvy

6.23 Timer.c File Reference

Implementation for timer module.

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

Timer0A

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].

Timer1A

void Timer1A_Init (uint32_t time_ms)

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

Timer2A

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].

void Timer3A_Init (uint32_t time_ms)

Initialize Timer3A as a 32-bit, periodic, countdown timer that triggers ADC sample capture.

6.23.1 Detailed Description

Implementation for timer module.

Author

Bryan McElvy

6.24 Timer.h File Reference

Driver module for general-purpose timer modules.

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

Timer0A

• 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].

Timer1A

• void Timer1A_Init (uint32_t time_ms)

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

Timer2A

• 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].

void Timer3A_Init (uint32_t time_ms)

Initialize Timer3A as a 32-bit, periodic, countdown timer that triggers ADC sample capture.

6.24.1 Detailed Description

Driver module for general-purpose timer modules.

Author

Bryan McElvy

Timer	Function
0A	Debouncing
1A	LCD Interrupts
2A	ILI9341 Resets
3A	ADC Interrupts

6.25 UART.c File Reference

Source code for UART module.

```
#include "UART.h"
#include "GPIO.h"
#include "NewAssert.h"
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Data Structures

struct UART t

Macros

• #define ASCII_CONVERSION 0x30

Typedefs

typedef volatile uint32 t * register t

Enumerations

Functions

```
    UART_t * UART_Init (GPIO_Port_t *port, UART_Num_t uartNum)
```

Initialize the specified UART peripheral.

unsigned char UART_ReadChar (UART_t *uart)

Read a single ASCII character from the UART.

• void UART_WriteChar (UART_t *uart, unsigned char input_char)

Write a single character to the UART.

void UART_WriteStr (UART_t *uart, void *input_str)

Write a C string to the UART.

void UART WriteInt (UART t *uart, int32 t n)

Write a 32-bit unsigned integer the UART.

• void UART_WriteFloat (UART_t *uart, double n, uint8_t num_decimals)

Write a floating-point number the UART.

6.25.1 Detailed Description

Source code for UART module.

Author

6.26 UART.h File Reference

Driver module for serial communication via UART0 and UART 1.

```
#include "GPIO.h"
#include "NewAssert.h"
#include "tm4c123gh6pm.h"
#include <stdbool.h>
#include <stdint.h>
```

Enumerations

```
enum UART_Num_t {UART0 , UART1 , UART2 , UART3 ,UART4 , UART5 , UART6 , UART7 }
```

Functions

```
• UART_t * UART_Init (GPIO_Port_t *port, UART_Num_t uartNum)

Initialize the specified UART peripheral.
```

unsigned char UART_ReadChar (UART_t *uart)

Read a single ASCII character from the UART.

void UART_WriteChar (UART_t *uart, unsigned char input_char)

Write a single character to the UART.

void UART_WriteStr (UART_t *uart, void *input_str)

Write a C string to the UART.

• void UART_WriteInt (UART_t *uart, int32_t n)

Write a 32-bit unsigned integer the UART.

• void UART_WriteFloat (UART_t *uart, double n, uint8_t num_decimals)

Write a floating-point number the UART.

6.26.1 Detailed Description

Driver module for serial communication via UART0 and UART 1.

Author

Bryan McElvy

```
UARTO uses PAO and PA1, which are not broken out but can connect to a PC's serial port via USB.  
UART1 uses PBO (Rx) and PB1 (Tx), which are broken out but do not connect to a serial port.
```

6.27 main.c File Reference

Main program file for ECG-HRM.

```
#include "DAQ.h"
#include "Debug.h"
#include "LCD.h"
#include "QRS.h"
#include "PLL.h"
```

- · int main (void)
- void ADC0_SS3_Handler (void)

Interrupt service routine (ISR) for collecting ADC samples.

void Timer1A_Handler (void)

Interrupt service routine (ISR) for outputting data to the LCD.

6.27.1 Detailed Description

Main program file for ECG-HRM.

Author

Bryan McElvy

6.28 ILI9341.c File Reference

Source code for ILI9341 module.

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

Enumerations

```
    enum Cmd_t {
    NOP = 0x00 , SWRESET = 0x01 , SPLIN = 0x10 , SPLOUT = 0x11 ,
    PTLON = 0x12 , NORON = 0x13 , DINVOFF = 0x20 , DINVON = 0x21 ,
    CASET = 0x2A , PASET = 0x2B , RAMWR = 0x2C , DISPOFF = 0x28 ,
    DISPON = 0x29 , PLTAR = 0x30 , VSCRDEF = 0x33 , MADCTL = 0x36 ,
    VSCRSADD = 0x37 , IDMOFF = 0x38 , IDMON = 0x39 , PIXSET = 0x3A ,
    FRMCTR1 = 0xB1 , FRMCTR2 = 0xB2 , FRMCTR3 = 0xB3 , PRCTR = 0xB5 ,
    IFCTL = 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 isSleeping)

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 isNormal, bool isFullColors)

Set the display area and color expression.

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_setScrollArea (uint16_t topFixedArea, uint16_t vertScrollArea, uint16_t bottFixedArea)

Set the vertical scrolling area of the display. The sum of the three parameters should be equal to the max number of rows $NUM_ROWS = 320$.

void ILI9341 setScrollStart (uint16 t startRow)

Set the start row for vertical scrolling.

void ILI9341_setMemAccessCtrl (bool areRowsFlipped, bool areColsFlipped, bool areRowsAndCols
 — Switched, 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 setFrameRateNorm (uint8 t divisionRatio, uint8 t clocksPerLine)

TODO: Write brief.

• void ILI9341 setFrameRateIdle (uint8 t divisionRatio, uint8 t clocksPerLine)

TODO: Write brief.

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 startRow, uint16 t endRow)

not using backlight, so these aren't necessary

void ILI9341_setColAddress (uint16_t startCol, uint16_t endCol)

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_writePixel (uint8_t red, uint8_t green, uint8_t blue, bool is_16bit)

Write a single pixel to frame memory.

6.28.1 Detailed Description

Source code for ILI9341 module.

Author

6.29 ILI9341.h File Reference

Driver module for interfacing with an ILI9341 LCD driver.

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

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 isSleeping)

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 isNormal, bool isFullColors)

Set the display area and color expression.

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 setScrollArea (uint16 t topFixedArea, uint16 t vertScrollArea, uint16 t bottFixedArea)

Set the vertical scrolling area of the display. The sum of the three parameters should be equal to the max number of rows $NUM_ROWS = 320$.

void ILI9341 setScrollStart (uint16 t startRow)

Set the start row for vertical scrolling.

void ILI9341_setMemAccessCtrl (bool areRowsFlipped, bool areColsFlipped, bool areRowsAndCols
 — Switched, 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_setFrameRateNorm (uint8_t divisionRatio, uint8_t clocksPerLine)

TODO: Write brief.

