

EE445M/EE380L Lab 5 Documentation

Generated by Doxygen 1.8.11

Contents

| | | |
|----------|---|----------|
| 1 | Todo List | 1 |
| 2 | Module Index | 3 |
| 2.1 | Modules | 3 |
| 3 | Data Structure Index | 5 |
| 3.1 | Data Structures | 5 |
| 4 | File Index | 7 |
| 4.1 | File List | 7 |
| 5 | Module Documentation | 9 |
| 5.1 | ELF Loader | 9 |
| 5.1.1 | Detailed Description | 9 |
| 5.1.2 | Enumeration Type Documentation | 9 |
| 5.1.2.1 | ELFSecPerm_t | 9 |
| 5.1.3 | Function Documentation | 9 |
| 5.1.3.1 | exec_elf(const char *path, const ELFEnv_t *env) | 9 |

| | |
|--|-----------|
| 6 Data Structure Documentation | 11 |
| 6.1 <code>_tcb_s</code> Struct Reference | 11 |
| 6.2 DIR Struct Reference | 12 |
| 6.2.1 Detailed Description | 12 |
| 6.3 <code>Elf32_Dyn</code> Struct Reference | 12 |
| 6.4 <code>Elf32_Ehdr</code> Struct Reference | 13 |
| 6.5 <code>Elf32_Phdr</code> Struct Reference | 13 |
| 6.6 <code>Elf32_Rel</code> Struct Reference | 13 |
| 6.7 <code>Elf32_Rela</code> Struct Reference | 14 |
| 6.8 <code>Elf32_Shdr</code> Struct Reference | 14 |
| 6.9 <code>Elf32_Sym</code> Struct Reference | 14 |
| 6.10 <code>ELFEnv_t</code> Struct Reference | 15 |
| 6.10.1 Detailed Description | 15 |
| 6.10.2 Field Documentation | 15 |
| 6.10.2.1 <code>exported</code> | 15 |
| 6.10.2.2 <code>exported_size</code> | 15 |
| 6.11 <code>ELFSymbol_t</code> Struct Reference | 15 |
| 6.11.1 Detailed Description | 16 |
| 6.11.2 Field Documentation | 16 |
| 6.11.2.1 <code>name</code> | 16 |
| 6.11.2.2 <code>ptr</code> | 16 |
| 6.12 <code>event_t</code> Struct Reference | 16 |
| 6.13 FATFS Struct Reference | 16 |
| 6.13.1 Detailed Description | 17 |
| 6.14 FIL Struct Reference | 17 |
| 6.14.1 Detailed Description | 18 |
| 6.15 FILINFO Struct Reference | 18 |
| 6.15.1 Detailed Description | 18 |
| 6.16 <code>heap_stats</code> Struct Reference | 19 |
| 6.17 <code>Sema4</code> Struct Reference | 19 |

| | |
|--|-----------|
| 7 File Documentation | 21 |
| 7.1 inc/ADC.h File Reference | 21 |
| 7.1.1 Detailed Description | 22 |
| 7.1.2 Function Documentation | 22 |
| 7.1.2.1 ADC_Collect(uint32_t channelNum, uint32_t fs, void(*handler)(unsigned long)) | 22 |
| 7.1.2.2 ADC_In(void) | 22 |
| 7.1.2.3 ADC_Init(uint32_t channelNum) | 22 |
| 7.2 inc/diskio.h File Reference | 23 |
| 7.2.1 Detailed Description | 24 |
| 7.2.2 Function Documentation | 24 |
| 7.2.2.1 disk_initialize(BYTE drv) | 24 |
| 7.2.2.2 disk_ioctl(BYTE drv, BYTE cmd, void *buff) | 25 |
| 7.2.2.3 disk_read(BYTE drv, BYTE *buff, DWORD sector, UINT count) | 25 |
| 7.2.2.4 disk_status(BYTE drv) | 25 |
| 7.2.2.5 disk_write(BYTE drv, const BYTE *buff, DWORD sector, UINT count) | 26 |
| 7.3 inc/ff.h File Reference | 26 |
| 7.3.1 Detailed Description | 29 |
| 7.3.2 Function Documentation | 29 |
| 7.3.2.1 f_chdir(const TCHAR *path) | 29 |
| 7.3.2.2 f_chdrive(const TCHAR *path) | 29 |
| 7.3.2.3 f_chmod(const TCHAR *path, BYTE value, BYTE mask) | 29 |
| 7.3.2.4 f_close(FIL *fp) | 29 |
| 7.3.2.5 f_closedir(DIR *dp) | 29 |
| 7.3.2.6 f_disk(BYTE pdrv, const DWORD szt[], void *work) | 30 |
| 7.3.2.7 f_forward(FIL *fp, UINT(*func)(const BYTE *, UINT), UINT btf, UINT *bf) | 30 |
| 7.3.2.8 f_getcwd(TCHAR *buff, UINT len) | 30 |
| 7.3.2.9 f_getfree(const TCHAR *path, DWORD *nclst, FATFS **fatfs) | 30 |
| 7.3.2.10 f_getlabel(const TCHAR *path, TCHAR *label, DWORD *vsn) | 30 |
| 7.3.2.11 f_gets(TCHAR *buff, int len, FIL *fp) | 30 |
| 7.3.2.12 f_lseek(FIL *fp, DWORD ofs) | 30 |

| | | |
|----------|--|----|
| 7.3.2.13 | f_mkdir(const TCHAR *path) | 30 |
| 7.3.2.14 | f_mkfs(const TCHAR *path, BYTE sf, UINT au) | 30 |
| 7.3.2.15 | f_mount(FATFS *fs, const TCHAR *path, BYTE opt) | 30 |
| 7.3.2.16 | f_open(FIL *fp, const TCHAR *path, BYTE mode) | 31 |
| 7.3.2.17 | f_opendir(DIR *dp, const TCHAR *path) | 31 |
| 7.3.2.18 | f_printf(FIL *fp, const TCHAR *str,...) | 31 |
| 7.3.2.19 | f_putc(TCHAR c, FIL *fp) | 31 |
| 7.3.2.20 | f_puts(const TCHAR *str, FIL *cp) | 31 |
| 7.3.2.21 | f_read(FIL *fp, void *buff, UINT btr, UINT *br) | 31 |
| 7.3.2.22 | f_readdir(DIR *dp, FILINFO *fno) | 31 |
| 7.3.2.23 | f_rename(const TCHAR *path_old, const TCHAR *path_new) | 31 |
| 7.3.2.24 | f_setlabel(const TCHAR *label) | 31 |
| 7.3.2.25 | f_stat(const TCHAR *path, FILINFO *fno) | 31 |
| 7.3.2.26 | f_sync(FIL *fp) | 32 |
| 7.3.2.27 | f_truncate(FIL *fp) | 32 |
| 7.3.2.28 | f_unlink(const TCHAR *path) | 32 |
| 7.3.2.29 | f_ftime(const TCHAR *path, const FILINFO *fno) | 32 |
| 7.3.2.30 | f_write(FIL *fp, const void *buff, UINT btw, UINT *bw) | 32 |
| 7.4 | inc/heap.h File Reference | 32 |
| 7.4.1 | Detailed Description | 33 |
| 7.4.2 | Function Documentation | 34 |
| 7.4.2.1 | Heap_Calloc(int32_t desiredBytes) | 34 |
| 7.4.2.2 | Heap_Free(void *pointer) | 34 |
| 7.4.2.3 | Heap_Init(void) | 34 |
| 7.4.2.4 | Heap_Malloc(int32_t desiredBytes) | 34 |
| 7.4.2.5 | Heap_Realloc(void *oldBlock, int32_t desiredBytes) | 35 |
| 7.4.2.6 | Heap_Stats(void) | 35 |
| 7.4.2.7 | Heap_Test(void) | 35 |
| 7.5 | inc/interpreter.h File Reference | 35 |
| 7.5.1 | Detailed Description | 36 |

| | | |
|----------|--|----|
| 7.5.2 | Function Documentation | 36 |
| 7.5.2.1 | interpreter_cmd(char *cmd_str) | 36 |
| 7.6 | inc/misc_macros.h File Reference | 37 |
| 7.6.1 | Detailed Description | 37 |
| 7.7 | inc/OS.h File Reference | 37 |
| 7.7.1 | Detailed Description | 39 |
| 7.7.2 | Macro Definition Documentation | 39 |
| 7.7.2.1 | OS_AddPeriodicThread | 39 |
| 7.7.2.2 | OS_AddThread | 40 |
| 7.7.3 | Function Documentation | 40 |
| 7.7.3.1 | OS_AddSW1Task(void(*task)(void), unsigned long priority) | 40 |
| 7.7.3.2 | OS_AddSW2Task(void(*task)(void), unsigned long priority) | 40 |
| 7.7.3.3 | OS_bSignal(Sema4Type *semaPt) | 41 |
| 7.7.3.4 | OS_bWait(Sema4Type *semaPt) | 41 |
| 7.7.3.5 | OS_ClearMsTime(void) | 41 |
| 7.7.3.6 | OS_Fifo_Get(void) | 41 |
| 7.7.3.7 | OS_Fifo_Init(unsigned long size) | 42 |
| 7.7.3.8 | OS_Fifo_Put(unsigned long data) | 42 |
| 7.7.3.9 | OS_Fifo_Size(void) | 42 |
| 7.7.3.10 | OS_Id(void) | 42 |
| 7.7.3.11 | OS_Init(void) | 43 |
| 7.7.3.12 | OS_InitSemaphore(Sema4Type *semaPt, long value) | 43 |
| 7.7.3.13 | OS_Kill(void) | 44 |
| 7.7.3.14 | OS_Launch(unsigned long theTimeSlice) | 44 |
| 7.7.3.15 | OS_MailBox_Init(void) | 44 |
| 7.7.3.16 | OS_MailBox_Recv(void) | 44 |
| 7.7.3.17 | OS_MailBox_Send(unsigned long data) | 44 |
| 7.7.3.18 | OS_MsTime(void) | 45 |
| 7.7.3.19 | OS_Signal(Sema4Type *semaPt) | 45 |
| 7.7.3.20 | OS_Sleep(unsigned long sleepTime) | 45 |

