EE445M/EE380L Lab 2 Documentation

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

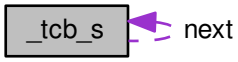
Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 _tcb_s Struct Reference

Collaboration diagram for _tcb_s:



Data Fields

- long * **sp**
- struct _tcb_s * **next**
- uint32_t **wake_time**
- unsigned long **id**
- uint8_t **priority**
- uint32_t **period**
- unsigned long magic

  magic field must contain TCB_MAGIC for TCB to be valid
- void(* **task**)(void)
- char * **task_name**

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

- inc/OS.h

3.2 event_t Struct Reference

Data Fields

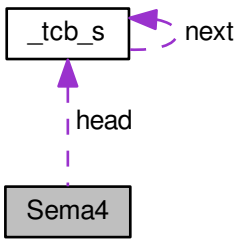
- event_type_e **type**
- int **magic**
- char * **name**
- unsigned long long **timestamp**

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

- inc/profiler.h

3.3 Sema4 Struct Reference

Collaboration diagram for Sema4:



Data Fields

- long **Value**
- struct _tcb_s * **head**

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

- inc/OS.h

Chapter 4

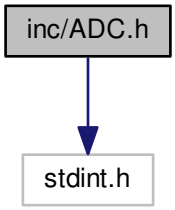
File Documentation

4.1 inc/ADC.h File Reference

ADC driver for the TM4C123G. Provides interfaces for collecting single samples or a series at a given sampling frequency. Does not allow for sampling of more than one channel at any given time. Timer 2 is reserved for this driver.

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

Include dependency graph for ADC.h:



Functions

- int ADC_Init (uint32_t channelNum)

  Configure an ADC channel for continuous sampling. Retrieve measurements from this channel with ADC_In().

- uint16_t ADC_In (void)

  Returns the most recent sample collected by the channel configured in ADC_Init(...)

- int ADC_Collect (uint32_t channelNum, uint32_t fs, void(∗handler)(unsigned long))

  Kick off collection of a sequence of samples to be passed to a user-provided handler. The ADC and Timer will be configured to collect samples at frequency fs.

4.1.1 Detailed Description

ADC driver for the TM4C123G. Provides interfaces for collecting single samples or a series at a given sampling frequency. Does not allow for sampling of more than one channel at any given time. Timer 2 is reserved for this driver.

Author

Riley Wood and Jeageun Jung

4.1.2 Function Documentation

4.1.2.1 int ADC_Collect (uint32_t *channelNum,* uint32_t *fs,* void(*)(unsigned long) *handler*)

Kick off collection of a sequence of samples to be passed to a user-provided handler. The ADC and Timer will be configured to collect samples at frequency fs.

Parameters

channelNum	ADC channel to sample
fs	Sampling frequency
handler	Function which will be passed each sample as it is collected.

Returns

int 0 on success, -1 on failure.

4.1.2.2 uint16_t ADC_In (void)

Returns the most recent sample collected by the channel configured in ADC_Init(...)

If the channel has not finished collecting its first sample, this function returns 0xFFFF.

If you call this rapidly, faster than the ADC samples, this function may repeat values (since it always returns the most recent).

Returns

uint16_t The conversion result

4.1.2.3 int ADC_Init (uint32_t *channelNum*)

Configure an ADC channel for continuous sampling. Retrieve measurements from this channel with ADC_In().

Parameters

channelNum	The channel to set up

Returns

int 0 on success, -1 on failure.

4.2 inc/interpreter.h File Reference

Functions

- void interpreter_task (void)
  OS Task that sends characters to the interpreter.
- void interpreter_cmd (char *cmd_str)
  Pass user input to the interpreter and act on their command.

4.2.1 Detailed Description

List of commands

- adc
  - Prints 2 consecutive ADC samples of channel 0 to the LCD and UART0
- lcd
  - Prints strings on each line of each logical display on the LCD.

4.2.2 Function Documentation

4.2.2.1 void interpreter_cmd (char * *cmd_str*)

Pass user input to the interpreter and act on their command.

Parameters

cmd_str	String containing the entire user command.

4.3 inc/misc_macros.h File Reference

Some helper macros.

Macros

- #define lengthof(array) (sizeof(array)/sizeof((array)[0]))
  Get the number of elements in an array.
- #define zeroes(array) memset(array, 0, sizeof(array))
  Zeroes out an array.

4.3.1 Detailed Description

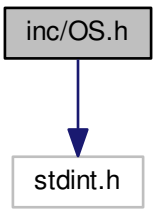
Some helper macros.

4.4 inc/OS.h File Reference

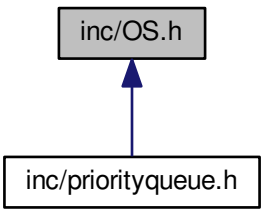
Real Time Operating System for Labs 2 and 3 EE445M/EE380L.12.

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

Include dependency graph for OS.h:



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



Data Structures

- struct _tcb_s
- struct Sema4

Macros

- #define **TIME_1MS** 80000
- #define **TIME_2MS** (2 ∗ TIME_1MS)
- #define **TIME_500US** (TIME_1MS / 2)
- #define **TIME_250US** (TIME_1MS / 4)
- #define **TASK_STACK_SIZE** 128
- #define **TCB_MAGIC** (0x900d900d)
- #define OS_AddThread(task, stackSize, priority) OS_AddThread_priv(task, stackSize, priority, #task)
- #define OS_AddPeriodicThread(task, period, priority) OS_AddPeriodicThread_priv(task, period, priority, #task)