• void ILI9341_setFrameRateIdle (uint8_t divisionRatio, uint8_t clocksPerLine)

TODO: Write brief.

• 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 startRow, uint16_t endRow)

not using backlight, so these aren't necessary

void ILI9341_setColAddress (uint16_t startCol, uint16_t endCol)

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_writePixel (uint8_t red, uint8_t green, uint8_t blue, bool is_16bit)

Write a single pixel to frame memory.

6.29.1 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.

6.30 Led.c File Reference

Source code for LED module.

```
#include "Led.h"
#include "GPIO.h"
#include "NewAssert.h"
#include <stdbool.h>
#include <stdint.h>
```

Data Structures

• struct Led_t

6.31 Led.h File Reference 87

Functions

6.30.1 Detailed Description

Source code for LED module.

Author

Bryan McElvy

6.31 Led.h File Reference

Interface for LED module.

```
#include "GPIO.h"
#include "NewAssert.h"
#include <stdbool.h>
#include <stdint.h>
```

Macros

• #define LED_POOL_SIZE 3

Functions

6.31.1 Detailed Description

Interface for LED module.

Author

Bryan McElvy

6.32 test_adc.c File Reference

Test script for analog-to-digital conversion (ADC) module.

```
#include "ADC.h"
#include "PLL.h"
#include "GPIO.h"
#include "Timer.h"
#include "FIFO.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
```

Macros

- #define LED_PINS (GPIO_Pin_t)(GPIO_PIN1 | GPIO_PIN2 | GPIO_PIN3)
- #define SAMPLING_PERIOD_MS (uint32_t) 5
- #define NUM_SAMPLES (uint32_t) 1000

Functions

- int main (void)
- void ADC0_SS3_Handler (void)

Variables

- volatile bool buffer_is_full = false
- volatile FIFO_t * fifo_ptr = 0
- volatile uint32_t fifo_buffer [NUM_SAMPLES]

6.32.1 Detailed Description

Test script for analog-to-digital conversion (ADC) module.

Author

6.33 test_daq.c File Reference

Test script for the data acquisition (DAQ) module.

```
#include "DAQ.h"
#include "Debug.h"
#include "LCD.h"
#include "ADC.h"
#include "PLL.h"
#include "FIFO.h"
#include "lookup.h"
#include "arm_math_types.h"
#include <stdbool.h>
#include <stdint.h>
```

Macros

- #define **DAQ_BUFFER_SIZE** 128
- #define LCD_TOP_LINE (Y MAX 48)
- #define LCD_NUM_Y_VALS 128
- #define LCD X AXIS OFFSET 32
- #define LCD_Y_MIN (0 + LCD_X_AXIS_OFFSET)
- #define LCD_Y_MAX (LCD_NUM_Y_VALS + LCD_X_AXIS_OFFSET)

Functions

- void LCD_plotNewSample (uint16_t x, volatile const float32_t sample)
- int main (void)
- void ADC0_SS3_Handler (void)

Variables

```
volatile FIFO_t * input_fifo_ptr = 0
```

- volatile uint32_t input_buffer [DAQ_BUFFER_SIZE] = { 0 }
- volatile bool sampleReady = false

6.33.1 Detailed Description

Test script for the data acquisition (DAQ) module.

Author

Bryan McElvy

6.34 test_debug.c File Reference

Test script for Debug module.

```
#include "Debug.h"
#include "GPIO.h"
#include "PLL.h"
#include "Timer.h"
#include <stdint.h>
```

• int main (void)

6.34.1 Detailed Description

Test script for Debug module.

Author

Bryan McElvy

6.35 test_fifo.c File Reference

Test script for FIFO buffer.

```
#include "FIFO.h"
#include "PLL.h"
#include "UART.h"
#include "GPIO.h"
#include "Timer.h"
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
```

Macros

- #define FIFO_LEN 10
- #define **LED_PINS** (GPIO_PIn_t)(GPIO_PIN1 | GPIO_PIN2 | GPIO_PIN3)

Functions

- void FIFO_reportStatus (FIFO_t *fifo_ptr)
- int main (void)

Variables

UART_t * uart

6.35.1 Detailed Description

Test script for FIFO buffer.

Author

6.36 test_lcd_image.c File Reference

Test script for writing images onto the display.

```
#include "LCD.h"
#include "GPIO.h"
#include "PLL.h"
#include "Timer.h"
#include "ILI9341.h"
#include "tm4c123gh6pm.h"
#include <stdint.h>
#include <stdbool.h>
```

Macros

- #define X_OFFSET (uint16_t) 0
- #define SIZE (uint16_t) 4
- #define LED_PINS (GPIO_Pin_t)(GPIO_PIN1 | GPIO_PIN2 | GPIO_PIN3)

Functions

• int main (void)

Variables

- const uint8_t **COLOR_ARR** [6] = { LCD_RED, LCD_YELLOW, LCD_GREEN, LCD_CYAN, LCD_BLUE, LCD_PURPLE }
- uint8_t color_idx

6.36.1 Detailed Description

Test script for writing images onto the display.

Author

Bryan McElvy

6.37 test_lcd_scroll.c File Reference

Test script for writing different colors on the LCD.

```
#include "LCD.h"
#include "PLL.h"
#include "GPIO.h"
#include "Timer.h"
#include <stdint.h>
```

Macros

- #define LED_PINS (GPIO_Pin_t)(GPIO_PIN1 | GPIO_PIN2 | GPIO_PIN3)
- #define TOP_LINE_OFFSET (uint16_t) 180
- #define TOP_LINE_THICKNESS (uint16_t) 5
- #define DX (uint16 t) 5
- #define **DY** (uint16_t) 10
- #define COL_Y_MIN (uint16_t) 0
- #define COL_Y_MAX (uint16_t) 177

Functions

• int main (void)

6.37.1 Detailed Description

Test script for writing different colors on the LCD.

Author

Bryan McElvy

6.38 test_pll.c File Reference

Test script for the PLL module.

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

Macros

- #define **RED** (uint8_t) 0x02
- #define **BLUE** (uint8_t) 0x04
- #define GREEN (uint8_t) 0x08

Functions

- void **GPIO_PortF_Init** (void)
- int main ()

6.38.1 Detailed Description

Test script for the PLL module.

Author

6.39 test_spi.c File Reference

Test script for initializing SSI0 and writing data/commands via SPI.

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

Functions

• int main ()

6.39.1 Detailed Description

Test script for initializing SSI0 and writing data/commands via SPI.

Author

Bryan McElvy

6.40 test_systick_int.c File Reference

Test script for SysTick interrupts.

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

Functions

- void **GPIO_PortF_Init** (void)
- int **main** ()
- void SysTick_Handler (void)

Variables

- const uint8_t **color_table** [6] = { 0x02, 0x06, 0x04, 0x0C, 0x08, 0x0A }
- volatile uint8_t color_idx = 0
- volatile uint8_t led_is_on = 0

6.40.1 Detailed Description

Test script for SysTick interrupts.