| | | |
|-----------|--|----|
| 7.7.3.21 | OS_Suspend(void) | 45 |
| 7.7.3.22 | OS_Time(void) | 45 |
| 7.7.3.23 | OS_TimeDifference(unsigned long long start, unsigned long long stop) | 46 |
| 7.7.3.24 | OS_Wait(Sema4Type *semaPt) | 46 |
| 7.8 | inc/PLL.h File Reference | 46 |
| 7.8.1 | Detailed Description | 49 |
| 7.8.2 | Function Documentation | 49 |
| 7.8.2.1 | PLL_Init(uint32_t freq) | 49 |
| 7.9 | inc/profiler.h File Reference | 49 |
| 7.9.1 | Detailed Description | 50 |
| 7.9.2 | Function Documentation | 50 |
| 7.9.2.1 | Profiler_Event(event_type_e event_type, char *event_name) | 50 |
| 7.9.2.2 | Profiler_Foreach(void(*f)(const event_t *)) | 50 |
| 7.10 | inc/ST7735.h File Reference | 51 |
| 7.10.1 | Detailed Description | 52 |
| 7.11 | inc/ST7735_lab3.h File Reference | 52 |
| 7.11.1 | Detailed Description | 55 |
| 7.11.2 | Function Documentation | 55 |
| 7.11.2.1 | Output_Color(uint32_t newColor) | 55 |
| 7.11.2.2 | ST7735_Color565(uint8_t r, uint8_t g, uint8_t b) | 55 |
| 7.11.2.3 | ST7735_DrawBitmap(int16_t x, int16_t y, const uint16_t *image, int16_t w, int16_t h) | 55 |
| 7.11.2.4 | ST7735_DrawChar(int16_t x, int16_t y, char c, int16_t textColor, int16_t bgColor, uint8_t size) | 57 |
| 7.11.2.5 | ST7735_DrawCharS(int16_t x, int16_t y, char c, int16_t textColor, int16_t bgColor, uint8_t size) | 57 |
| 7.11.2.6 | ST7735_DrawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color) | 57 |
| 7.11.2.7 | ST7735_DrawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color) | 58 |
| 7.11.2.8 | ST7735_DrawPixel(int16_t x, int16_t y, uint16_t color) | 58 |
| 7.11.2.9 | ST7735_DrawString(uint16_t x, uint16_t y, char *pt, int16_t textColor, int16_t bgColor) | 58 |
| 7.11.2.10 | ST7735_FillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color) | 59 |

| | |
|---|----|
| 7.11.2.11 ST7735_FillScreen(uint16_t color) | 59 |
| 7.11.2.12 ST7735_InitR(enum initRFlags option) | 59 |
| 7.11.2.13 ST7735_InvertDisplay(int i) | 59 |
| 7.11.2.14 ST7735_Message(int device, int line, char *string, int32_t value) | 60 |
| 7.11.2.15 ST7735_OutChar(char ch) | 60 |
| 7.11.2.16 ST7735_OutString(char *ptr) | 60 |
| 7.11.2.17 ST7735_OutUDec(uint32_t n) | 60 |
| 7.11.2.18 ST7735_PlotBar(int32_t y) | 60 |
| 7.11.2.19 ST7735_PlotClear(int32_t ymin, int32_t ymax) | 61 |
| 7.11.2.20 ST7735_PlotDBfs(int32_t y) | 61 |
| 7.11.2.21 ST7735_PlotLine(int32_t y) | 61 |
| 7.11.2.22 ST7735_PlotPoint(int32_t y) | 61 |
| 7.11.2.23 ST7735_PlotPoints(int32_t y1, int32_t y2) | 61 |
| 7.11.2.24 ST7735_SetCursor(uint32_t newX, uint32_t newY) | 62 |
| 7.11.2.25 ST7735_SetRotation(uint8_t m) | 62 |
| 7.11.2.26 ST7735_SetTextColor(uint16_t color) | 62 |
| 7.11.2.27 ST7735_SwapColor(uint16_t x) | 62 |
| 7.12 inc/UART.h File Reference | 63 |
| 7.12.1 Detailed Description | 64 |
| 7.12.2 Function Documentation | 64 |
| 7.12.2.1 UART_InChar(void) | 64 |
| 7.12.2.2 UART_InString(char *bufPt, uint16_t max) | 64 |
| 7.12.2.3 UART_InUDec(void) | 65 |
| 7.12.2.4 UART_InUHex(void) | 65 |
| 7.12.2.5 UART_OutChar(char data) | 65 |
| 7.12.2.6 UART_OutString(char *pt) | 65 |
| 7.12.2.7 UART_OutUDec(uint32_t n) | 66 |
| 7.12.2.8 UART_OutUHex(uint32_t number) | 66 |

Chapter 1

Todo List

Global `exec_elf` (const char *path, const `ELFEnv_t` *env)

Error information

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

| | |
|----------------------|---|
| ELF Loader | 9 |
|----------------------|---|

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|--|----|
| _tcb_s | 11 |
| DIR | |
| Directory object structure (DIR) | 12 |
| Elf32_Dyn | 12 |
| Elf32_Ehdr | 13 |
| Elf32_Phdr | 13 |
| Elf32_Rel | 13 |
| Elf32_Rela | 14 |
| Elf32_Shdr | 14 |
| Elf32_Sym | 14 |
| ELFEnv_t | 15 |
| ELFSymbol_t | 15 |
| event_t | 16 |
| FATFS | |
| File system object structure (FATFS) | 16 |
| FIL | |
| File object structure (FIL) | 17 |
| FILINFO | |
| File status structure (FILINFO) | 18 |
| heap_stats | 19 |
| Sema4 | 19 |

Chapter 4

File Index

4.1 File List

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

| | | |
|--------------------------------|--|----|
| inc/ ADC.h | 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 | 21 |
| inc/ asmdefs.h | | ?? |
| inc/ diskio.h | Low level disk interface module include file (C)ChaN, 2014 converted to TM4C123 Jonathan Valvano, January 13, 2015 | 23 |
| inc/ elf.h | | ?? |
| inc/ ff.h | FatFs - FAT file system module include file R0.10c (C)ChaN, 2014 FatFs module is a generic FAT file system module for small embedded systems. This is a free software that opened for education, research and commercial developments under license policy of following terms. Copyright (C) 2014, , all right reserved. The FatFs module is a free software and there is NO WARRANTY. No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial product UNDER YOUR RESPONSIBILITY. Redistributions of source code must retain the above copyright notice | 26 |
| inc/ ffconf.h | | ?? |
| inc/ FIFO.h | | ?? |
| inc/ heap.h | Implements memory heap for dynamic memory allocation. Follows standard malloc/calloc/realloc/free interface for allocating/unallocating memory. modified 8/31/08 Jonathan Valvano for style modified 12/16/11 Jonathan Valvano for 32-bit machine modified August 10, 2014 for C99 syntax This example accompanies the book "Embedded Systems: Real Time Operating Systems for ARM Cortex M Microcontrollers", ISBN: 978-1466468863, Jonathan Valvano, copyright (c) 2014 Implementation Notes: This is a Knuth Heap. Each block has a header and a trailer, which I shall call the meta-sections. The meta-sections are each a single int32_t that tells how many int32_ts/words can be stored between the header and trailer. If the block is used, the meta-sections record the room as a positive number. If the block is unused, the meta-sections record the room as a negative number. Copyright 2014 by Jonathan W. Valvano, valvano@mail.utexas.edu You may use, edit, run or distribute this file as long as the above copyright notice remains THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER. For more information about my classes, my research, and my books, see http://users.ece.utexas.edu/~valvano/ | 32 |

| | |
|---|----|
| inc/hw_adc.h | ?? |
| inc/hw_aes.h | ?? |
| inc/hw_can.h | ?? |
| inc/hw_ccm.h | ?? |
| inc/hw_comp.h | ?? |
| inc/hw_des.h | ?? |
| inc/hw_eeprom.h | ?? |
| inc/hw_emac.h | ?? |
| inc/hw_epi.h | ?? |
| inc/hw_ethernet.h | ?? |
| inc/hw_fan.h | ?? |
| inc/hw_flash.h | ?? |
| inc/hw_gpio.h | ?? |
| inc/hw_hibernate.h | ?? |
| inc/hw_i2c.h | ?? |
| inc/hw_i2s.h | ?? |
| inc/hw_ints.h | ?? |
| inc/hw_lcd.h | ?? |
| inc/hw_lpc.h | ?? |
| inc/hw_memmap.h | ?? |
| inc/hw_nvic.h | ?? |
| inc/hw_peci.h | ?? |
| inc/hw_pwm.h | ?? |
| inc/hw_qei.h | ?? |
| inc/hw_shamd5.h | ?? |
| inc/hw_ssi.h | ?? |
| inc/hw_sysctl.h | ?? |
| inc/hw_sysexc.h | ?? |
| inc/hw_timer.h | ?? |
| inc/hw_types.h | ?? |
| inc/hw_uart.h | ?? |
| inc/hw_udma.h | ?? |
| inc/hw_usb.h | ?? |
| inc/hw_watchdog.h | ?? |
| inc/integer.h | ?? |
| inc/interpreter.h | 35 |
| inc/loader.h | ?? |
| inc/loader_config.h | ?? |
| inc/misc_macros.h | |
| Some helper macros | 37 |
| inc/OS.h | |
| Real Time Operating System for Labs 2 and 3 EE445M/EE380L.12 | 37 |
| inc/PLL.h | |
| Runs on LM4F120/TM4C123 A software priorityqueue to change the bus frequency using the PLL | 46 |
| inc/priorityqueue.h | ?? |
| inc/profiler.h | |
| Thread profiler utility | 49 |
| inc/ST7735.h | |
| Low level drivers for the ST7735 | 51 |
| inc/ST7735_lab3.h | |
| This is a library for the Adafruit 1.8" SPI display | 52 |
| inc/Switch.h | ?? |
| inc/timeMeasure.h | ?? |
| inc/tm4c123gh6pm.h | ?? |
| inc/UART.h | |
| 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 | 63 |