Typedefs

- typedef struct _tcb_s **tcb_t**
- typedef struct Sema4 **Sema4Type**

Functions

- void OS_Init (void)
- void OS_InitSemaphore (Sema4Type ∗semaPt, long value)
- void OS_Wait (Sema4Type ∗semaPt)
- void OS_Signal (Sema4Type ∗semaPt)
- void OS_bWait (Sema4Type ∗semaPt)
- void OS_bSignal (Sema4Type ∗semaPt)
- int **OS_AddThread_priv** (void(∗task)(void), unsigned long stackSize, unsigned long priority, char ∗task_↩ name)
- unsigned long OS_Id (void)
- int **OS_AddPeriodicThread_priv** (void(∗task)(void), unsigned long period, unsigned long priority, char ∗task_name)
- int OS_AddSW1Task (void(∗task)(void), unsigned long priority)
- int OS_AddSW2Task (void(∗task)(void), unsigned long priority)
- void OS_Sleep (unsigned long sleepTime)
- void OS_Kill (void)
- void OS_Suspend (void)
- void OS_Fifo_Init (unsigned long size)
- int OS_Fifo_Put (unsigned long data)
- unsigned long OS_Fifo_Get (void)
- long OS_Fifo_Size (void)
- void OS_MailBox_Init (void)
- void OS_MailBox_Send (unsigned long data)
- unsigned long OS_MailBox_Recv (void)
- unsigned long long OS_Time (void)
- unsigned long long OS_TimeDifference (unsigned long long start, unsigned long long stop)
- void OS_ClearMsTime (void)
- unsigned long OS_MsTime (void)
- void OS_Launch (unsigned long theTimeSlice)
- long **StartCritical** (void)
- void **EndCritical** (long sr)
- void **DisableInterrupts** (void)
- void **EnableInterrupts** (void)

4.4.1 Detailed Description

Real Time Operating System for Labs 2 and 3 EE445M/EE380L.12.

RTOS kernel capable of round-robin scheduling, up to 2 low-jitter periodic tasks.

Reserves WTIMER1A and B for periodic task scheduling. Reserves SysTick timer for round-robin scheduling. Reserves WTIMER0 as a 64-bit time source.

Interface by Jonathan W. Valvano 2/20/17, valvano@mail.utexas.edu Implementation by Riley Wood and Jeageun Jung

Author

Riley Wood and Jeageun Jung

4.4.2 Macro Definition Documentation

4.4.2.1 #define OS_AddPeriodicThread(*task, period, priority*) OS_AddPeriodicThread_priv(task, period, priority, #task)

Add a background periodic task. Typically this function receives the highest priority You are free to select the time resolution for this function It is assumed that the user task will run to completion and return This task can not spin, block, loop, sleep, or kill This task can call OS_Signal OS_bSignal OS_AddThread This task does not have a Thread ID In lab 2, this command will be called 0 or 1 times In lab 2, the priority field can be ignored In lab 3, this command will be called 0 1 or 2 times In lab 3, there will be up to four background threads, and this priority field determines the relative priority of these four threads

Parameters

pointer	to a void/void background function
period	given in system time units (12.5ns)
priority	0 is the highest, 5 is the lowest

Returns

1 if successful, 0 if this thread can not be added

4.4.2.2 #define OS_AddThread(*task, stackSize, priority*) OS_AddThread_priv(task, stackSize, priority, #task)

add a foregound thread to the scheduler stack size must be divisable by 8 (aligned to double word boundary) In Lab 2, you can ignore both the stackSize and priority fields In Lab 3, you can ignore the stackSize fields

Parameters

task	Task function
stackSize	Size of the stack in bytes. Should be divisible by 8
priority	Priority of the task. 0 is highest, 5 is lowest.

Returns

1 if successful, 0 if this thread can not be added

4.4.3 Function Documentation

4.4.3.1 int OS_AddSW1Task (void(*)(void) *task,* unsigned long *priority*)

add a background task to run whenever the SW1 (PF4) button is pushed

Parameters

pointer	to a void/void background function
priority	0 is the highest, 5 is the lowest

Returns

1 if successful, 0 if this thread can not be added It is assumed that the user task will run to completion and return This task can not spin, block, loop, sleep, or kill This task can call OS_Signal OS_bSignal OS_Add↩ Thread This task does not have a Thread ID In labs 2 and 3, this command will be called 0 or 1 times In lab 2, the priority field can be ignored In lab 3, there will be up to four background threads, and this priority field determines the relative priority of these four threads

4.4.3.2 int OS_AddSW2Task (void(*)(void) *task,* unsigned long *priority*)

add a background task to run whenever the SW2 (PF0) button is pushed

Parameters

pointer	to a void/void background function
priority	0 is highest, 5 is lowest

Returns

1 if successful, 0 if this thread can not be added It is assumed user task will run to completion and return This task can not spin block loop sleep or kill This task can call issue OS_Signal, it can call OS_AddThread This task does not have a Thread ID In lab 2, this function can be ignored In lab 3, this command will be called will be called 0 or 1 times In lab 3, there will be up to four background threads, and this priority field determines the relative priority of these four threads

4.4.3.3 void OS_bSignal (Sema4Type * *semaPt*)

Lab2 spinlock, set to 1 Lab3 wakeup blocked thread if appropriate

Parameters

sema↩ Pt	pointer to a binary semaphore

4.4.3.4 void OS_bWait (Sema4Type * *semaPt*)

Lab2 spinlock, set to 0 Lab3 block if less than zero

Parameters

sema↩ Pt	pointer to a binary semaphore

4.4.3.5 void OS_ClearMsTime (void)

Sets the system time to zero (from Lab 1). You are free to change how this works.