Author

6.41 test_timer1_int.c File Reference

Test script for Timer1A interrupts.

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

Functions

- int main (void)
- void Timer1A_Handler (void)

Variables

```
• uint8 t is led on = 0
```

6.41.1 Detailed Description

Test script for Timer1A interrupts.

Author

Bryan McElvy

6.42 test_uart_interrupt.c File Reference

(DISABLED) Test script for writing to serial port via UART0

```
#include "PLL.h"
#include "GPIO.h"
#include "Timer.h"
#include "UART.h"
#include <stdint.h>
```

Functions

• int main (void)

Variables

- const uint8_t COLOR_LIST [8]
- const char * COLOR_NAMES [8]

6.42.1 Detailed Description

(DISABLED) Test script for writing to serial port via UART0

Author

Bryan McElvy

6.42.2 Variable Documentation

COLOR_LIST

COLOR_NAMES

6.43 test_uart_la.c File Reference

Test script for using a USB logic analyzer to decode UART signals.

```
#include "PLL.h"
#include "GPIO.h"
#include "Timer.h"
#include "UART.h"
```

Functions

• int main (void)

6.43.1 Detailed Description

Test script for using a USB logic analyzer to decode UART signals.

Author

6.44 test_uart_write.c File Reference

Test script for writing to serial port via UART0.

```
#include "PLL.h"
#include "GPIO.h"
#include "Led.h"
#include "Timer.h"
#include "UART.h"
```

Functions

• int main (void)

Variables

- volatile unsigned char in_char
- uint32_t counter

6.44.1 Detailed Description

Test script for writing to serial port via UART0.

Author

Bryan McElvy

6.45 test_userctrl.c File Reference

Test file for GPIO/UserCtrl modules and GPIO interrupts.

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

Functions

• int main ()

6.45.1 Detailed Description

Test file for GPIO/UserCtrl modules and GPIO interrupts.