Chapter 5

Module Documentation

5.1 ELF Loader

Data Structures

- struct [ELFSymbol_t](#)
- struct [ELFEnv_t](#)

Enumerations

- enum [ELFSecPerm_t](#) { [ELF_SEC_WRITE](#) = 0x1, [ELF_SEC_READ](#) = 0x2, [ELF_SEC_EXEC](#) = 0x4 }

Functions

- int [exec_elf](#) (const char *path, const [ELFEnv_t](#) *env)

5.1.1 Detailed Description

5.1.2 Enumeration Type Documentation

5.1.2.1 enum ELFSecPerm_t

Protection flags of memory

Enumerator

ELF_SEC_WRITE Enable for write
ELF_SEC_READ Enable for read
ELF_SEC_EXEC Enable for execution (instruction fetch)

5.1.3 Function Documentation

5.1.3.1 int exec_elf (const char * path, const ELFEnv_t * env)

Execute ELF file from "path" with environment "env"

Parameters

| | |
|-------------|-------------------------------|
| <i>path</i> | Path to file to load |
| <i>env</i> | Pointer to environment struct |

Return values

| | |
|-----------|---------------|
| <i>0</i> | On successful |
| <i>-1</i> | On fail |

Todo Error information

Chapter 6

Data Structure Documentation

6.1 `_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:

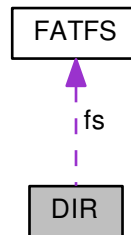
- inc/[OS.h](#)

6.2 DIR Struct Reference

Directory object structure ([DIR](#))

```
#include <ff.h>
```

Collaboration diagram for DIR:



Data Fields

- [FATFS](#) * **fs**
- WORD **id**
- WORD **index**
- DWORD **sclust**
- DWORD **clust**
- DWORD **sect**
- BYTE * **dir**
- BYTE * **fn**

6.2.1 Detailed Description

Directory object structure ([DIR](#))

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

- inc/[ff.h](#)

6.3 Elf32_Dyn Struct Reference

Data Fields

- Elf32_Sword **d_tag**
- union {
 - Elf32_Word **d_val**
 - Elf32_Addr **d_ptr**
 } **d_un**

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

- inc/[elf.h](#)

6.4 Elf32_Ehdr Struct Reference

Data Fields

- unsigned char **e_ident** [EI_NIDENT]
- Elf32_Half **e_type**
- Elf32_Half **e_machine**
- Elf32_Word **e_version**
- Elf32_Addr **e_entry**
- Elf32_Off **e_phoff**
- Elf32_Off **e_shoff**
- Elf32_Word **e_flags**
- Elf32_Half **e_ehsize**
- Elf32_Half **e_phentsize**
- Elf32_Half **e_phnum**
- Elf32_Half **e_shentsize**
- Elf32_Half **e_shnum**
- Elf32_Half **e_shstrndx**

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

- inc/elf.h

6.5 Elf32_Phdr Struct Reference

Data Fields

- Elf32_Word **p_type**
- Elf32_Off **p_offset**
- Elf32_Addr **p_vaddr**
- Elf32_Addr **p_paddr**
- Elf32_Word **p_filesz**
- Elf32_Word **p_memsz**
- Elf32_Word **p_flags**
- Elf32_Word **p_align**

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

- inc/elf.h

6.6 Elf32_Rel Struct Reference

Data Fields

- Elf32_Addr **r_offset**
- Elf32_Word **r_info**

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

- inc/elf.h

6.7 Elf32_Rela Struct Reference

Data Fields

- Elf32_Addr **r_offset**
- Elf32_Word **r_info**
- Elf32_Sword **r_addend**

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

- inc/elf.h

6.8 Elf32_Shdr Struct Reference

Data Fields

- Elf32_Word **sh_name**
- Elf32_Word **sh_type**
- Elf32_Word **sh_flags**
- Elf32_Addr **sh_addr**
- Elf32_Off **sh_offset**
- Elf32_Word **sh_size**
- Elf32_Word **sh_link**
- Elf32_Word **sh_info**
- Elf32_Word **sh_addralign**
- Elf32_Word **sh_entsize**

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

- inc/elf.h

6.9 Elf32_Sym Struct Reference

Data Fields

- Elf32_Word **st_name**
- Elf32_Addr **st_value**
- Elf32_Word **st_size**
- unsigned char **st_info**
- unsigned char **st_other**
- Elf32_Half **st_shndx**

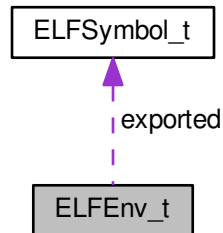
The documentation for this struct was generated from the following file:

- inc/elf.h

6.10 ELFEnv_t Struct Reference

```
#include <loader.h>
```

Collaboration diagram for ELFEnv_t:



Data Fields

- const [ELFSymbol_t](#) * `exported`
- unsigned int `exported_size`

6.10.1 Detailed Description

Environment for execution

6.10.2 Field Documentation

6.10.2.1 const [ELFSymbol_t](#)* `ELFEnv_t::exported`

Pointer to exported symbols array

6.10.2.2 unsigned int `ELFEnv_t::exported_size`

Elements on exported symbol array

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

- `inc/loader.h`

6.11 ELFSymbol_t Struct Reference

```
#include <loader.h>
```

Data Fields

- const char * [name](#)
- void * [ptr](#)

6.11.1 Detailed Description

Exported symbol struct

6.11.2 Field Documentation

6.11.2.1 const char* ELFSymbol_t::name

Name of symbol

6.11.2.2 void* ELFSymbol_t::ptr

Pointer of symbol in memory

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

- inc/loader.h

6.12 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](#)

6.13 FATFS Struct Reference

File system object structure ([FATFS](#))

```
#include <ff.h>
```

Data Fields

- BYTE **fs_type**
- BYTE **drv**
- BYTE **csize**
- BYTE **n_fats**
- BYTE **wflag**
- BYTE **fsi_flag**
- WORD **id**
- WORD **n_rootdir**
- DWORD **last_clust**
- DWORD **free_clust**
- DWORD **cdir**
- DWORD **n_fatent**
- DWORD **fsize**
- DWORD **volbase**
- DWORD **fatbase**
- DWORD **dirbase**
- DWORD **database**
- DWORD **winsect**
- BYTE **win** [_MAX_SS]

6.13.1 Detailed Description

File system object structure ([FATFS](#))

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

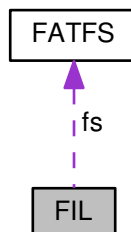
- [inc/ff.h](#)

6.14 FIL Struct Reference

File object structure ([FIL](#))

```
#include <ff.h>
```

Collaboration diagram for FIL:



Data Fields

- [FATFS](#) * **fs**
- WORD **id**
- BYTE **flag**
- BYTE **err**
- DWORD **fptr**
- DWORD **fsize**
- DWORD **sclust**
- DWORD **clust**
- DWORD **dsect**
- DWORD **dir_sect**
- BYTE * **dir_ptr**
- BYTE **buf** [_MAX_SS]

6.14.1 Detailed Description

File object structure ([FIL](#))

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

- inc/[ff.h](#)

6.15 FILINFO Struct Reference

File status structure ([FILINFO](#))

```
#include <ff.h>
```

Data Fields

- DWORD **fsize**
- WORD **fdate**
- WORD **ftime**
- BYTE **fattrib**
- TCHAR **fname** [13]

6.15.1 Detailed Description

File status structure ([FILINFO](#))

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

- inc/[ff.h](#)

6.16 heap_stats Struct Reference

Data Fields

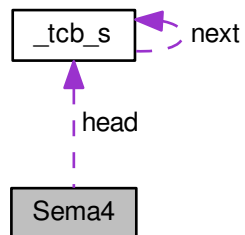
- int32_t **wordsAllocated**
- int32_t **wordsAvailable**
- int32_t **wordsOverhead**
- int32_t **blocksUsed**
- int32_t **blocksUnused**

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

- inc/[heap.h](#)

6.17 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 7

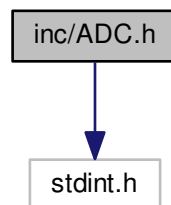
File Documentation

7.1 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.

7.1.1 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

7.1.2 Function Documentation

7.1.2.1 `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

| | |
|-------------------|---|
| <i>channelNum</i> | ADC channel to sample |
| <i>fs</i> | Sampling frequency |
| <i>handler</i> | Function which will be passed each sample as it is collected. |

Returns

int 0 on success, -1 on failure.

7.1.2.2 `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

7.1.2.3 `int ADC_Init (uint32_t channelNum)`

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

Parameters

| | |
|-------------------|-----------------------|
| <i>channelNum</i> | The channel to set up |
|-------------------|-----------------------|

Returns

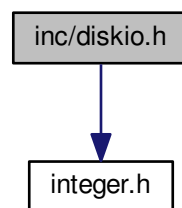
int 0 on success, -1 on failure.