Returns

none

4.4.3.6 unsigned long OS_Fifo_Get (void)

Remove one data sample from the Fifo. Called in foreground, will spin/block if empty

Returns

data

4.4.3.7 void OS_Fifo_Init (unsigned long *size*)

Initialize the Fifo to be empty. In Lab 2, you can ignore the size field. In Lab 3, you should implement the user-defined fifo size. In Lab 3, you can put whatever restrictions you want on size e.g., 4 to 64 elements e.g., must be a power of 2,4,8,16,32,64,128

Parameters

size	Size of the fifo

Returns

none

4.4.3.8 int OS_Fifo_Put (unsigned long *data*)

Enter one data sample into the Fifo. Called from the background, so no waiting. Since this is called by interrupt handlers this function can not disable or enable interrupts.

Parameters

data	Data to put in the FIFO

Returns

true if data is properly saved, false if data not saved, because it was full

4.4.3.9 long OS_Fifo_Size (void)

Check the status of the Fifo.

Returns

returns the number of elements in the Fifo. Greater than zero if a call to OS_Fifo_Get will return right away, zero or less than zero if the Fifo is empty, zero or less than zero if a call to OS_Fifo_Get will spin or block

4.4.3.10 unsigned long OS_Id (void)

returns the thread ID for the currently running thread

Returns

Thread ID, number greater than zero

4.4.3.11 void OS_Init (void)

initialize operating system, disable interrupts until OS_Launch initialize OS controlled I/O: serial, ADC, systick, LaunchPad I/O and timers

4.4.3.12 void OS_InitSemaphore (Sema4Type * *semaPt,* long *value*)

initialize semaphore

Parameters

sema↩ Pt	pointer to a semaphore

4.4.3.13 void OS_Kill (void)

kill the currently running thread, release its TCB and stack

4.4.3.14 void OS_Launch (unsigned long *theTimeSlice*)

Start the scheduler, enable interrupts. In Lab 2, you can ignore the theTimeSlice field. In Lab 3, you should implement the user-defined TimeSlice field. It is ok to limit the range of theTimeSlice to match the 24-bit SysTick.

Parameters

theTimeSlice	number of 12.5ns clock cycles for each time slice

Returns

none (does not return)

4.4.3.15 void OS_MailBox_Init (void)

Initialize communication channel

Returns

none

4.4.3.16 unsigned long OS_MailBox_Recv (void)

Remove mail from the MailBox. This function will be called from a foreground thread. It will spin/block if the MailBox is empty.

Returns

data received

4.4.3.17 void OS_MailBox_Send (unsigned long *data*)

Enter mail into the MailBox. This function will be called from a foreground thread. It will spin/block if the MailBox contains data not yet received

Parameters

data	to be sent

Returns

none

4.4.3.18 unsigned long OS_MsTime (void)

Reads the current time in msec (from Lab 1). You are free to select the time resolution for this function. It is ok to make the resolution to match the first call to OS_AddPeriodicThread.

Returns

time in ms units

4.4.3.19 void OS_Signal (Sema4Type * *semaPt*)

increment semaphore Lab2 spinlock Lab3 wakeup blocked thread if appropriate

Parameters

sema↩ Pt	pointer to a counting semaphore

4.4.3.20 void OS_Sleep (unsigned long *sleepTime*)

Place this thread into a dormant state. You are free to select the time resolution for this function. OS_Sleep(0) implements cooperative multitasking.

Parameters

sleepTime	number of msec to sleep

4.4.3.21 void OS_Suspend (void)

suspend execution of currently running thread. scheduler will choose another thread to execute. Can be used to implement cooperative multitasking. Same function as OS_Sleep(0).

4.4.3.22 unsigned long long OS_Time (void)

Return the system time in system time units (12.5ns)

Returns

time in 12.5ns units, 0 to 4294967295

4.4.3.23 unsigned long long OS_TimeDifference (unsigned long long *start,* unsigned long long *stop*)

Calculates difference between two times. The time resolution should be less than or equal to 1us, and the precision at least 12 bits. It is ok to change the resolution and precision of this function as long as this function and OS_Time have the same resolution and precision.

Parameters

start	Start time measured with OS_Time
stop	Stop time measured with OS_Time

Returns

time difference in 12.5ns units

4.4.3.24 void OS_Wait (Sema4Type * *semaPt*)

decrement semaphore Lab2 spinlock Lab3 block if less than zero

Parameters

sema↩ Pt	pointer to a counting semaphore

4.5 inc/PLL.h File Reference

Runs on LM4F120/TM4C123 A software function to change the bus frequency using the PLL.

Macros

- #define **Bus80MHz** 4
- #define **Bus80_000MHz** 4
- #define **Bus66_667MHz** 5
- #define **Bus50_000MHz** 7
- #define **Bus50MHz** 7
- #define **Bus44_444MHz** 8
- #define **Bus40_000MHz** 9
- #define **Bus40MHz** 9
- #define **Bus36_364MHz** 10
- #define **Bus33_333MHz** 11
- #define **Bus30_769MHz** 12
- #define **Bus28_571MHz** 13
- #define **Bus26_667MHz** 14
- #define **Bus25_000MHz** 15
- #define **Bus25MHz** 15
- #define **Bus23_529MHz** 16
- #define **Bus22_222MHz** 17
- #define **Bus21_053MHz** 18
- #define **Bus20_000MHz** 19
- #define **Bus20MHz** 19
- #define **Bus19_048MHz** 20
- #define **Bus18_182MHz** 21
- #define **Bus17_391MHz** 22
- #define **Bus16_667MHz** 23
- #define **Bus16_000MHz** 24
- #define **Bus16MHz** 24
- #define **Bus15_385MHz** 25
- #define **Bus14_815MHz** 26
- #define **Bus14_286MHz** 27
- #define **Bus13_793MHz** 28