Author

Index

ADC., 59 ADC., ConvertToVolts Analog-to-Digital Conversion (ADC), 6 Analog-to-Digital Conversion (ADC), 6 Analog-to-Digital Conversion (ADC), 6 ADC., ConvertToVolts, 6 ASSert Common, 41 ILI9341, 20 FIFO., 45 FIFO., 45 FIFO., 43 FIFO., 45 FIFO., 44 FIFO., 44 FIFO., 44 FIFO., 45 FIFO., 44 FIFO., 45 FIFO., 46 FIFO., 47 FIFO., 46 FIFO., 46 FIFO., 46 FIFO., 47 FIFO., 46 FIFO., 46 FIFO., 46 FIFO., 47 FIFO., 47 FIFO., 48 FIFO., 48 FIFO., 48 FIFO., 48 FIFO., 49 FIFO., 46 FIFO., 46 FIFO., 47 FIFO., 47 FIFO., 47 FIFO., 48 FIFO., 48 FIFO., 48 FIFO., 49 FIFO., 46 FIFO., 47 FIFO., 46 FIFO., 47 FIFO., 47 FIFO., 48 FIFO., 46 FIFO., 47 FIFO., 47 FIFO., 47 FIFO., 48 FIFO., 48 FIFO., 48 FIFO., 49 F	ADC.c, 59	FIFO_Init, 43
Analog-to-Digital Conversion (ADC), 6 Analog-to-Digital Conversion (ADC), 6 ADC_ConvertToVolts, 6 ADC_ConvertT	ADC.h, 59	FIFO_isEmpty, 44
Analog-to-Digital Conversion (ADC), 6 Analog-to-Digital Conversion (ADC), 6 ADC_ConvertToVolts, 6 ADC_ConvertT	ADC ConvertToVolts	FIFO isFull, 44
Analog-to-Digital Conversion (ADC), 6 ADC_ConvertToVolts, 6 Application Software, 30 Assert Common, 41 CASET Ll.19341, 20 CML.1 Ll.19341, 20 COLOR_LIST test_uart_interrupt.c, 95 COLOR_NAMES Lest_uart_interrupt.c, 95 COLOR_NAMES Lest_uart_interrupt.c, 95 COLOR_NAMES TIFO_GetCurrSize COLOR_OLLST DAQ_Fitter DAQ_Fitter DAQ_Fitter, 32 Loakup_Assert, 50 Debug_Assert, 50 Debug_Assert, 51 Debug_Assert Debug_h, 51 Debug_SendFromList, 51 Debug_SendMsg Debug_h, 51 De		FIFO PeekAll, 45
ADC ConvertToVolts, 6 Application Software, 30 Assert Common, 41 FIFO_RansferAll, 46 FIFO_TransferAll, 46 FIFO_LTALL, 56 FIFO_LTALL, 57 FIFO_LT		-
Application Software, 30 Assert Assert Common, 41 FIFO_TransferAll, 46 FIFO_TransferOne, 46 FIFO_C, 55 FIFO_D, 56 FIFO_D, 56 FIFO_D, 56 FIFO_D, 43 FIFO_B, 43 FIFO_Beat, 45 FIFO_C, 43 FIFO_B, 43 FIFO_Beat, 45 FIFO_B, 43 FIFO_BEAT, 40 FIFO_COnfigNation, 62 FIFO_COnfigNation, 62 FIFO_COnfigNation, 62 FIFO_COnfigNation, 62 FIFO_COnfigNation, 62 FIFO_COnfigNation, 63 FIFO_COnfigNation, 64 FIFO_CONFIG		-
Assert Common, 41 Common, 41 Common, 41 FIFO_Transfer/One, 46 FIFO_D, 55 FIFO_D, 56 FIFO_D, 43 FIFO_Get FIFO, 43 FIFO_Get FIFO, 43 FIFO_Unit FIFO, 44 FIFO_Unit FIFO, 44 FIFO_Unit FIFO, 45 FIFO_Unit FIFO, 44 FIFO_Unit FIFO, 45 FIFO_Unit FIFO, 45 FIFO_Unit FIFO, 45 FIFO_PeekAll FIFO, 45 FIFO_Debug_Assert, 50 Debug_Assert, 50 Debug_Assert, 50 Debug_SendFromList, 51 Debug_SendFromList, 51 Debug_SendFromList, 51 Debug_SendFromList Debug_N, 50 FIFO, 46 FIFO_Transfer/One FIFO, 45 FIFO_Transfer/One FIFO, 46 FIFO_Transfer/One FIFO, 45 FIFO_Transfer/On		
Common, 41 CASET		
CASET		
CASET	Common, 41	
ILI9341, 20	CASET	
Cmd t FIFO, 43 ILI9341, 20 FIFO Get COLOR_LIST FIFO_Get test_uart_interrupt.c, 95 FIFO_getCurrSize FIFO_getCurrSize FIFO_d3 test_uart_interrupt.c, 95 FIFO_Int Common, 40 FIFO_13 Assert, 41 FIFO_16, 43 DAQ.c, 48 FIFO_ISEmpty DAQ.Filter FIFO, 44 DAQ.Filter FIFO_16, 44 DAQ.Filter FIFO, 44 DAQ.Filter FIFO, 44 DAQ.Filter, 32 FIFO, 45 Lookup_GetPtr_ADC, 32 FIFO, 45 Debug, 32 FIFO, 45 Debug, 32 FIFO, 45 Debug, 50 FIFO_Reset Debug_SendMromList, 51 FIFO_Reset Debug_SendMromList, 51 FIFO_L, 47 Debug_SendFromList FIFO_A6 Debug_SendMromList FIFO_TansferOne Debug_SendMromList FIFO_TansferOne Debug_SendMromList FIFO_TansferOne Debug_SendMromList FIFO_TansferOne FIFO_A6 FRMCTR3		
ILI9341, 20	•	
COLOR_LIST test_uart_interrupt.c, 95 COLOR_NAMES test_uart_interrupt.c, 95 Common, 40 Assert, 41 DAQ.c, 48 DAQ.h, 49 DAQ. filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug, 32 Debug_Assert, 50 Debug_SendRyg, 51 Debug_SendRyg, 51 Debug_Assert Debug_h, 50 Debug_SendRyg, 51 Debug_MriteFloat, 51 Debug_SendRyg Debug, 50 Debug_SendRyg Debug, 50 Debug_Assert Debug, 50 Debug_Assert Debug, 50 Debug_Assert Debug, 50 Debug_BendFromList Debug, 51 Debug_Nertifloat, 51 Debug_SendRyg Debug, 51 Debug_SendRyg Debug, 51 Debug_SendRyg Debug, 51 Debug_SendRyg Debug, 51 Debug_WriteFloat Debug, 51 Debug_WriteFloat Debug, 51 Debug_SendRyg Debug, 5	_	
test_uart_interrupt.c, 95 COLOR_NAMES COLOR_NAMES test_uart_interrupt.c, 95 Common, 40 Assert, 41 DAQ.c, 48 DAQ.n, 49 DAQ_Filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug_h, 50 Debug_ SendFromList, 51 Debug_ SendMsg, 51 Debug_ SendFromList, 51 Debug_ SendFromList Debug_ N, 50 Debug_ SendFromList Debug_N, 51 Debug_ SendMsg Debug_N, 51 Debug_ SendMsg Debug_N, 51 Debug_ SendFromList Debug_N, 51 Debug_ SendFromList Debug_N, 51 Debug_ SendFromList Debug_N, 51 Debug_ SendMsg Debug_N, 51 Debug_ SendMsg Debug_N, 51 Debug_ SendFromList Debug_N,		
COLOR_NAMES test_uart_interrupt.c, 95 Common, 40 Assert, 41 DAQ.c, 48 DAQ.h, 49 DAQ Filter Data Acquisition (DAQ), 32 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug_Assert, 50 Debug_SendFromList, 51 Debug_WriteFloat, 51 Debug_Assert Debug, Assert Debug, Assert Debug, Assert Debug, Assert Debug, Assert Debug, Assert Debug, SendFromList Debug, SendFromList Debug, Assert Debug, SendFromList Debug, SendFromList Debug, SendHromList Debug, Assert Debug, SendFromList FIFO, 46 FIFO, 47 FIFO, 46 FIFO, 46 FIFO, 46 FIFO, 46 FIFO, 47 FIFO, 45	-	
test_uart_interrupt.c, 95 Common, 40 Assert, 41 FIFO_ isEmpty FIFO, 44 FIFO_ isEmpty FIFO, 44 FIFO_ isFull FIFO, 44 FIFO_ isFull FIFO, 44 FIFO_ isFull FIFO, 44 FIFO_ isFull FIFO, 45 FIFO_ 45 FIFO_ eekAll FIFO_ eekOne FIFO, 45 FIFO_ eekOne FIFO_ eek		
Common, 40	-	•
Assert, 41 DAQ.c, 48 DAQ.h, 49 DAQ_Filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug, 50 Debug_Assert, 50 Debug_SendMsg, 51 Debug_SendMsg, 51 Debug_SendFromList, 51 Debug_Assert Debug, h, 50 Debug_SendFromList Debug_Assert Debug, h, 50 Debug_SendFromList Debug_SendFromList Debug_MriteFloat, 51 Debug_SendFromList Debug_N, 50 Debug_SendFromList Debug_N, 51 Debug_SendFromList Debug_N, 51 Debug_SendMsg Debug, h, 51 Debug_WriteFloat Debug_N, 51 Debug_WriteFloat Debug_N, 51 Debug_WriteFloat Debug_N, 51 Debug_SendMsg Debug, h, 51 Debug_SendMsg Debug, h, 51 Debug_SendMsg Debug, h, 51 Debug_SendMsg Debug, h, 51 Debug_WriteFloat Debug, h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 GPIO_ConfigAltMode, 62 GPIO_ConfigDirlnput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigInterrupts_LevelTrig, 64		FIFO_Init
DAQ.c, 48 DAQ.h, 49 DAQ. Filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ. Filter Data Acquisition (DAQ), 31 DAQ. Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug_Assert, 50 Debug_SendFromList, 51 Debug_SendMsg, 51 Debug_SendMsg, 51 Debug_Assert Debug, 50 Debug_Assert Debug, 50 Debug_SendFromList Debug, 50 Debug_SendFromList Debug, 50 Debug_SendFromList Debug, 50 Debug, 51 Debug, 61 Debug, 62 GPIO, ConfigAltMode, 62 GPIO, ConfigDirlnput, 62 GPIO, ConfigDirlnput, 62 GPIO, ConfigDirlouput, 62 GPIO, ConfigDirlouput, 62 GPIO, ConfigDirlouput, 62 GPIO, ConfigDirlouput, 62 GPIO, ConfigDirlerrupts, BothEdges, 63 GPIO, ConfigInterrupts, BothEdges, 63 GPIO, ConfigInterrupts, LevelTrig, 64 GPIO, ConfigInterrupts, LevelTrig, 64 GPIO, ConfigInterrupts, LevelTrig, 64 GPIO, ConfigOrltcrif, 64		FIFO, 43
DAQ.c, 48 FIFO_isFull DAQ, 49 FIFO, 44 DAQ_Filter FIFO, 45 Data Acquisition (DAQ), 31 FIFO_PeekOne DAQ_Filter, 32 FIFO_PeekOne Lookup_GetPtr_ADC, 32 FIFO_Pet Debug, 32 FIFO_Put Debug, 50 FIFO_Reset Debug_SendFromList, 51 FIFO_Reset Debug_SendMsg, 51 FIFO_L, 47 Debug_SendFromList, 51 FIFO_TransferAll Debug_Assert FIFO_TransferOne Debug_SendFromList FIFO_TransferOne Debug, 50 FIFO_TransferOne FIFO_46 FRMCTR1 Debug, SendMsg FRMCTR2 Debug_SendFromList FRMCTR2 Debug_SendFromList FRMCTR2 Debug_MriteFloat GPIO_Config. Debug_MriteFloat GPIO_TRANSFER Debug_MriteFloat GPIO_TRANSFER Debug_MriteFloat GPIO_TRANSFER Device Drivers, 4 GPIO_TRANSFER DinyOon GPIO_C, 60 GPIO_ConfigAltMode, 62 GPIO_ConfigAltMode, 62 GPIO_ConfigDirioutput, 62 GPIO_ConfigDirioutput, 62	Assert, 41	FIFO_isEmpty
DAQ.h, 49 DAQ_Filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug, 50 Debug_Assert, 50 Debug_SendMsg, 51 Debug_Assert Debug, h, 50 Debug_Assert Debug, h, 50 Debug_Assert Debug, h, 50 Debug_BendMsg, 51 Debug_BendMsg, 51 Debug_BendMsg, 51 Debug_MriteFloat, 51 Debug_SendMsg, 51 Debug_SendFromList Debug, h, 51 Debug_SendMsg Debug, h, 51 Debug_WriteFloat Debug, h, 51 Debug_WriteFloat Debug, h, 51 Debug_SendMsg Debug, h, 51 Debug_BendMsg Debug, h,	P.O. 15	FIFO, 44
DAQ_Filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug, 32 Debug, 50 Debug_Assert, 50 Debug_SendFromList, 51 Debug_SendMsg, 51 Debug_Assert Debug, 50 Debug_Assert Debug, 50 Debug_BendFromList Debug_BendFromList Debug_BendFromList Debug_SendMsg TIFO_TransferAll TIFO_46 FIFO_TransferOne FIFO_46 FIFO_CONTIGNITION FIFO_CONTIGNITION FIFO_TRANSferOne FIFO_CONTIGNITION FIFO_CONTIGNITION GPIO_CONTIGNITION FIFO_TRANSferOne FIFO_CONTIGNITION GPIO_CONTIGNITION GPIO_CONTIGNI		FIFO_isFull
DAQ_Filter Data Acquisition (DAQ), 32 Data Acquisition (DAQ), 31 DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug_ 32 Debug_ 50 Debug_ Assert, 50 Debug_ SendFromList, 51 Debug_ SendFromList, 51 Debug_ Assert Debug_ Assert Debug_ Assert Debug_ SendFromList Debug_ Assert Debug_ Asert Debug_ Assert		FIFO, 44