7.2 inc/diskio.h File Reference

Low level disk interface module include file (C)ChaN, 2014 converted to TM4C123 Jonathan Valvano, January 13, 2015.

```
#include "integer.h"
```

Include dependency graph for diskio.h:



Macros

- `#define _USE_WRITE 1` /* 1: Enable [disk_write\(\)](#) function */
- `#define _USE_IOCTL 1` /* 1: Enable [disk_ioctl\(\)](#) function */
- `#define STA_NOINIT 0x01` /* Drive not initialized */
- `#define STA_NODISK 0x02` /* No medium in the drive */
- `#define STA_PROTECT 0x04` /* Write protected */
- `#define CTRL_SYNC 0` /* Complete pending write process (needed at _FS_READONLY == 0) */
- `#define GET_SECTOR_COUNT 1` /* Get media size (needed at _USE_MKFS == 1) */
- `#define GET_SECTOR_SIZE 2` /* Get sector size (needed at _MAX_SS != _MIN_SS) */
- `#define GET_BLOCK_SIZE 3` /* Get erase block size (needed at _USE_MKFS == 1) */
- `#define CTRL_TRIM 4` /* Inform device that the data on the block of sectors is no longer used (needed at _USE_TRIM == 1) */
- `#define CTRL_FORMAT 5` /* Create physical format on the media */
- `#define CTRL_POWER_IDLE 6` /* Put the device idle state */
- `#define CTRL_POWER_OFF 7` /* Put the device off state */
- `#define CTRL_LOCK 8` /* Lock media removal */
- `#define CTRL_UNLOCK 9` /* Unlock media removal */
- `#define CTRL_EJECT 10` /* Eject media */
- `#define MMC_GET_TYPE 50` /* Get card type */
- `#define MMC_GET_CSD 51` /* Get CSD */
- `#define MMC_GET_CID 52` /* Get CID */
- `#define MMC_GET_OCR 53` /* Get OCR */
- `#define MMC_GET_SDSTAT 54` /* Get SD status */
- `#define ATA_GET_REV 60` /* Get F/W revision */
- `#define ATA_GET_MODEL 61` /* Get model name */

- `#define ATA_GET_SN 62` /* Get serial number */
- `#define CT_MMC 0x01` /* MMC ver 3 */
- `#define CT_SD1 0x02` /* SD ver 1 */
- `#define CT_SD2 0x04` /* SD ver 2 */
- `#define CT_SDC (CT_SD1|CT_SD2)` /* SD */
- `#define CT_BLOCK 0x08` /* Block addressing */

Typedefs

- typedef BYTE **DSTATUS**
Status of Disk Functions.

Enumerations

- enum **DRESULT** {
 RES_OK = 0, **RES_ERROR**, **RES_WRPRT**, **RES_NOTRDY**,
 RES_PARERR }
Results of Disk Functions.

Functions

- **DSTATUS** **disk_initialize** (BYTE drv)
Initialize disk drive.
- **DSTATUS** **disk_status** (BYTE drv)
Get disk status.
- **DRESULT** **disk_read** (BYTE drv, BYTE *buff, DWORD sector, UINT count)
Read sector(s)
- **DRESULT** **disk_write** (BYTE drv, const BYTE *buff, DWORD sector, UINT count)
Write sector(s)
- **DRESULT** **disk_ioctl** (BYTE drv, BYTE cmd, void *buff)
Miscellaneous drive controls.

7.2.1 Detailed Description

Low level disk interface module include file (C)ChaN, 2014 converted to TM4C123 Jonathan Valvano, January 13, 2015.

7.2.2 Function Documentation

7.2.2.1 **DSTATUS** **disk_initialize** (BYTE drv)

Initialize disk drive.

Parameters

| | |
|------------|--|
| <i>drv</i> | Physical drive number, which must be 0 |
|------------|--|

Returns

status (see DSTATUS)

7.2.2.2 DRESULT disk_ioctl (BYTE *drv*, BYTE *cmd*, void * *buff*)

Miscellaneous drive controls.

Parameters

| | |
|-------------|-----------------------------|
| <i>drv</i> | Physical drive number (0) |
| <i>cmd</i> | Control command code |
| <i>buff</i> | Pointer to the control data |

Returns

status (see DRESULT)

7.2.2.3 DRESULT disk_read (BYTE *drv*, BYTE * *buff*, DWORD *sector*, UINT *count*)

Read sector(s)

Parameters

| | |
|---------------|---|
| <i>drv</i> | Physical drive number (0) |
| <i>buff</i> | Pointer to the data buffer to store read data |
| <i>sector</i> | Start sector number (LBA) |
| <i>count</i> | Number of sectors to read (1..128) |

Returns

status (see DRESULT)

7.2.2.4 DSTATUS disk_status (BYTE *drv*)

Get disk status.

Parameters

| | |
|------------|--|
| <i>drv</i> | Physical drive number, which must be 0 |
|------------|--|

Returns

status (see DSTATUS)

7.2.2.5 DRESULT disk_write (BYTE *drv*, const BYTE * *buff*, DWORD *sector*, UINT *count*)

Write sector(s)

Parameters

| | |
|---------------|---|
| <i>drv</i> | Physical drive number (0) |
| <i>buff</i> | Pointer to the data buffer to write to disk |
| <i>sector</i> | Start sector number (LBA) |
| <i>count</i> | Number of sectors to write (1..128) |

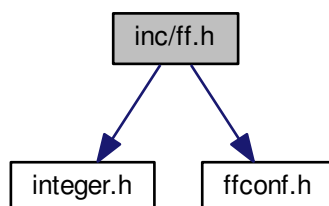
Returns

status (see DRESULT)

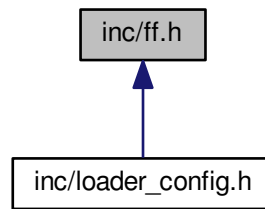
7.3 inc/ff.h File Reference

FatFs - FAT file system module include file R0.10c (C)ChaN, 2014 FatFs module is a generic FAT file system module for small embedded systems. This is a free software that opened for education, research and commercial developments under license policy of following terms. Copyright (C) 2014, , all right reserved. The FatFs module is a free software and there is NO WARRANTY. No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial product UNDER YOUR RESPONSIBILITY. Redistributions of source code must retain the above copyright notice.

```
#include "integer.h"
#include "ffconf.h"
Include dependency graph for ff.h:
```



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



Data Structures

- struct [FATFS](#)
File system object structure ([FATFS](#))
- struct [FIL](#)
File object structure ([FIL](#))
- struct [DIR](#)
Directory object structure ([DIR](#))
- struct [FILINFO](#)
File status structure ([FILINFO](#))

Macros

- `#define _FATFS 80376` /* Revision ID */
- `#define LD2PD(vol) (BYTE)(vol)` /* Each logical drive is bound to the same physical drive number */
- `#define LD2PT(vol) 0` /* Find first valid partition or in SFD */
- `#define _T(x) x`
- `#define _TEXT(x) x`
- `#define f_eof(fp) ((int)((fp)->fptr == (fp)->fsize))`
- `#define f_error(fp) ((fp)->err)`
- `#define f_tell(fp) ((fp)->fptr)`
- `#define f_size(fp) ((fp)->fsize)`
- `#define EOF (-1)`
- `#define FA_READ 0x01`
- `#define FA_OPEN_EXISTING 0x00`
- `#define FA_WRITE 0x02`
- `#define FA_CREATE_NEW 0x04`
- `#define FA_CREATE_ALWAYS 0x08`
- `#define FA_OPEN_ALWAYS 0x10`
- `#define FA__WRITTEN 0x20`
- `#define FA__DIRTY 0x40`
- `#define FS_FAT12 1`
- `#define FS_FAT16 2`
- `#define FS_FAT32 3`
- `#define AM_RDO 0x01` /* Read only */
- `#define AM_HID 0x02` /* Hidden */

- `#define AM_SYS 0x04 /* System */`
- `#define AM_VOL 0x08 /* Volume label */`
- `#define AM_LFN 0x0F /* LFN entry */`
- `#define AM_DIR 0x10 /* Directory */`
- `#define AM_ARC 0x20 /* Archive */`
- `#define AM_MASK 0x3F /* Mask of defined bits */`
- `#define CREATE_LINKMAP 0xFFFFFFFF`
- `#define LD_WORD(ptr) (WORD)(((WORD)*((BYTE*)(ptr)+1)<<8)|((WORD)*((BYTE*)(ptr))`
- `#define LD_DWORD(ptr) (DWORD)(((DWORD)*((BYTE*)(ptr)+3)<<24)|((DWORD)*((BYTE*)(ptr)+2)<<16)|((WORD)*((BYTE*)(ptr)+1)<<8)|((WORD)*((BYTE*)(ptr))`
- `#define ST_WORD(ptr, val) *((BYTE*)(ptr))=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8)`
- `#define ST_DWORD(ptr, val) *((BYTE*)(ptr))=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8); *((BYTE*)(ptr)+2)=(BYTE)((DWORD)(val)>>16); *((BYTE*)(ptr)+3)=(BYTE)((DWORD)(val)>>24)`

Typedefs

- `typedef char TCHAR`

Enumerations

- `enum FRESULT {`
`FR_OK = 0, FR_DISK_ERR, FR_INT_ERR, FR_NOT_READY,`
`FR_NO_FILE, FR_NO_PATH, FR_INVALID_NAME, FR_DENIED,`
`FR_EXIST, FR_INVALID_OBJECT, FR_WRITE_PROTECTED, FR_INVALID_DRIVE,`
`FR_NOT_ENABLED, FR_NO_FILESYSTEM, FR_MKFS_ABORTED, FR_TIMEOUT,`
`FR_LOCKED, FR_NOT_ENOUGH_CORE, FR_TOO_MANY_OPEN_FILES, FR_INVALID_PARAMETER }`

File function return code (FRESULT)