- #define **Bus13_333MHz** 29
- #define **Bus12_903MHz** 30
- #define **Bus12_500MHz** 31
- #define **Bus12_121MHz** 32
- #define **Bus11_765MHz** 33
- #define **Bus11_429MHz** 34
- #define **Bus11_111MHz** 35
- #define **Bus10_811MHz** 36
- #define **Bus10_526MHz** 37
- #define **Bus10_256MHz** 38
- #define **Bus10_000MHz** 39
- #define **Bus10MHz** 39
- #define **Bus9_756MHz** 40
- #define **Bus9_524MHz** 41
- #define **Bus9_302MHz** 42
- #define **Bus9_091MHz** 43
- #define **Bus8_889MHz** 44
- #define **Bus8_696MHz** 45
- #define **Bus8_511MHz** 46
- #define **Bus8_333MHz** 47
- #define **Bus8_163MHz** 48
- #define **Bus8_000MHz** 49
- #define **Bus8MHz** 49
- #define **Bus7_843MHz** 50
- #define **Bus7_692MHz** 51
- #define **Bus7_547MHz** 52
- #define **Bus7_407MHz** 53
- #define **Bus7_273MHz** 54
- #define **Bus7_143MHz** 55
- #define **Bus7_018MHz** 56
- #define **Bus6_897MHz** 57
- #define **Bus6_780MHz** 58
- #define **Bus6_667MHz** 59
- #define **Bus6_557MHz** 60
- #define **Bus6_452MHz** 61
- #define **Bus6_349MHz** 62
- #define **Bus6_250MHz** 63
- #define **Bus6_154MHz** 64
- #define **Bus6_061MHz** 65
- #define **Bus5_970MHz** 66
- #define **Bus5_882MHz** 67
- #define **Bus5_797MHz** 68
- #define **Bus5_714MHz** 69
- #define **Bus5_634MHz** 70
- #define **Bus5_556MHz** 71
- #define **Bus5_479MHz** 72
- #define **Bus5_405MHz** 73
- #define **Bus5_333MHz** 74
- #define **Bus5_263MHz** 75
- #define **Bus5_195MHz** 76
- #define **Bus5_128MHz** 77
- #define **Bus5_063MHz** 78
- #define **Bus5_000MHz** 79
- #define **Bus4_938MHz** 80
- #define **Bus4_878MHz** 81

- #define **Bus4_819MHz** 82
- #define **Bus4_762MHz** 83
- #define **Bus4_706MHz** 84
- #define **Bus4_651MHz** 85
- #define **Bus4_598MHz** 86
- #define **Bus4_545MHz** 87
- #define **Bus4_494MHz** 88
- #define **Bus4_444MHz** 89
- #define **Bus4_396MHz** 90
- #define **Bus4_348MHz** 91
- #define **Bus4_301MHz** 92
- #define **Bus4_255MHz** 93
- #define **Bus4_211MHz** 94
- #define **Bus4_167MHz** 95
- #define **Bus4_124MHz** 96
- #define **Bus4_082MHz** 97
- #define **Bus4_040MHz** 98
- #define **Bus4_000MHz** 99
- #define **Bus4MHz** 99
- #define **Bus3_960MHz** 100
- #define **Bus3_922MHz** 101
- #define **Bus3_883MHz** 102
- #define **Bus3_846MHz** 103
- #define **Bus3_810MHz** 104
- #define **Bus3_774MHz** 105
- #define **Bus3_738MHz** 106
- #define **Bus3_704MHz** 107
- #define **Bus3_670MHz** 108
- #define **Bus3_636MHz** 109
- #define **Bus3_604MHz** 110
- #define **Bus3_571MHz** 111
- #define **Bus3_540MHz** 112
- #define **Bus3_509MHz** 113
- #define **Bus3_478MHz** 114
- #define **Bus3_448MHz** 115
- #define **Bus3_419MHz** 116
- #define **Bus3_390MHz** 117
- #define **Bus3_361MHz** 118
- #define **Bus3_333MHz** 119
- #define **Bus3_306MHz** 120
- #define **Bus3_279MHz** 121
- #define **Bus3_252MHz** 122
- #define **Bus3_226MHz** 123
- #define **Bus3_200MHz** 124
- #define **Bus3_175MHz** 125
- #define **Bus3_150MHz** 126
- #define **Bus3_125MHz** 127

Functions

- void PLL_Init (uint32_t freq)

  configure the system to get its clock from the PLL

4.5.1 Detailed Description

Runs on LM4F120/TM4C123 A software function to change the bus frequency using the PLL.

Author

Daniel Valvano

4.5.2 Function Documentation

4.5.2.1 void PLL_Init (uint32_t *freq*)

configure the system to get its clock from the PLL

Parameters

freq	Macro defined in PLL.h to choose frequency

4.6 inc/profiler.h File Reference

Thread profiler utility.

Data Structures

- struct event_t

Macros

- #define **EVENT_MAGIC** (0x02344629)
- #define **MAX_EVENTS** (100)

Enumerations

- enum **event_type_e** { **EVENT_FGTH_START**, **EVENT_PTH_START**, **EVENT_PTH_END**, **EVENT_NUM↩ _TYPES** }

Functions

- void Profiler_Init (void)

  Initialize the thread profiler. Call before use.
- int Profiler_Event (event_type_e event_type, char ∗event_name)

  Register an event has occurred in the profiler.
- void Profiler_Clear (void)

  Clear profiler history.
- void Profiler_Foreach (void(∗f)(const event_t ∗))

  Executes a function f on each event in the log in the order they occurred in the system.

4.6.1 Detailed Description

Thread profiler utility.

Author

Riley Wood (riley.wood@utexas.edu)

4.6.2 Function Documentation

4.6.2.1 int Profiler_Event (event_type_e *event_type*, char * *event_name*)

Register an event has occurred in the profiler.