Data Acquisition (DAQ), 31 FIFO, 45 DAQ_Filter, 32 FIFO_PeekOne Lookup_GetPtr_ADC, 32 FIFO_Put Debug, 32 FIFO_Put Debug, 50 FIFO_Ut Debug_SendFromList, 51 FIFO_Ut Debug_SendMsg, 51 FIFO_Ut Debug_SendMsg, 51 FIFO_Ut Debug_SendMsg, 51 FIFO_Ut Debug_SendMsg, 51 FIFO_TransferAll Debug_Assert FIFO_TransferOne Debug_SendFromList FIFO_UT Debug, 50 FIFO_UT Debug, 51 FIFO_UT Debug, 51 ILI9341, 20 Debug, 51 ILI9341, 20 Debug, 51 FIFO_UT Debug, 51 ILI9341, 20 Device Drivers, 4 FIFO_UT DINVOFF GPIO_ConfigAltMode, 62 ILI9341, 20 GPIO_ConfigAltMode, 62 GPIO_ConfigDriveStrength, 62 GPIO_ConfigDriveStrength, 62 ILI9341, 20 GPIO_ConfigDriveStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_LevelTrig, 64	DAQ_Filter	
Data Acquisition (DAQ), 31 FIFO_PeekOne DAQ_Filter, 32 FIFO, 45 Lookup_GetPtr_ADC, 32 FIFO_Put Debug, 32 FIFO, 45 Debug_Assert, 50 FIFO_Reset Debug_SendFromList, 51 FIFO_145 Debug_SendMsg, 51 FIFO_146 Debug_Assert FIFO_146 Debug_Assert FIFO_146 Debug_SendFromList FIFO_146 Debug_SendMsg FRMCTR1 Debug_SendMsg FRMCTR1 Debug_SendMsg FRMCTR2 Debug_NtiteFloat REMCTR2 Debug_MriteFloat FRMCTR3 Debug_NtiteFloat FRMCTR3 Device Drivers, 4 FIFO_7 DINVOFF GPIO_7 ILI9341, 20 GPIO_ConfigAltMode, 62 DISPOFF GPIO_ConfigDirOutput, 62 ILI9341, 20 GPIO_ConfigDirOutput, 62 DISPON GPIO_ConfigDirOutput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigNVIC, 64 G	Data Acquisition (DAQ), 32	_
DAQ_Filter, 32 Lookup_GetPtr_ADC, 32 Debug, 32 Debug, 50 Debug_Assert, 50 Debug_SendFromList, 51 Debug_SendMsg, 51 Debug_Assert Debug_h, 50 Debug_h, 50 Debug_WriteFloat, 51 Debug_SendFromList Debug_h, 50 Debug_SendFromList Debug_h, 50 Debug_SendFromList Debug, 50 Debug_SendFromList Debug, 51 Debug_SendMsg Debug, 51 Debug_WriteFloat Debug, 51 Debug_WriteFloat Debug, 51 Debug_WriteFloat Debug, 51 Debug, 61 Debug, 61 Debug, 62 Debug, 51 Debug, 61 Debug	Data Acquisition (DAQ), 31	
Lookup_GetPtr_ADC, 32	DAQ_Filter, 32	
Debug, 32 FIFO, 45 Debug, h, 50 FIFO_Reset Debug_SendFromList, 51 FIFO_t, 47 Debug_SendMsg, 51 FIFO_TransferAll Debug_MriteFloat, 51 FIFO_TransferOne Debug_Assert FIFO_TransferOne Debug, h, 50 FIFO_TransferOne Debug, h, 50 FIFO_TransferOne Debug, h, 51 ILI9341, 20 Debug_SendMsg FRMCTR1 Debug_SendMsg FRMCTR2 Debug, h, 51 ILI9341, 20 Debug_WriteFloat FRMCTR3 Debug, h, 51 ILI9341, 20 Device Drivers, 4 GPIO, 7 DINVOR GPIO_ConfigAltMode, 62 ILI9341, 20 GPIO_ConfigAldMode, 62 DISPOFF GPIO_ConfigDirlout, 62 ILI9341, 20 GPIO_ConfigDirlout, 62 DISPON GPIO_ConfigDirlout, 62 ILI9341, 20 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LeveITrig, 64 FIFO, 41 GPIO_ConfigNVIC, 64 FIFO_Get, 43 GPIO_ConfigPortCitf, 64	Lookup_GetPtr_ADC, 32	, and the second
Debug_h, 50 FIFO_Reset Debug_Assert, 50 FIFO_L, 45 Debug_SendFromList, 51 FIFO_t, 47 Debug_SendMsg, 51 FIFO_TransferAll Debug_Assert FIFO_TransferOne Debug_Assert FIFO_TransferOne Debug_SendFromList FRMCTR1 Debug_SendMsg FRMCTR2 Debug_h, 51 ILI9341, 20 Debug_WriteFloat FRMCTR3 Debug_h, 51 FRMCTR3 Device Drivers, 4 FRMCTR3 DINVOFF GPIO, 7 ILI9341, 20 GPIO_ConfigAltMode, 62 DISPOFF GPIO_ConfigAnalog, 62 ILI9341, 20 GPIO_ConfigDirloutput, 62 DISPON GPIO_ConfigDirloutput, 62 ILI9341, 20 GPIO_ConfigDirloutput, 62 DISPON GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigPortCitl, 64 FIFO, 43 GPIO_ConfigPortCitl, 64	Debug, 32	_
Debug_Assert, 50 Debug_SendFromList, 51 Debug_SendMsg, 51 Debug_WriteFloat, 51 Debug_SendMsg Debug_SendFromList Debug_SendFromList Debug_SendFromList Debug_SendFromList Debug_SendMsg Debug_SendMsg Debug_SendMsg Debug_SendMsg Debug_MriteFloat Debug_WriteFloat Debug_MriteFloat Debug_MriteFloat Debug_N, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 DISPO, 41 FIFO_Flush, 43 FIFO_Get, 43 FIFO_Get, 43 FIFO_Get, 43 FIFO_ConfigInterrupts_BothEdges, 63 FIFO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigOrCtrl. 64		,
Debug_SendFromList, 51 Debug_SendMsg, 51 Debug_WriteFloat, 51 Debug_Assert Debug_Lh, 50 Debug_SendMsg Debug_Lh, 51 Debug_SendMsg Debug_SendMsg Debug_SendMsg Debug_SendMsg Debug_SendMsg Debug_WriteFloat Debug_WriteFloat Debug_WriteFloat Debug_N, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 DISPOR ILIPSHIP ILIP	_	
Debug_SendMsg, 51 Debug_WriteFloat, 51 Debug_Assert Debug_Assert Debug_h, 50 Debug_SendFromList Debug_h, 51 Debug_SendMsg Debug_h, 51 Debug_BendMsg Debug_h, 51 Debug_WriteFloat Debug_writeFloat Debug_h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 DISPOR ILI9341, 20 DISPOR ILI9341, 20 DISPON ILI93	-	•
Debug_WriteFloat, 51 Debug_Assert Debug_h, 50 Debug_SendFromList Debug_h, 51 Debug_SendMsg Debug_h, 51 Debug_BendMsg Debug_h, 51 Debug_WriteFloat Debug_h, 51 Debug_WriteFloat Debug_h, 51 Debug_writeFloat Debug_h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPOFF ILI9341, 20 DISPOR ILI9341, 20 DISPON ILI9341, 20 GPIO_ConfigDiroutput, 62 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigNVIC, 64	-	
Debug_Assert Debug_h, 50 Debug_SendFromList Debug_SendMsg Debug_SendMsg Debug_M, 51 Debug_SendMsg Debug_MriteFloat Debug_WriteFloat Debug_MriteFloat Debug_h, 51 Debug_WriteFloat Debug_writeFloat Debug_MriteFloat Debug_MriteF		
Debug_SendFromList Debug_SendMsg Debug_SendMsg Debug_SendMsg Debug_MriteFloat Debug_WriteFloat Debug_MriteFloat Debug_MriteFl	-	,
Debug_SendFromList Debug_N, 51 Debug_SendMsg Debug_N, 51 Debug_WriteFloat Debug_WriteFloat Debug_N, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 DISPON ILI9341, 20 FRMCTR3 ILI9341, 20 GPIO_ConfigDirlnput, 62 GPIO_ConfigDirlnput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	-	
Debug.h, 51 Debug_SendMsg Debug.h, 51 Debug_WriteFloat Debug.h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 GPIO_ConfigDirOutput, 62 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	_	
Debug_SendMsg Debug.h, 51 Debug_WriteFloat Debug.h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 FRMCTR3 ILI9341, 20 GPIO, 7 GPIO, 60 GPIO_ConfigAltMode, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigNVIC, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	-	FRMCTR1
Debug.h, 51 Debug_WriteFloat Debug.h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 DISPON ILI9341, 20 DISPON ILI9341, 20 DISPON ILI9341, 20 FIFO_Get, 43 FIFO_Get, 43 FIFO_Get, 43 FIFO_Get, 43 ILI9341, 20 FRMCTR3 ILI9341, 20 GPIO, 7 GPIO, 60 GPIO_ConfigAltMode, 62 GPIO_ConfigAnalog, 62 GPIO_ConfigDirlout, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigDirloutput, 62 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	•	ILI9341, <mark>20</mark>
Debug_WriteFloat Debug.h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 GPIO_ConfigDriveStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	-	FRMCTR2
Debug.h, 51 Device Drivers, 4 DINVOFF ILI9341, 20 GPIO, 7 ILI9341, 20 GPIO_ConfigAltMode, 62 GPIO_ConfigAnalog, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDiroutput, 62 GPIO_ConfigDiroutput, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 FIFO_ConfigPortCtrl, 64	•	ILI9341, 20
Device Drivers, 4 DINVOFF ILI9341, 20 DINVON ILI9341, 20 DISPOFF ILI9341, 20 DISPON ILI9341, 20 DISPON ILI9341, 20 DISPON ILI9341, 20 FIFO, 41 FIFO_Flush, 43 FIFO_Get, 43 FIFO_Get, 43 GPIO, Tonfig AltMode, 62 GPIO_Config Analog, 62 GPIO_Config Dirloput, 62 GPIO_Config Dirloput, 62 GPIO_Config Dirve Strength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	-	FRMCTR3
DINVOFF ILI9341, 20 GPIO.c, 60 DINVON ILI9341, 20 GPIO_ConfigAltMode, 62 GPIO_ConfigAnalog, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDiroutput, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64		ILI9341, 20
ILI9341, 20 DINVON ILI9341, 20 GPIO_ConfigAltMode, 62 GPIO_ConfigAnalog, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDiroutput, 63 GPIO_ConfigDireStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64	•	
DINVON ILI9341, 20 GPIO_ConfigAltMode, 62 GPIO_ConfigAnalog, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDirveStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64	DINVOFF	GPIO, 7
ILI9341, 20 DISPOFF ILI9341, 20 GPIO_ConfigAnalog, 62 GPIO_ConfigDirInput, 62 GPIO_ConfigDirOutput, 62 GPIO_ConfigDiroutput, 62 GPIO_ConfigDiroutput, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64	ILI9341, <mark>20</mark>	GPIO.c, 60
DISPOFF ILI9341, 20 DISPON ILI9341, 20 GPIO_ConfigDirloutput, 62 GPIO_ConfigDriveStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64	DINVON	GPIO_ConfigAltMode, 62
DISPOFF ILI9341, 20 DISPON ILI9341, 20 GPIO_ConfigDirloutput, 62 GPIO_ConfigDriveStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64	ILI9341, 20	GPIO ConfigAnalog, 62
ILI9341, 20 DISPON GPIO_ConfigDirOutput, 62 GPIO_ConfigDriveStrength, 63 ILI9341, 20 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigPortCtrl, 64 GPIO_ConfigPortCtrl, 64	DISPOFF	
DISPON ILI9341, 20 GPIO_ConfigDriveStrength, 63 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	ILI9341, 20	_ • •
ILI9341, 20 GPIO_ConfigInterrupts_BothEdges, 63 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	DISPON	_ • •
FIFO, 41 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigInterrupts_Edge, 63 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64		
FIFO, 41 FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigInterrupts_LevelTrig, 64 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	•	
FIFO_Flush, 43 FIFO_Get, 43 GPIO_ConfigNVIC, 64 GPIO_ConfigPortCtrl, 64	FIFO, 41	_ · · - ·
FIFO_Get, 43 GPIO_ConfigPortCtrl, 64		
Trio Connucioni 04	- ' '	
	- · · ·	ar io_comigronom, 04