Functions

- `FRESULT f_open (FIL *fp, const TCHAR *path, BYTE mode)`
- `FRESULT f_close (FIL *fp)`
- `FRESULT f_read (FIL *fp, void *buff, UINT btr, UINT *br)`
- `FRESULT f_write (FIL *fp, const void *buff, UINT btw, UINT *bw)`
- `FRESULT f_forward (FIL *fp, UINT(*func)(const BYTE *, UINT), UINT btf, UINT *bf)`
- `FRESULT f_lseek (FIL *fp, DWORD ofs)`
- `FRESULT f_truncate (FIL *fp)`
- `FRESULT f_sync (FIL *fp)`
- `FRESULT f_opendir (DIR *dp, const TCHAR *path)`
- `FRESULT f_closedir (DIR *dp)`
- `FRESULT f_readdir (DIR *dp, FILINFO *fno)`
- `FRESULT f_mkdir (const TCHAR *path)`
- `FRESULT f_unlink (const TCHAR *path)`
- `FRESULT f_rename (const TCHAR *path_old, const TCHAR *path_new)`
- `FRESULT f_stat (const TCHAR *path, FILINFO *fno)`
- `FRESULT f_chmod (const TCHAR *path, BYTE value, BYTE mask)`
- `FRESULT f_utime (const TCHAR *path, const FILINFO *fno)`
- `FRESULT f_chdir (const TCHAR *path)`
- `FRESULT f_chdrive (const TCHAR *path)`
- `FRESULT f_getcwd (TCHAR *buff, UINT len)`
- `FRESULT f_getfree (const TCHAR *path, DWORD *nclst, FATFS **fatfs)`
- `FRESULT f_getlabel (const TCHAR *path, TCHAR *label, DWORD *vsn)`

- [FRESULT f_setlabel](#) (const TCHAR *label)
- [FRESULT f_mount](#) (FATFS *fs, const TCHAR *path, BYTE opt)
- [FRESULT f_mkfs](#) (const TCHAR *path, BYTE sdf, UINT au)
- [FRESULT f_fdisk](#) (BYTE pdrv, const DWORD szt[], void *work)
- int [f_putc](#) (TCHAR c, [FIL](#) *fp)
- int [f_puts](#) (const TCHAR *str, [FIL](#) *cp)
- int [f_printf](#) ([FIL](#) *fp, const TCHAR *str,...)
- TCHAR * [f_gets](#) (TCHAR *buff, int len, [FIL](#) *fp)

7.3.1 Detailed Description

FatFs - FAT file system module include file R0.10c (C)ChaN, 2014 FatFs module is a generic FAT file system module for small embedded systems. This is a free software that opened for education, research and commercial developments under license policy of following terms. Copyright (C) 2014, , all right reserved. The FatFs module is a free software and there is NO WARRANTY. No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial product UNDER YOUR RESPONSIBILITY. Redistributions of source code must retain the above copyright notice.

Author

ChaN

7.3.2 Function Documentation

7.3.2.1 FRESULT f_chdir (const TCHAR * *path*)

Change current directory

7.3.2.2 FRESULT f_chdrive (const TCHAR * *path*)

Change current drive

7.3.2.3 FRESULT f_chmod (const TCHAR * *path*, BYTE *value*, BYTE *mask*)

Change attribute of the file/dir

7.3.2.4 FRESULT f_close ([FIL](#) * *fp*)

Close an open file object

7.3.2.5 FRESULT f_closedir ([DIR](#) * *dp*)

Close an open directory

7.3.2.6 FRESULT f_disk (BYTE *pdrv*, const DWORD *szf[]*, void * *work*)

Divide a physical drive into some partitions

7.3.2.7 FRESULT f_forward (FIL * *fp*, UINT(*) (const BYTE *, UINT) *func*, UINT *btf*, UINT * *bf*)

Forward data to the stream

7.3.2.8 FRESULT f_getcwd (TCHAR * *buff*, UINT *len*)

Get current directory

7.3.2.9 FRESULT f_getfree (const TCHAR * *path*, DWORD * *nclst*, FATFS ** *fatfs*)

Get number of free clusters on the drive

7.3.2.10 FRESULT f_getlabel (const TCHAR * *path*, TCHAR * *label*, DWORD * *vsn*)

Get volume label

7.3.2.11 TCHAR* f_gets (TCHAR * *buff*, int *len*, FIL * *fp*)

Get a string from the file

7.3.2.12 FRESULT f_lseek (FIL * *fp*, DWORD *ofs*)

Move file pointer of a file object

7.3.2.13 FRESULT f_mkdir (const TCHAR * *path*)

Create a sub directory

7.3.2.14 FRESULT f_mkfs (const TCHAR * *path*, BYTE *sfd*, UINT *au*)

Create a file system on the volume

7.3.2.15 FRESULT f_mount (FATFS * *fs*, const TCHAR * *path*, BYTE *opt*)

Mount/Unmount a logical drive

7.3.2.16 **FRESULT** `f_open (FIL * fp, const TCHAR * path, BYTE mode)`

Open or create a file

7.3.2.17 **FRESULT** `f_opendir (DIR * dp, const TCHAR * path)`

Open a directory

7.3.2.18 `int` `f_printf (FIL * fp, const TCHAR * str, ...)`

Put a formatted string to the file

7.3.2.19 `int` `f_putc (TCHAR c, FIL * fp)`

Put a character to the file

7.3.2.20 `int` `f_puts (const TCHAR * str, FIL * fp)`

Put a string to the file

7.3.2.21 **FRESULT** `f_read (FIL * fp, void * buff, UINT btr, UINT * br)`

Read data from a file

7.3.2.22 **FRESULT** `f_readdir (DIR * dp, FILINFO * fno)`

Read a directory item

7.3.2.23 **FRESULT** `f_rename (const TCHAR * path_old, const TCHAR * path_new)`

Rename/Move a file or directory

7.3.2.24 **FRESULT** `f_setlabel (const TCHAR * label)`

Set volume label

7.3.2.25 **FRESULT** `f_stat (const TCHAR * path, FILINFO * fno)`

Get file status

7.3.2.26 FRESULT f_sync (FILE * fp)

Flush cached data of a writing file

7.3.2.27 FRESULT f_truncate (FILE * fp)

Truncate file

7.3.2.28 FRESULT f_unlink (const TCHAR * path)

Delete an existing file or directory

7.3.2.29 FRESULT f_ftime (const TCHAR * path, const FILINFO * fno)

Change times-tamp of the file/dir

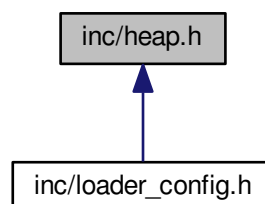
7.3.2.30 FRESULT f_write (FILE * fp, const void * buff, UINT btw, UINT * bw)

Write data to a file

7.4 inc/heap.h File Reference

Implements memory heap for dynamic memory allocation. Follows standard malloc/calloc/realloc/free interface for allocating/unallocating memory. modified 8/31/08 Jonathan Valvano for style modified 12/16/11 Jonathan Valvano for 32-bit machine modified August 10, 2014 for C99 syntax This example accompanies the book "Embedded Systems: Real Time Operating Systems for ARM Cortex M Microcontrollers", ISBN: 978-1466468863, Jonathan Valvano, copyright (c) 2014 Implementation Notes: This is a Knuth Heap. Each block has a header and a trailer, which I shall call the meta-sections. The meta-sections are each a single int32_t that tells how many int32_t's/words can be stored between the header and trailer. If the block is used, the meta-sections record the room as a positive number. If the block is unused, the meta-sections record the room as a negative number. Copyright 2014 by Jonathan W. Valvano, valvano@mail.utexas.edu You may use, edit, run or distribute this file as long as the above copyright notice remains THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER. For more information about my classes, my research, and my books, see <http://users.ece.utexas.edu/~valvano/>.

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



Data Structures

- struct [heap_stats](#)

Macros

- `#define HEAP_SIZE_BYTES (256)`
- `#define HEAP_SIZE_WORDS (HEAP_SIZE_BYTES / sizeof(int32_t))`
- `#define HEAP_OK 0`
- `#define HEAP_ERROR_CORRUPTED_HEAP 1`
- `#define HEAP_ERROR_POINTER_OUT_OF_RANGE 2`

Typedefs

- typedef struct [heap_stats](#) [heap_stats_t](#)

Functions

- `int32_t Heap_Init (void)`
Initialize the Heap notes: Initializes/resets the heap to a clean state where no memory is allocated.
- `void * Heap_Malloc (int32_t desiredBytes)`
Allocate memory, data not initialized.
- `void * Heap_Calloc (int32_t desiredBytes)`
Allocate memory, data are initialized to 0 notes: the allocated memory block will be zeroed out.
- `void * Heap_Realloc (void *oldBlock, int32_t desiredBytes)`
Reallocate buffer to a new size notes: the given block will be unallocated after its contents are copied to the new block.
- `int32_t Heap_Free (void *pointer)`
return a block to the heap
- `int32_t Heap_Test (void)`
Test the heap.
- `heap_stats_t Heap_Stats (void)`
return the current status of the heap

7.4.1 Detailed Description

Implements memory heap for dynamic memory allocation. Follows standard malloc/calloc/realloc/free interface for allocating/unallocating memory. modified 8/31/08 Jonathan Valvano for style modified 12/16/11 Jonathan Valvano for 32-bit machine modified August 10, 2014 for C99 syntax This example accompanies the book "Embedded Systems: Real Time Operating Systems for ARM Cortex M Microcontrollers", ISBN: 978-1466468863, Jonathan Valvano, copyright (c) 2014 Implementation Notes: This is a Knuth Heap. Each block has a header and a trailer, which I shall call the meta-sections. The meta-sections are each a single int32_t that tells how many int32_t's/words can be stored between the header and trailer. If the block is used, the meta-sections record the room as a positive number. If the block is unused, the meta-sections record the room as a negative number. Copyright 2014 by Jonathan W. Valvano, valvano@mail.utexas.edu You may use, edit, run or distribute this file as long as the above copyright notice remains THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER. For more information about my classes, my research, and my books, see <http://users.ece.utexas.edu/~valvano/>.