Parameters

event_id	ID of the event that occurred

Returns

-1 on error, 0 on success

4.6.2.2 void Profiler_Foreach (void(*)(const event_t *) *f*)

Executes a function f on each event in the log in the order they occurred in the system.

Parameters

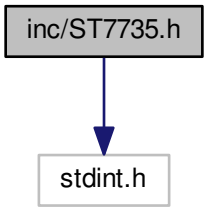
f	Function to execute on each event in the log.

4.7 inc/ST7735.h File Reference

This is a library for the Adafruit 1.8" SPI display.

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

Include dependency graph for ST7735.h:



Macros

- #define **ST7735_TFTWIDTH** 128
- #define **ST7735_TFTHEIGHT** 160
- #define **ST7735_BLACK** 0x0000
- #define **ST7735_BLUE** 0xF800
- #define **ST7735_RED** 0x001F
- #define **ST7735_GREEN** 0x07E0
- #define **ST7735_CYAN** 0xFFE0
- #define **ST7735_MAGENTA** 0xF81F
- #define **ST7735_YELLOW** 0x07FF
- #define **ST7735_WHITE** 0xFFFF

Enumerations

- enum initRFlags { **none**, **INITR_GREENTAB**, **INITR_REDTAB**, **INITR_BLACKTAB** }

  some flags for ST7735_InitR()

Functions

- void ST7735_InitB (void)

  Initialization for ST7735B screens.
- void ST7735_InitR (enum initRFlags option)

  Initialization for ST7735R screens (green or red tabs).
- void ST7735_DrawPixel (int16_t x, int16_t y, uint16_t color)

  Color the pixel at the given coordinates with the given color. Requires 13 bytes of transmission.
- void ST7735_DrawFastVLine (int16_t x, int16_t y, int16_t h, uint16_t color)

  Draw a vertical line at the given coordinates with the given height and color. A vertical line is parallel to the longer side of the rectangular display Requires (11 + 2∗h) bytes of transmission (assuming image fully on screen)
- void ST7735_DrawFastHLine (int16_t x, int16_t y, int16_t w, uint16_t color)

  Draw a horizontal line at the given coordinates with the given width and color. A horizontal line is parallel to the shorter side of the rectangular display Requires (11 + 2∗w) bytes of transmission (assuming image fully on screen)
- void ST7735_FillScreen (uint16_t color)

  Fill the screen with the given color. Requires 40,971 bytes of transmission.

- void ST7735_FillRect (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

  Draw a filled rectangle at the given coordinates with the given width, height, and color. Requires (11 + 2∗w∗h) bytes of transmission (assuming image fully on screen)
- uint16_t ST7735_Color565 (uint8_t r, uint8_t g, uint8_t b)

  Pass 8-bit (each) R,G,B and get back 16-bit packed color.
- uint16_t ST7735_SwapColor (uint16_t x)

  Swaps the red and blue values of the given 16-bit packed color; green is unchanged.
- void ST7735_DrawBitmap (int16_t x, int16_t y, const uint16_t ∗image, int16_t w, int16_t h)

  Displays a 16-bit color BMP image. A bitmap file that is created by a PC image processing program has a header and may be padded with dummy columns so the data have four byte alignment. This function assumes that all of that has been stripped out, and the array image[] has one 16-bit halfword for each pixel to be displayed on the screen (encoded in reverse order, which is standard for bitmap files). An array can be created in this format from a 24-bit-per-pixel .bmp file using the associated converter program. (x,y) is the screen location of the lower left corner of BMP image Requires (11 + 2∗w∗h) bytes of transmission (assuming image fully on screen) Must be less than or equal to 128 pixels wide by 160 pixels high.
- void ST7735_DrawCharS (int16_t x, int16_t y, char c, int16_t textColor, int16_t bgColor, uint8_t size)

  Simple character draw function. This is the same function from Adafruit_GFX.c but adapted for this processor. However, each call to ST7735_DrawPixel() calls setAddrWindow(), which needs to send many extra data and commands. If the background color is the same as the text color, no background will be printed, and text can be drawn right over existing images without covering them with a box. Requires (11 + 2∗size∗size)∗6∗8 (image fully on screen; textcolor != bgColor)
- void ST7735_DrawChar (int16_t x, int16_t y, char c, int16_t textColor, int16_t bgColor, uint8_t size)

  Advanced character draw function. This is similar to the function from Adafruit_GFX.c but adapted for this processor. However, this function only uses one call to setAddrWindow(), which allows it to run at least twice as fast. Requires (11 + size∗size∗6∗8) bytes of transmission (assuming image fully on screen)
- uint32_t ST7735_DrawString (uint16_t x, uint16_t y, char ∗pt, int16_t textColor, int16_t bgColor)

  String draw function. 16 rows (0 to 15) and 21 characters (0 to 20) Requires (11 + size∗size∗6∗8) bytes of transmission for each character If bgColor is same as textColor, no background will be filled in for chars.
- void ST7735_SetCursor (uint32_t newX, uint32_t newY)

  Move the cursor to the desired X- and Y-position. The next character will be printed here. X=0 is the leftmost column. Y=0 is the top row.
- void ST7735_OutUDec (uint32_t n)

  Output a 32-bit number in unsigned decimal format Position determined by ST7735_SetCursor command Color set by ST7735_SetTextColor.
- void ST7735_SetRotation (uint8_t m)

  Change the image rotation. Requires 2 bytes of transmission.
- void ST7735_InvertDisplay (int i)

  Send the command to invert all of the colors. Requires 1 byte of transmission.
- void ST7735_PlotClear (int32_t ymin, int32_t ymax)

  Clear the graphics buffer, set X coordinate to 0 This routine clears the display.
- void ST7735_PlotPoint (int32_t y)