GPIO ConfigPullDown, 65	GPIO_ConfigPortCtrl
GPIO ConfigPullUp, 65	GPIO.c, 64
GPIO_DisableDigital, 65	GPIO.h, 72
	GPIO_ConfigPullDown
GPIO_EnableDigital, 65	
GPIO_InitPort, 66	GPIO.c, 65
GPIO_isPortInit, 66	GPIO.h, 72
GPIO_ReadPins, 66	GPIO_ConfigPullUp
GPIO_Toggle, 66	GPIO.c, 65
GPIO_WriteHigh, 67	GPIO.h, 72
GPIO WriteLow, 67	GPIO_DisableDigital
GPIO.h, 67	GPIO.c, 65
GPIO ConfigAltMode, 69	GPIO.h, 72
GPIO_ConfigAnalog, 69	GPIO_EnableDigital
GPIO_ConfigDirInput, 69	GPIO.c, 65
GPIO_ConfigDirOutput, 70	GPIO.h, 73
GPIO_ConfigDriveStrength, 70	GPIO_InitPort
GPIO_ConfigInterrupts_BothEdges, 70	GPIO.c, 66
GPIO_ConfigInterrupts_Edge, 71	GPIO.h, 73
GPIO_ConfigInterrupts_LevelTrig, 71	GPIO_isPortInit
GPIO ConfigNVIC, 71	GPIO.c, 66
GPIO ConfigPortCtrl, 72	GPIO.h, 73
GPIO_ConfigPullDown, 72	GPIO_Port_t, 47
GPIO ConfigPullUp, 72	GPIO ReadPins
	-
GPIO_DisableDigital, 72	GPIO.c, 66
GPIO_EnableDigital, 73	GPIO.h, 74
GPIO_InitPort, 73	GPIO_Toggle
GPIO_isPortInit, 73	GPIO.c, 66
GPIO_ReadPins, 74	GPIO.h, 74
GPIO Toggle, 74	GPIO_WriteHigh
GPIO WriteHigh, 74	GPIO.c, 67
GPIO_WriteLow, 74	GPIO.h, 74
GPIO_ConfigAltMode	GPIO_WriteLow
GPIO.c, 62	GPIO.c, 67
•	
GPIO.h, 69	GPIO.h, 74
GPIO_ConfigAnalog	IDMOFF
GPIO.c, 62	
GPIO.h, 69	ILI9341, 20
GPIO_ConfigDirInput	IDMON
GPIO.c, 62	ILI9341, 20
GPIO.h, 69	IFCTL
GPIO_ConfigDirOutput	ILI9341, 20
GPIO.c, 62	ILI9341, 18
GPIO.h, 70	CASET, 20
GPIO ConfigDriveStrength	Cmd t, 20
_ •	DINVOFF, 20
GPIO.c, 63	DINVON, 20
GPIO.h, 70	DISPOFF, 20
GPIO_ConfigInterrupts_BothEdges	
GPIO.c, 63	DISPON, 20
GPIO.h, 70	FRMCTR1, 20
GPIO_ConfigInterrupts_Edge	FRMCTR2, 20
GPIO.c, 63	FRMCTR3, 20
GPIO.h, 71	IDMOFF, 20
GPIO_ConfigInterrupts_LevelTrig	IDMON, 20
GPIO.c, 64	IFCTL, 20
	ILI9341 resetHard, 21
GPIO.h, 71	ILI9341 resetSoft, 21
GPIO_ConfigNVIC	ILI9341 setColAddress, 21
GPIO.c, 64	-
GPIO.h, 71	ILI9341_setColorDepth, 21
	ILI9341_setDispInversion, 22