Author

Jacob Egnér

Date

2008-07-31

7.4.2 Function Documentation**7.4.2.1 void* Heap_Calloc (int32_t *desiredBytes*)**

Allocate memory, data are initialized to 0 notes: the allocated memory block will be zeroed out.

Parameters

| | |
|---------------------|-------------------------------------|
| <i>desiredBytes</i> | desired number of bytes to allocate |
|---------------------|-------------------------------------|

Returns

void* pointing to the allocated memory block or will return NULL if there isn't sufficient space to satisfy allocation request

7.4.2.2 int32_t Heap_Free (void * *pointer*)

return a block to the heap

Parameters

| | |
|----------------|-------------------------------------|
| <i>pointer</i> | the pointer to memory to unallocate |
|----------------|-------------------------------------|

Returns

HEAP_OK if everything is ok; HEAP_ERROR_POINTER_OUT_OF_RANGE if pointer points outside the heap; HEAP_ERROR_CORRUPTED_HEAP if heap has been corrupted or trying to unallocate memory that has already been unallocated;

7.4.2.3 int32_t Heap_Init (void)

Initialize the Heap notes: Initializes/resets the heap to a clean state where no memory is allocated.

Returns

always HEAP_OK

7.4.2.4 void* Heap_Malloc (int32_t *desiredBytes*)

Allocate memory, data not initialized.

Parameters

| | |
|---------------------|-------------------------------------|
| <i>desiredBytes</i> | desired number of bytes to allocate |
|---------------------|-------------------------------------|

Returns

void* pointing to the allocated memory or will return NULL if there isn't sufficient space to satisfy allocation request

7.4.2.5 void* Heap_Realloc (void * *oldBlock*, int32_t *desiredBytes*)

Reallocate buffer to a new size notes: the given block will be unallocated after its contents are copied to the new block.

Parameters

| | |
|---------------------|---|
| <i>oldBlock</i> | pointer to a block |
| <i>desiredBytes</i> | a desired number of bytes for a new block where the contents of the old block will be copied to |

Returns

void* pointing to the new block or will return NULL if there is any reason the reallocation can't be completed

7.4.2.6 heap_stats_t Heap_Stats (void)

return the current status of the heap

Returns

a heap_stats_t that describes the current usage of the heap

7.4.2.7 int32_t Heap_Test (void)

Test the heap.

Returns

validity of the heap - either HEAP_OK or HEAP_ERROR_HEAP_CORRUPTED

7.5 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.

7.5.1 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.
- `critical`
 - Prints the percentage of CPU time spent with interrupts disabled.
- `log`
 - Prints profiler events logged
- `clear`
 - Clears the profiler event log and restarts the profiler
- `format`
 - Formats the filesystem on the SD card
- `ls`
 - List all files in the filesystem
- `cat`
 - Print out file in the filesystem.
 - Takes one argument: the name of the file to print
- `rm`
 - Remove file in the filesystem.
 - Takes one argument: the name of the file to remove
- `touch`
 - Create a file in the filesystem.
 - Takes one argument: the name of the file to create
- `echo`
 - Append characters to a file
 - Takes two arguments in this order:
 - * The name of the file to append to
 - * Remaining characters are written to the file
- `increase`
 - Artificially increase the time spent in critical sections to test the "critical" command.

7.5.2 Function Documentation

7.5.2.1 `void interpreter_cmd (char * cmd_str)`

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

Parameters

| | |
|----------------------|--|
| <code>cmd_str</code> | String containing the entire user command. |
|----------------------|--|

7.6 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.

7.6.1 Detailed Description

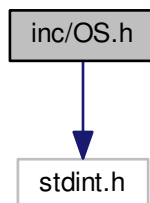
Some helper macros.