  Used in the voltage versus time plot, plot one point at y It does output to display.
- void ST7735_PlotLine (int32_t y)

  Used in the voltage versus time plot, plot line to new point It does output to display.
- void ST7735_PlotPoints (int32_t y1, int32_t y2)

  Used in the voltage versus time plot, plot two points at y1, y2 It does output to display.
- void ST7735_PlotBar (int32_t y)

  Used in the voltage versus time bar, plot one bar at y It does not output to display until RIT128x96x4ShowPlot called.
- void ST7735_PlotdBfs (int32_t y)

  Used in the amplitude versus frequency plot, plot bar point at y 0 to 0.625V scaled on a log plot from min to max It does output to display.
- void ST7735_PlotNext (void)

  Used in all the plots to step the X coordinate one pixel X steps from 0 to 127, then back to 0 again It does not output to display.

- void ST7735_PlotNextErase (void)

  Used in all the plots to step the X coordinate one pixel X steps from 0 to 127, then back to 0 again It clears the vertical space into which the next pixel will be drawn.
- void ST7735_OutChar (char ch)

  Output one character to the LCD Position determined by ST7735_SetCursor command Color set by ST7735_Set↩ TextColor.
- void ST7735_OutString (char ∗ptr)

  Print a string of characters to the ST7735 LCD. Position determined by ST7735_SetCursor command Color set by ST7735_SetTextColor The string will not automatically wrap.
- void ST7735_SetTextColor (uint16_t color)

  Sets the color in which the characters will be printed Background color is fixed at black.
- void Output_Init (void)

  Standard device driver initialization function for printf Initialize ST7735 LCD.
- void Output_Clear (void)

  Clear display.
- void Output_Off (void)

  Turn off display (low power)
- void Output_On (void)

  Turn on display.
- void Output_Color (uint32_t newColor)

  set the color for future output Background color is fixed at black
- void ST7735_Message (int device, int line, char ∗string, int32_t value)

  Display a string and number on one of two logical displays at a given line number relative to that display. The LCD display is logically divided into two displays: top and bottom. These logical displays are identified with a device ID. Device 0 is the top display, device 1 is the bottom display. Each logical device has 4 lines, numbered 0 to 3. Prints in black text on a white background. This function is not (yet) reentrant.

4.7.1 Detailed Description

This is a library for the Adafruit 1.8" SPI display.

4.7.2 Function Documentation

4.7.2.1 void Output_Color (uint32_t *newColor*)

set the color for future output Background color is fixed at black

Parameters

newColor	16-bit packed color

4.7.2.2 uint16_t ST7735_Color565 (uint8_t *r*, uint8_t *g*, uint8_t *b*)

Pass 8-bit (each) R,G,B and get back 16-bit packed color.

Parameters

r	red value

Parameters

g	green value
b	blue value

Returns

uint16_t 16-bit color

4.7.2.3 void ST7735_DrawBitmap (int16_t *x*, int16_t *y*, const uint16_t * *image*, int16_t *w*, int16_t *h*)

Displays a 16-bit color BMP image. A bitmap file that is created by a PC image processing program has a header and may be padded with dummy columns so the data have four byte alignment. This function assumes that all of that has been stripped out, and the array image[] has one 16-bit halfword for each pixel to be displayed on the screen (encoded in reverse order, which is standard for bitmap files). An array can be created in this format from a 24-bit-per-pixel .bmp file using the associated converter program. (x,y) is the screen location of the lower left corner of BMP image Requires (11 + 2∗w∗h) bytes of transmission (assuming image fully on screen) Must be less than or equal to 128 pixels wide by 160 pixels high.

Parameters

x	horizontal position of the bottom left corner of the image, columns from the left edge
y	vertical position of the bottom left corner of the image, rows from the top edge
image	pointer to a 16-bit color BMP image
w	number of pixels wide
h	number of pixels tall

4.7.2.4 void ST7735_DrawChar (int16_t *x*, int16_t *y*, char *c*, int16_t *textColor*, int16_t *bgColor*, uint8_t *size*)

Advanced character draw function. This is similar to the function from Adafruit_GFX.c but adapted for this processor. However, this function only uses one call to setAddrWindow(), which allows it to run at least twice as fast. Requires (11 + size∗size∗6∗8) bytes of transmission (assuming image fully on screen)

Parameters

x	horizontal position of the top left corner of the character, columns from the left edge
y	vertical position of the top left corner of the character, rows from the top edge
c	character to be printed
textColor	16-bit color of the character
bgColor	16-bit color of the background
size	number of pixels per character pixel (e.g. size==2 prints each pixel of font as 2x2 square)

4.7.2.5 void ST7735_DrawCharS (int16_t *x*, int16_t *y*, char *c*, int16_t *textColor*, int16_t *bgColor*, uint8_t *size*)

Simple character draw function. This is the same function from Adafruit_GFX.c but adapted for this processor. However, each call to ST7735_DrawPixel() calls setAddrWindow(), which needs to send many extra data and commands. If the background color is the same as the text color, no background will be printed, and text can be drawn

right over existing images without covering them with a box. Requires (11 + 2∗size∗size)∗6∗8 (image fully on screen; textcolor != bgColor)

Parameters

x	horizontal position of the top left corner of the character, columns from the left edge
y	vertical position of the top left corner of the character, rows from the top edge
c	character to be printed
textColor	16-bit color of the character
bgColor	16-bit color of the background
size	number of pixels per character pixel (e.g. size==2 prints each pixel of font as 2x2 square)

4.7.2.6 void ST7735_DrawFastHLine (int16_t *x*, int16_t *y*, int16_t *w*, uint16_t *color*)