ILI9341_setDispMode, 22	ILI9341_setSleepMode
ILI9341_setDispOutput, 22	ILI9341, 25
ILI9341_setFrameRateIdle, 23	ILI9341_writeMemCmd
ILI9341_setFrameRateNorm, 23	ILI9341, <mark>26</mark>
ILI9341_setInterface, 23	ILI9341_writePixel
ILI9341_setMemAccessCtrl, 24	ILI9341, <mark>26</mark>
ILI9341_setPartialArea, 24	
ILI9341_setRowAddress, 25	LCD, 33
ILI9341_setScrollArea, 25	LCD_Draw, 34
ILI9341_setScrollStart, 25	LCD_drawHoriLine, 34
ILI9341_setSleepMode, 25	LCD_drawRectangle, 35
ILI9341_writeMemCmd, 26	LCD_drawVertLine, 35
ILI9341_writePixel, 26	LCD_graphSample, 35
MADCTL, 20	LCD_setArea, 36
NORON, 20	LCD_setColor, 36
PASET, 20	LCD_setColor_3bit, 37
PIXSET, 20	LCD_setColorDepth, 37
PLTAR, 20	LCD_setColorInversionMode, 38
PRCTR, 20	LCD_setOutputMode, 38
PTLON, 20	LCD_setX, 38
RAMWR, 20	LCD_setY, 39
SPLIN, 20	LCD_toggleColorDepth, 39
SPLOUT, 20	LCD_toggleColorInversion, 39
SWRESET, 20	LCD_toggleOutput, 39
VSCRDEF, 20	LCD.c, 51
VSCRSADD, 20	LCD.h, 53 LCD Draw
ILI9341.c, 83	LCD_Blaw LCD, 34
ILI9341.h, 85	LCD drawHoriLine
ILI9341_resetHard	LCD, 34
ILI9341, 21	LCD_drawRectangle
ILI9341_resetSoft	LCD, 35
ILI9341, 21	LCD_drawVertLine
ILI9341_setColAddress	LCD, 35
ILI9341, 21	LCD_graphSample
ILI9341_setColorDepth	LCD, 35
ILI9341, 21	LCD setArea
ILI9341_setDispInversion	LCD, 36
ILI9341, 22	LCD_setColor
ILI9341_setDispMode	LCD, 36
ILI9341, 22 ILI9341_setDispOutput	LCD setColor 3bit
ILI9341, 22	LCD, 37
ILI9341_setFrameRateIdle	LCD_setColorDepth
ILI9341, 23	LCD, 37
ILI9341_setFrameRateNorm	LCD_setColorInversionMode
ILI9341, 23	_ LCD, 38
ILI9341_setInterface	LCD setOutputMode
ILI9341, 23	 LCD, 38
ILI9341_setMemAccessCtrl	LCD_setX
ILI9341, 24	 LCD, 38
ILI9341_setPartialArea	LCD_setY
ILI9341, 24	LCD, 39
ILI9341_setRowAddress	LCD_toggleColorDepth
ILI9341, 25	LCD, 39
ILI9341_setScrollArea	LCD_toggleColorInversion
ILI9341, 25	LCD, 39
ILI9341_setScrollStart	LCD_toggleOutput
ILI9341, 25	LCD, 39
	LED, 27