7.7 inc/OS.h File Reference

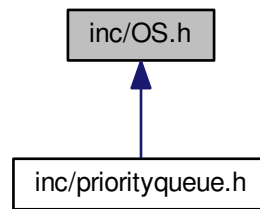
Real Time Operating System for Labs 2 and 3 EE445M/EE380L.12.

```
#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)
 - void [Jitter](#) (void)
- Print the max periodic task jitter measured thus far to the ST7735 display.*
- 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)
- int **OS_AddProcess** (void(*entry)(void), void *text, void *data, unsigned long stackSize, unsigned long priority)
- long **StartCritical** (void)
- void **EndCritical** (long sr)
- void **DisableInterrupts** (void)
- void **EnableInterrupts** (void)

7.7.1 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

7.7.2 Macro Definition Documentation

7.7.2.1 `#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

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

Returns

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

7.7.2.2 `#define OS_AddThread(task, stackSize, priority) OS_AddThread_priv(task, stackSize, priority, #task)`

add a foreground thread to the scheduler stack size must be divisible 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

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

Returns

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

7.7.3 Function Documentation

7.7.3.1 `int OS_AddSW1Task (void(*) (void) task, unsigned long priority)`

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

Parameters

| | |
|-----------------|------------------------------------|
| <i>pointer</i> | to a void/void background function |
| <i>priority</i> | 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_AddThread 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

7.7.3.2 `int OS_AddSW2Task (void(*) (void) task, unsigned long priority)`

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

Parameters

| | |
|-----------------|------------------------------------|
| <i>pointer</i> | to a void/void background function |
| <i>priority</i> | 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

7.7.3.3 void OS_bSignal (Sema4Type * *semaPt*)

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

Parameters

| | |
|---------------|-------------------------------|
| <i>semaPt</i> | pointer to a binary semaphore |
|---------------|-------------------------------|

7.7.3.4 void OS_bWait (Sema4Type * *semaPt*)

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

Parameters

| | |
|---------------|-------------------------------|
| <i>semaPt</i> | pointer to a binary semaphore |
|---------------|-------------------------------|

7.7.3.5 void OS_ClearMsTime (void)

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

Returns

none

7.7.3.6 unsigned long OS_Fifo_Get (void)

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

Returns

data

7.7.3.7 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

| | |
|-------------|------------------|
| <i>size</i> | Size of the fifo |
|-------------|------------------|

Returns

none

7.7.3.8 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

| | |
|-------------|-------------------------|
| <i>data</i> | Data to put in the FIFO |
|-------------|-------------------------|

Returns

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

7.7.3.9 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

7.7.3.10 unsigned long OS_Id (void)

returns the thread ID for the currently running thread

Returns

Thread ID, number greater than zero

7.7.3.11 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

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

initialize semaphore

Parameters

| | |
|----------------------------|------------------------|
| <i>sema</i> ↔ <i>Pt</i> | pointer to a semaphore |
|----------------------------|------------------------|

7.7.3.13 void OS_Kill (void)

kill the currently running thread, release its TCB and stack

7.7.3.14 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

| | |
|---------------------|---|
| <i>theTimeSlice</i> | number of 12.5ns clock cycles for each time slice |
|---------------------|---|

Returns

none (does not return)

7.7.3.15 void OS_MailBox_Init (void)

Initialize communication channel

Returns

none

7.7.3.16 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

7.7.3.17 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

| | |
|-------------|------------|
| <i>data</i> | to be sent |
|-------------|------------|

Returns

none

7.7.3.18 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

7.7.3.19 void OS_Signal (Sema4Type * *semaPt*)

increment semaphore Lab2 spinlock Lab3 wakeup blocked thread if appropriate

Parameters

| | |
|---------------|---------------------------------|
| <i>semaPt</i> | pointer to a counting semaphore |
|---------------|---------------------------------|

7.7.3.20 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

| | |
|------------------|-------------------------|
| <i>sleepTime</i> | number of msec to sleep |
|------------------|-------------------------|

7.7.3.21 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).

7.7.3.22 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

7.7.3.23 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

| | |
|--------------|----------------------------------|
| <i>start</i> | Start time measured with OS_Time |
| <i>stop</i> | Stop time measured with OS_Time |

Returns

time difference in 12.5ns units

7.7.3.24 void OS_Wait (Sema4Type * *semaPt*)

decrement semaphore Lab2 spinlock Lab3 block if less than zero

Parameters

| | |
|---------------|---------------------------------|
| <i>semaPt</i> | pointer to a counting semaphore |
|---------------|---------------------------------|

7.8 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

7.8.1 Detailed Description

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

Author

Daniel Valvano

7.8.2 Function Documentation

7.8.2.1 void PLL_Init (uint32_t freq)

configure the system to get its clock from the PLL

Parameters

| | |
|-------------|--|
| <i>freq</i> | Macro defined in PLL.h to choose frequency |
|-------------|--|

7.9 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(*)(const event_t *))
Executes a function f on each event in the log in the order they occurred in the system.

7.9.1 Detailed Description

Thread profiler utility.

Author

Riley Wood (riley.wood@utexas.edu)

7.9.2 Function Documentation

7.9.2.1 int Profiler_Event (event_type_e event_type, char * event_name)

Register an event has occurred in the profiler.

Parameters

| | |
|------------------|-------------------------------|
| <i>event↵_id</i> | ID of the event that occurred |
|------------------|-------------------------------|

Returns

-1 on error, 0 on success

7.9.2.2 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

| | |
|----------|---|
| <i>f</i> | Function to execute on each event in the log. |
|----------|---|

7.10 inc/ST7735.h File Reference

Low level drivers for the ST7735.

Macros

- `#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, INTR_GREENTAB, INTR_REDTAB, INTR_BLACKTAB,
 none, INTR_GREENTAB, INTR_REDTAB, INTR_BLACKTAB }`

Functions

- `void ST7735_InitB (void)`
- `void ST7735_InitR (enum initRFlags option)`
- `void ST7735_DrawPixel (short x, short y, unsigned short color)`
- `void ST7735_DrawFastVLine (short x, short y, short h, unsigned short color)`
- `void ST7735_DrawFastHLine (short x, short y, short w, unsigned short color)`
- `void ST7735_FillScreen (unsigned short color)`
- `void ST7735_FillRect (short x, short y, short w, short h, unsigned short color)`
- `unsigned short ST7735_Color565 (unsigned char r, unsigned char g, unsigned char b)`
- `unsigned short ST7735_SwapColor (unsigned short x)`
- `void ST7735_DrawBitmap (short x, short y, const unsigned short *image, short w, short h)`
- `void ST7735_DrawCharS (short x, short y, char c, short textColor, short bgColor, unsigned char size)`
- `void ST7735_DrawChar (short x, short y, char c, short textColor, short bgColor, unsigned char size)`
- `unsigned long ST7735_OutString (unsigned short x, unsigned short y, char *pt, short textColor)`
- `void ST7735_Message (unsigned long d, unsigned long l, char *pt, long value)`
- `void ST7735_SetRotation (unsigned char m)`
- `void ST7735_InvertDisplay (int i)`

7.10.1 Detailed Description

Low level drivers for the ST7735.

Runs on LM4F120/TM4C123. Low level drivers for the ST7735 160x128 LCD based off of the file described above. This version coexists with the SDC

Version

V1.0

Author

Valvano

Copyright

Copyright 2017 by Jonathan W. Valvano, valvano@mail.utexas.edu,

Warning

AS-IS

Note

For more information see <http://users.ece.utexas.edu/~valvano/>

Date

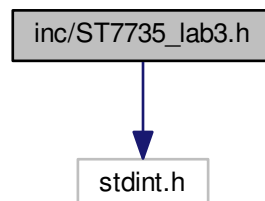
March 9, 2017

7.11 inc/ST7735_lab3.h File Reference

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

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

Include dependency graph for ST7735_lab3.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, **INTR_GREENTAB**, **INTR_REDTAB**, **INTR_BLACKTAB**,
none, **INTR_GREENTAB**, **INTR_REDTAB**, **INTR_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 $(11 + 2 \times \text{size} \times \text{size}) \times 6 \times 8$ (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 $(11 + \text{size} \times \text{size} \times 6 \times 8)$ 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 $(11 + \text{size} \times \text{size} \times 6 \times 8)$ 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_SetTextColor`.
- 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.

7.11.1 Detailed Description

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

7.11.2 Function Documentation

7.11.2.1 void `Output_Color` (uint32_t newColor)

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

Parameters

| | |
|-----------------|---------------------|
| <i>newColor</i> | 16-bit packed color |
|-----------------|---------------------|

7.11.2.2 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

| | |
|----------|-------------|
| <i>r</i> | red value |
| <i>g</i> | green value |
| <i>b</i> | blue value |

Returns

uint16_t 16-bit color

7.11.2.3 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

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

7.11.2.4 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

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

7.11.2.5 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

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

7.11.2.6 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

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

7.11.2.7 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

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

7.11.2.8 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

| | |
|--------------|--|
| <i>x</i> | 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 |
| <i>y</i> | 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 |
| <i>color</i> | 16-bit color, which can be produced by ST7735_Color565() |

7.11.2.9 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

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

Returns

uint32_t number of characters printed

7.11.2.10 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

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

7.11.2.11 void ST7735_FillScreen (uint16_t color)

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

Parameters

| | |
|--------------|--|
| <i>color</i> | 16-bit color, which can be produced by ST7735_Color565() |
|--------------|--|

7.11.2.12 void ST7735_InitR (enum initRFlags option)

Initialization for ST7735R screens (green or red tabs).

Parameters

| | |
|-------------------|---|
| <i>initRFlags</i> | one of the enumerated options depending on tabs |