Draw a horizontal line at the given coordinates with the given width and color. A horizontal line is parallel to the shorter side of the rectangular display Requires (11 + 2∗w) bytes of transmission (assuming image fully on screen)

Parameters

x	horizontal position of the start of the line, columns from the left edge
y	vertical position of the start of the line, rows from the top edge
w	horizontal width of the line
color	16-bit color, which can be produced by ST7735_Color565()

4.7.2.7 void ST7735_DrawFastVLine (int16_t *x*, int16_t *y*, int16_t *h*, uint16_t *color*)

Draw a vertical line at the given coordinates with the given height and color. A vertical line is parallel to the longer side of the rectangular display Requires (11 + 2∗h) bytes of transmission (assuming image fully on screen)

Parameters

x	horizontal position of the start of the line, columns from the left edge
y	vertical position of the start of the line, rows from the top edge
h	vertical height of the line
color	16-bit color, which can be produced by ST7735_Color565()

4.7.2.8 void ST7735_DrawPixel (int16_t *x*, int16_t *y*, uint16_t *color*)

Color the pixel at the given coordinates with the given color. Requires 13 bytes of transmission.

Parameters

x	horizontal position of the pixel, columns from the left edge must be less than 128 0 is on the left, 126 is near the right
y	vertical position of the pixel, rows from the top edge must be less than 160 159 is near the wires, 0 is the side opposite the wires
color	16-bit color, which can be produced by ST7735_Color565()

4.7.2.9 uint32_t ST7735_DrawString (uint16_t *x*, uint16_t *y*, char * *pt*, int16_t *textColor*, int16_t *bgColor*)

String draw function. 16 rows (0 to 15) and 21 characters (0 to 20) Requires (11 + size*size*6*8) bytes of transmission for each character If bgColor is same as textColor, no background will be filled in for chars.

Parameters

x	columns from the left edge (0 to 20)
y	rows from the top edge (0 to 15)
pt	pointer to a null terminated string to be printed
textColor	16-bit color of the characters
bgColor	16-bit color of the background

Returns

uint32_t number of characters printed

4.7.2.10 void ST7735_FillRect (int16_t *x*, int16_t *y*, int16_t *w*, int16_t *h*, uint16_t *color*)

Draw a filled rectangle at the given coordinates with the given width, height, and color. Requires (11 + 2*w*h) bytes of transmission (assuming image fully on screen)

Parameters

x	horizontal position of the top left corner of the rectangle, columns from the left edge
y	vertical position of the top left corner of the rectangle, rows from the top edge
w	horizontal width of the rectangle
h	vertical height of the rectangle
color	16-bit color, which can be produced by ST7735_Color565()

4.7.2.11 void ST7735_FillScreen (uint16_t *color*)

Fill the screen with the given color. Requires 40,971 bytes of transmission.

Parameters

color	16-bit color, which can be produced by ST7735_Color565()

4.7.2.12 void ST7735_InitR (enum **initRFlags** *option*)

Initialization for ST7735R screens (green or red tabs).

Parameters

initRFlags	one of the enumerated options depending on tabs

4.7.2.13 void ST7735_InvertDisplay (int *i*)

Send the command to invert all of the colors. Requires 1 byte of transmission.

Parameters

i	0 to disable inversion; non-zero to enable inversion

4.7.2.14 void ST7735_Message (int *device,* int *line,* char * *string,* int32_t *value*)

Display a string and number on one of two logical displays at a given line number relative to that display. The LCD display is logically divided into two displays: top and bottom. These logical displays are identified with a device ID. Device 0 is the top display, device 1 is the bottom display. Each logical device has 4 lines, numbered 0 to 3. Prints in black text on a white background. This function is not (yet) reentrant.

Parameters

device	Device ID, 0 or 1
line	Line number, 0 to 3, relative to the logical display.
string	Null-terminated string to print on the select logical display and line.
value	Integer value printed after the string.

4.7.2.15 void ST7735_OutChar (char *ch*)

Output one character to the LCD Position determined by ST7735_SetCursor command Color set by ST7735_Set↩ TextColor.

Parameters

ch	8-bit ASCII character

4.7.2.16 void ST7735_OutString (char * *ptr*)

Print a string of characters to the ST7735 LCD. Position determined by ST7735_SetCursor command Color set by ST7735_SetTextColor The string will not automatically wrap.

Parameters

ptr	pointer to NULL-terminated ASCII string

4.7.2.17 void ST7735_OutUDec (uint32_t *n*)

Output a 32-bit number in unsigned decimal format Position determined by ST7735_SetCursor command Color set by ST7735_SetTextColor.

Parameters

n	32-bit number to be transferred

4.7.2.18 void ST7735_PlotBar (int32_t *y*)

Used in the voltage versus time bar, plot one bar at y It does not output to display until RIT128x96x4ShowPlot called.

Parameters

y	the y coordinate of the bar plotted

4.7.2.19 void ST7735_PlotClear (int32_t *ymin,* int32_t *ymax*)

Clear the graphics buffer, set X coordinate to 0 This routine clears the display.

Parameters

ymin	Lower bound of plot
ymax	Upper bound of plot

4.7.2.20 void ST7735_PlotdBfs (int32_t *y*)

Used in the amplitude versus frequency plot, plot bar point at y 0 to 0.625V scaled on a log plot from min to max It does output to display.

Parameters

y	the y ADC value of the bar plotted

4.7.2.21 void ST7735_PlotLine (int32_t *y*)

Used in the voltage versus time plot, plot line to new point It does output to display.

Parameters

y	the y coordinate of the point plotted

4.7.2.22 void ST7735_PlotPoint (int32_t *y*)

Used in the voltage versus time plot, plot one point at y It does output to display.