Led_GetPin, 28	Serial Peripheral Interface (SPI), 8
Led_GetPort, 28	SPI_Init, 9
Led_Init, 29	SPI_Read, 9
Led_isOn, 29	SPI_SET_DC, 9
Led_Toggle, 29	SPI_WriteCmd, 9
Led_TurnOff, 29	SPI WriteData, 10
Led_TurnOn, 30	SPI.c, 76
	SPI.h, 77
Led.c, 86	
Led.h, 87	SPI_Init
Led_GetPin	Serial Peripheral Interface (SPI), 9
LED, 28	SPI_Read
Led_GetPort	Serial Peripheral Interface (SPI), 9
LED, 28	SPI_SET_DC
Led_Init	Serial Peripheral Interface (SPI), 9
LED, 29	SPI_WriteCmd
Led_isOn	Serial Peripheral Interface (SPI), 9
LED, 29	SPI_WriteData
Led_t, 48	Serial Peripheral Interface (SPI), 10
Led_Toggle	SPLIN
LED, 29	ILI9341, 20
Led_TurnOff	SPLOUT
LED, 29	ILI9341, 20
Led_TurnOn	SWRESET
LED, 30	ILI9341, 20
lookup.c, 57	System Tick (SysTick), 10
lookup.h, 57	SysTick_Interrupt_Init, 11
Lookup_GetPtr_ADC	SysTick.c, 77
Data Acquisition (DAQ), 32	SysTick.h, 78
Data rioquiotion (Dria), 02	SysTick_Interrupt_Init
	Cystick interrupt int
MADCTL	• - •
MADCTL II 19341, 20	System Tick (SysTick), 11
ILI9341, 20	System Tick (SysTick), 11
ILI9341, 20 main.c, 82	System Tick (SysTick), 11 test_adc.c, 88
ILI9341, 20	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89
ILI9341, 20 main.c, 82 Middleware, 17	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20 PASET	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20 PASET ILI9341, 20	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20 PASET ILI9341, 20 Phase-Locked Loop (PLL), 7	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20 PASET ILI9341, 20 Phase-Locked Loop (PLL), 7 PIXSET	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20 PASET ILI9341, 20 Phase-Locked Loop (PLL), 7 PIXSET ILI9341, 20	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_la.c, 95
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_userctrl.c, 96
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON ILI9341, 20 PASET ILI9341, 20 Phase-Locked Loop (PLL), 7 PIXSET ILI9341, 20 PLL.c, 75 PLL.h, 75 PLTAR	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_systick_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_la.c, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer0A_Start, 12
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_systick_int.c, 94 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_la.c, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer0A_Start, 12
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_uart_interrupt.c, 94
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 test_uart_interrupt.c, 94
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_usrt_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_uart_write.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer0A_Wait1ms, 12 Timer1A_Init, 13 Timer2A_isCounting, 13
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_uart_write.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer1A_Init, 13 Timer2A_isCounting, 13 Timer2A_Start, 13
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_uart_interrupt.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer0A_Wait1ms, 12 Timer2A_isCounting, 13 Timer2A_Start, 13 Timer2A_Wait1ms, 13
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	System Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 COLOR_LIST, 95 COLOR_NAMES, 95 test_uart_la.c, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer0A_Wait1ms, 12 Timer1A_Init, 13 Timer2A_isCounting, 13 Timer2A_Start, 13 Timer2A_Wait1ms, 13 Timer3A_Init, 14 Timer.c, 78
ILI9341, 20 main.c, 82 Middleware, 17 NewAssert, 47 NewAssert.c, 58 NewAssert.h, 58 NORON	system Tick (SysTick), 11 test_adc.c, 88 test_daq.c, 89 test_debug.c, 89 test_fifo.c, 90 test_lcd_image.c, 91 test_lcd_scroll.c, 91 test_pll.c, 92 test_spi.c, 93 test_systick_int.c, 93 test_timer1_int.c, 94 color_LIST, 95 color_NAMES, 95 test_uart_la.c, 95 test_uart_write.c, 96 test_userctrl.c, 96 Timer, 11 Timer0A_isCounting, 12 Timer1A_Init, 13 Timer2A_isCounting, 13 Timer2A_Start, 13 Timer2A_Wait1ms, 13 Timer2A_Wait1ms, 13 Timer3A_Init, 14

```
Timer, 12
Timer0A_Start
    Timer, 12
Timer0A_Wait1ms
    Timer, 12
Timer1A Init
    Timer, 13
Timer2A_isCounting
    Timer, 13
Timer2A_Start
    Timer, 13
Timer2A_Wait1ms
    Timer, 13
Timer3A Init
    Timer, 14
UART.c, 80
UART.h, 82
UART_Init
                               Receiver/Transmitter
    Universal
              Asynchronous
        (UART), 15
UART_ReadChar
                               Receiver/Transmitter
    Universal Asynchronous
         (UART), 16
UART_t, 48
UART_WriteChar
    Universal Asynchronous
                               Receiver/Transmitter
         (UART), 16
UART_WriteFloat
    Universal Asynchronous
                               Receiver/Transmitter
         (UART), 16
UART WriteInt
                               Receiver/Transmitter
    Universal
               Asynchronous
         (UART), 17
UART WriteStr
    Universal
               Asynchronous
                               Receiver/Transmitter
         (UART), 17
Universal Asynchronous Receiver/Transmitter (UART),
         14
    UART_Init, 15
    UART_ReadChar, 16
    UART_WriteChar, 16
    UART_WriteFloat, 16
    UART_WriteInt, 17
    UART_WriteStr, 17
VSCRDEF
    ILI9341, 20
VSCRSADD
    ILI9341, 20
```