Parameters

y	the y coordinate of the point plotted

4.7.2.23 void ST7735_PlotPoints (int32_t *y1,* int32_t *y2*)

Used in the voltage versus time plot, plot two points at y1, y2 It does output to display.

Parameters

y1	the y coordinate of the first point plotted
y2	the y coordinate of the second point plotted

4.7.2.24 void ST7735_SetCursor (uint32_t *newX,* uint32_t *newY*)

Move the cursor to the desired X- and Y-position. The next character will be printed here. X=0 is the leftmost column. Y=0 is the top row.

Parameters

newX	new X-position of the cursor (0<=newX<=20)
newY	new Y-position of the cursor (0<=newY<=15)

4.7.2.25 void ST7735_SetRotation (uint8_t *m*)

Change the image rotation. Requires 2 bytes of transmission.

Parameters

m	new rotation value (0 to 3)

4.7.2.26 void ST7735_SetTextColor (uint16_t *color*)

Sets the color in which the characters will be printed Background color is fixed at black.

Parameters

color	16-bit packed color

4.7.2.27 uint16_t ST7735_SwapColor (uint16_t *x*)

Swaps the red and blue values of the given 16-bit packed color; green is unchanged.

Parameters

x	16-bit color in format B, G, R

Returns

uint16_t 16-bit color in format R, G, B

4.8 inc/UART.h File Reference

Runs on LM4F120/TM4C123 Use UART0 to implement bidirectional data transfer to and from a computer running HyperTerminal. This time, interrupts and FIFOs are used.

Macros

- #define **CR** 0x0D
- #define **LF** 0x0A
- #define **BS** 0x08
- #define **ESC** 0x1B
- #define **SP** 0x20
- #define **DEL** 0x7F

Functions

- void UART_Init (void)

  Initialize the UART for 115,200 baud rate (assuming 50 MHz clock), 8 bit word length, no parity bits, one stop bit, FIFOs enabled.
- char UART_InChar (void)

  Wait for new serial port input.
- void UART_OutChar (char data)

  8-bit to serial port
- void UART_OutString (char ∗pt)

  Output String (NULL termination)
- uint32_t UART_InUDec (void)

  InUDec accepts ASCII input in unsigned decimal format and converts to a 32-bit unsigned number valid range is 0 to 4294967295 (2^32-1) If you enter a number above 4294967295, it will return an incorrect value Backspace will remove last digit typed.
- void UART_OutUDec (uint32_t n)

  Output a 32-bit number in unsigned decimal format.
- uint32_t UART_InUHex (void)

  Accepts ASCII input in unsigned hexadecimal (base 16) format No '$' or '0x' need be entered, just the 1 to 8 hex digits It will convert lower case a-f to uppercase A-F and converts to a 16 bit unsigned number value range is 0 to FFFFFFFF If you enter a number above FFFFFFFF, it will return an incorrect value Backspace will remove last digit typed.
- void UART_OutUHex (uint32_t number)

  Output a 32-bit number in unsigned hexadecimal format Variable format 1 to 8 digits with no space before or after.
- void UART_InString (char ∗bufPt, uint16_t max)

  Accepts ASCII characters from the serial port and adds them to a string until <enter> is typed or until max length of the string is reached. It echoes each character as it is inputted. If a backspace is inputted, the string is modified and the backspace is echoed terminates the string with a null character uses busy-waiting synchronization on RDRF Modified by Agustinus Darmawan + Mingjie Qiu.

4.8.1 Detailed Description

Runs on LM4F120/TM4C123 Use UART0 to implement bidirectional data transfer to and from a computer running HyperTerminal. This time, interrupts and FIFOs are used.

Author

Daniel Valvano

4.8.2 Function Documentation

4.8.2.1 char UART_InChar (void)

Wait for new serial port input.

Returns

char ASCII code for key typed

4.8.2.2 void UART_InString (char * *bufPt,* uint16_t *max*)

Accepts ASCII characters from the serial port and adds them to a string until <enter> is typed or until max length of the string is reached. It echoes each character as it is inputted. If a backspace is inputted, the string is modified and the backspace is echoed terminates the string with a null character uses busy-waiting synchronization on RDRF Modified by Agustinus Darmawan + Mingjie Qiu.

Parameters

bufPt	pointer to empty buffer
max	size of buffer

4.8.2.3 uint32_t UART_InUDec (void)

InUDec accepts ASCII input in unsigned decimal format and converts to a 32-bit unsigned number valid range is 0 to 4294967295 (2^32-1) If you enter a number above 4294967295, it will return an incorrect value Backspace will remove last digit typed.

Returns

uint32_t 32-bit unsigned number

4.8.2.4 uint32_t UART_InUHex (void)

Accepts ASCII input in unsigned hexadecimal (base 16) format No '$' or '0x' need be entered, just the 1 to 8 hex digits It will convert lower case a-f to uppercase A-F and converts to a 16 bit unsigned number value range is 0 to FFFFFFFF If you enter a number above FFFFFFFF, it will return an incorrect value Backspace will remove last digit typed.

Returns

uint32_t 32-bit unsigned number

4.8.2.5 void UART_OutChar (char *data*)

8-bit to serial port

Parameters

data	letter is an 8-bit ASCII character to be transferred

4.8.2.6 void UART_OutString (char * *pt*)

Output String (NULL termination)

Parameters

pt	pointer to a NULL-terminated string to be transferred

4.8.2.7 void UART_OutUDec (uint32_t *n*)

Output a 32-bit number in unsigned decimal format.

Parameters

n	32-bit number to be transferred

4.8.2.8 void UART_OutUHex (uint32_t *number*)

Output a 32-bit number in unsigned hexadecimal format Variable format 1 to 8 digits with no space before or after.

Parameters

number	32-bit number to be transferred

Index