|-------------------|---|

7.11.2.13 void ST7735_InvertDisplay (int i)

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

Parameters

| | |
|----------|--|
| <i>i</i> | 0 to disable inversion; non-zero to enable inversion |
|----------|--|

7.11.2.14 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

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

7.11.2.15 void ST7735_OutChar (char ch)

Output one character to the LCD Position determined by ST7735_SetCursor command Color set by ST7735_SetTextColor.

Parameters

| | |
|-----------|-----------------------|
| <i>ch</i> | 8-bit ASCII character |
|-----------|-----------------------|

7.11.2.16 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

| | |
|------------|---|
| <i>ptr</i> | pointer to NULL-terminated ASCII string |
|------------|---|

7.11.2.17 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

| | |
|----------|---------------------------------|
| <i>n</i> | 32-bit number to be transferred |
|----------|---------------------------------|

7.11.2.18 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

| | |
|----------|-------------------------------------|
| <i>y</i> | the y coordinate of the bar plotted |
|----------|-------------------------------------|

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

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

Parameters

| | |
|-------------|---------------------|
| <i>ymin</i> | Lower bound of plot |
| <i>ymax</i> | Upper bound of plot |

7.11.2.20 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

| | |
|----------|------------------------------------|
| <i>y</i> | the y ADC value of the bar plotted |
|----------|------------------------------------|

7.11.2.21 void ST7735_PlotLine (int32_t *y*)

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

Parameters

| | |
|----------|---------------------------------------|
| <i>y</i> | the y coordinate of the point plotted |
|----------|---------------------------------------|

7.11.2.22 void ST7735_PlotPoint (int32_t *y*)

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

Parameters

| | |
|----------|---------------------------------------|
| <i>y</i> | the y coordinate of the point plotted |
|----------|---------------------------------------|

7.11.2.23 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

| | |
|-----------|--|
| <i>y1</i> | the y coordinate of the first point plotted |
| <i>y2</i> | the y coordinate of the second point plotted |

7.11.2.24 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

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

7.11.2.25 void ST7735_SetRotation (uint8_t *m*)

Change the image rotation. Requires 2 bytes of transmission.

Parameters

| | |
|----------|-----------------------------|
| <i>m</i> | new rotation value (0 to 3) |
|----------|-----------------------------|

7.11.2.26 void ST7735_SetTextColor (uint16_t *color*)

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

Parameters

| | |
|--------------|---------------------|
| <i>color</i> | 16-bit packed color |
|--------------|---------------------|

7.11.2.27 uint16_t ST7735_SwapColor (uint16_t *x*)

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

Parameters

| | |
|----------|--------------------------------|
| <i>x</i> | 16-bit color in format B, G, R |
|----------|--------------------------------|

Returns

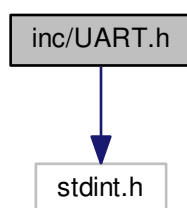
uint16_t 16-bit color in format R, G, B

7.12 inc/UART.h File Reference

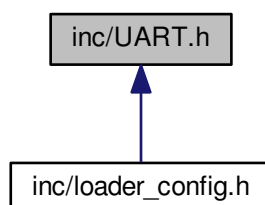
Runs on LM4F120/TM4C123 Use UART0 to implement bidirectional data transfer to and from a computer running HyperTerminal. This time, interrupts and FIFOs are used.

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

Include dependency graph for UART.h:



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



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.

7.12.1 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

7.12.2 Function Documentation

7.12.2.1 char UART_InChar (void)

Wait for new serial port input.

Returns

char ASCII code for key typed

7.12.2.2 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

| | |
|---------------------------|-------------------------|
| <i>buf</i> ↔ <i>Pt</i> | pointer to empty buffer |
| <i>max</i> | size of buffer |

7.12.2.3 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

7.12.2.4 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

7.12.2.5 void UART_OutChar (char *data*)

8-bit to serial port

Parameters

| | |
|-------------|--|
| <i>data</i> | letter is an 8-bit ASCII character to be transferred |
|-------------|--|

7.12.2.6 void UART_OutString (char * *pt*)

Output String (NULL termination)

Parameters

| | |
|-----------|---|
| <i>pt</i> | pointer to a NULL-terminated string to be transferred |
|-----------|---|

7.12.2.7 void UART_OutUDec (uint32_t *n*)

Output a 32-bit number in unsigned decimal format.

Parameters

| | |
|----------|---------------------------------|
| <i>n</i> | 32-bit number to be transferred |
|----------|---------------------------------|

7.12.2.8 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

| | |
|---------------|---------------------------------|
| <i>number</i> | 32-bit number to be transferred |
|---------------|---------------------------------|

Index

- [_tcb_s](#), [11](#)
- ADC.h
 - [ADC_Collect](#), [22](#)
 - [ADC_In](#), [22](#)
 - [ADC_Init](#), [22](#)
- [ADC_Collect](#)
 - [ADC.h](#), [22](#)
- [ADC_In](#)
 - [ADC.h](#), [22](#)
- [ADC_Init](#)
 - [ADC.h](#), [22](#)
- [DIR](#), [12](#)
- [disk_initialize](#)
 - [diskio.h](#), [24](#)
- [disk_ioctl](#)
 - [diskio.h](#), [25](#)
- [disk_read](#)
 - [diskio.h](#), [25](#)
- [disk_status](#)
 - [diskio.h](#), [25](#)
- [disk_write](#)
 - [diskio.h](#), [25](#)
- [diskio.h](#)
 - [disk_initialize](#), [24](#)
 - [disk_ioctl](#), [25](#)
 - [disk_read](#), [25](#)
 - [disk_status](#), [25](#)
 - [disk_write](#), [25](#)
- [ELF Loader](#), [9](#)
 - [ELF_SEC_EXEC](#), [9](#)
 - [ELF_SEC_READ](#), [9](#)
 - [ELF_SEC_WRITE](#), [9](#)
 - [ELFSecPerm_t](#), [9](#)
 - [exec_elf](#), [9](#)
- [ELF_SEC_EXEC](#)
 - [ELF Loader](#), [9](#)
- [ELF_SEC_READ](#)
 - [ELF Loader](#), [9](#)
- [ELF_SEC_WRITE](#)
 - [ELF Loader](#), [9](#)
- [ELFEnv_t](#), [15](#)
 - [exported](#), [15](#)
 - [exported_size](#), [15](#)
- [ELFSecPerm_t](#)
 - [ELF Loader](#), [9](#)
- [ELFSymbol_t](#), [15](#)
 - [name](#), [16](#)
 - [ptr](#), [16](#)
- [Elf32_Dyn](#), [12](#)
- [Elf32_Ehdr](#), [13](#)
- [Elf32_Phdr](#), [13](#)
- [Elf32_Rel](#), [13](#)
- [Elf32_Rela](#), [14](#)
- [Elf32_Shdr](#), [14](#)
- [Elf32_Sym](#), [14](#)
- [event_t](#), [16](#)
- [exec_elf](#)
 - [ELF Loader](#), [9](#)
- [exported](#)
 - [ELFEnv_t](#), [15](#)
- [exported_size](#)
 - [ELFEnv_t](#), [15](#)
- [f_chdir](#)
 - [ff.h](#), [29](#)
- [f_chdrive](#)
 - [ff.h](#), [29](#)
- [f_chmod](#)
 - [ff.h](#), [29](#)
- [f_close](#)
 - [ff.h](#), [29](#)
- [f_closedir](#)
 - [ff.h](#), [29](#)
- [f_fdisk](#)
 - [ff.h](#), [29](#)
- [f_forward](#)
 - [ff.h](#), [30](#)
- [f_getcwd](#)
 - [ff.h](#), [30](#)
- [f_getfree](#)
 - [ff.h](#), [30](#)
- [f_getlabel](#)
 - [ff.h](#), [30](#)
- [f_gets](#)
 - [ff.h](#), [30](#)
- [f_lseek](#)
 - [ff.h](#), [30](#)
- [f_mkdir](#)
 - [ff.h](#), [30](#)
- [f_mkfs](#)
 - [ff.h](#), [30](#)
- [f_mount](#)
 - [ff.h](#), [30](#)
- [f_open](#)
 - [ff.h](#), [30](#)
- [f_opendir](#)
 - [ff.h](#), [31](#)

- f_printf
 - ff.h, [31](#)
- f_putc
 - ff.h, [31](#)
- f_puts
 - ff.h, [31](#)
- f_read
 - ff.h, [31](#)
- f_readdir
 - ff.h, [31](#)
- f_rename
 - ff.h, [31](#)
- f_setlabel
 - ff.h, [31](#)
- f_stat
 - ff.h, [31](#)
- f_sync
 - ff.h, [31](#)
- f_truncate
 - ff.h, [32](#)
- f_unlink
 - ff.h, [32](#)
- f_utime
 - ff.h, [32](#)
- f_write
 - ff.h, [32](#)
- FATFS, [16](#)
- FILINFO, [18](#)
- FIL, [17](#)
- ff.h
 - f_chdir, [29](#)
 - f_chdrive, [29](#)
 - f_chmod, [29](#)
 - f_close, [29](#)
 - f_closedir, [29](#)
 - f_fdisk, [29](#)
 - f_forward, [30](#)
 - f_getcwd, [30](#)
 - f_getfree, [30](#)
 - f_getlabel, [30](#)
 - f_gets, [30](#)
 - f_lseek, [30](#)
 - f_mkdir, [30](#)
 - f_mkfs, [30](#)
 - f_mount, [30](#)
 - f_open, [30](#)
 - f_opendir, [31](#)
 - f_printf, [31](#)
 - f_putc, [31](#)
 - f_puts, [31](#)
 - f_read, [31](#)
 - f_readdir, [31](#)
 - f_rename, [31](#)
 - f_setlabel, [31](#)
 - f_stat, [31](#)
 - f_sync, [31](#)
 - f_truncate, [32](#)
 - f_unlink, [32](#)
 - f_utime, [32](#)
 - f_write, [32](#)
- heap.h
 - Heap_Calloc, [34](#)
 - Heap_Free, [34](#)
 - Heap_Init, [34](#)
 - Heap_Malloc, [34](#)
 - Heap_Realloc, [35](#)
 - Heap_Stats, [35](#)
 - Heap_Test, [35](#)
- Heap_Calloc
 - heap.h, [34](#)
- Heap_Free
 - heap.h, [34](#)
- Heap_Init
 - heap.h, [34](#)
- Heap_Malloc
 - heap.h, [34](#)
- Heap_Realloc
 - heap.h, [35](#)
- Heap_Stats
 - heap.h, [35](#)
- Heap_Test
 - heap.h, [35](#)
- heap_stats, [19](#)
- inc/ADC.h, [21](#)
- inc/OS.h, [37](#)
- inc/PLL.h, [46](#)
- inc/ST7735.h, [51](#)
- inc/ST7735_lab3.h, [52](#)
- inc/UART.h, [63](#)
- inc/diskio.h, [23](#)
- inc/ff.h, [26](#)
- inc/heap.h, [32](#)
- inc/interpreter.h, [35](#)
- inc/misc_macros.h, [37](#)
- inc/profiler.h, [49](#)
- interpreter.h
 - interpreter_cmd, [36](#)
- interpreter_cmd
 - interpreter.h, [36](#)
- name
 - ELFSymbol_t, [16](#)
- OS.h
 - OS_AddPeriodicThread, [39](#)
 - OS_AddSW1Task, [40](#)
 - OS_AddSW2Task, [40](#)
 - OS_AddThread, [40](#)
 - OS_ClearMsTime, [41](#)
 - OS_Fifo_Get, [41](#)
 - OS_Fifo_Init, [41](#)
 - OS_Fifo_Put, [42](#)
 - OS_Fifo_Size, [42](#)
 - OS_Id, [42](#)
 - OS_Init, [42](#)

- OS_InitSemaphore, [43](#)
- OS_Kill, [44](#)
- OS_Launch, [44](#)
- OS_MailBox_Init, [44](#)
- OS_MailBox_Recv, [44](#)
- OS_MailBox_Send, [44](#)
- OS_MsTime, [45](#)
- OS_Signal, [45](#)
- OS_Sleep, [45](#)
- OS_Suspend, [45](#)
- OS_Time, [45](#)
- OS_TimeDifference, [46](#)
- OS_Wait, [46](#)
- OS_bSignal, [41](#)
- OS_bWait, [41](#)
- OS_AddPeriodicThread
 - OS.h, [39](#)
- OS_AddSW1Task
 - OS.h, [40](#)
- OS_AddSW2Task
 - OS.h, [40](#)
- OS_AddThread
 - OS.h, [40](#)
- OS_ClearMsTime
 - OS.h, [41](#)
- OS_Fifo_Get
 - OS.h, [41](#)
- OS_Fifo_Init
 - OS.h, [41](#)
- OS_Fifo_Put
 - OS.h, [42](#)
- OS_Fifo_Size
 - OS.h, [42](#)
- OS_Id
 - OS.h, [42](#)
- OS_Init
 - OS.h, [42](#)
- OS_InitSemaphore
 - OS.h, [43](#)
- OS_Kill
 - OS.h, [44](#)
- OS_Launch
 - OS.h, [44](#)
- OS_MailBox_Init
 - OS.h, [44](#)
- OS_MailBox_Recv
 - OS.h, [44](#)
- OS_MailBox_Send
 - OS.h, [44](#)
- OS_MsTime
 - OS.h, [45](#)
- OS_Signal
 - OS.h, [45](#)
- OS_Sleep
 - OS.h, [45](#)
- OS_Suspend
 - OS.h, [45](#)
- OS_Time
 - OS.h, [45](#)
- OS_TimeDifference
 - OS.h, [46](#)
- OS_Wait
 - OS.h, [46](#)
- OS_bSignal
 - OS.h, [41](#)
- OS_bWait
 - OS.h, [41](#)
- Output_Color
 - ST7735_lab3.h, [55](#)
- PLL.h
 - PLL_Init, [49](#)
- PLL_Init
 - PLL.h, [49](#)
- profiler.h
 - Profiler_Event, [50](#)
 - Profiler_Foreach, [50](#)
- Profiler_Event
 - profiler.h, [50](#)
- Profiler_Foreach
 - profiler.h, [50](#)
- ptr
 - ELFSymbol_t, [16](#)
- ST7735_Color565
 - ST7735_lab3.h, [55](#)
- ST7735_DrawBitmap
 - ST7735_lab3.h, [55](#)
- ST7735_DrawChar
 - ST7735_lab3.h, [57](#)
- ST7735_DrawCharS
 - ST7735_lab3.h, [57](#)
- ST7735_DrawFastHLine
 - ST7735_lab3.h, [57](#)
- ST7735_DrawFastVLine
 - ST7735_lab3.h, [58](#)
- ST7735_DrawPixel
 - ST7735_lab3.h, [58](#)
- ST7735_DrawString
 - ST7735_lab3.h, [58](#)
- ST7735_FillRect
 - ST7735_lab3.h, [59](#)
- ST7735_FillScreen
 - ST7735_lab3.h, [59](#)
- ST7735_InitR
 - ST7735_lab3.h, [59](#)
- ST7735_InvertDisplay
 - ST7735_lab3.h, [59](#)
- ST7735_Message
 - ST7735_lab3.h, [59](#)
- ST7735_OutChar
 - ST7735_lab3.h, [60](#)
- ST7735_OutString
 - ST7735_lab3.h, [60](#)
- ST7735_OutUDec
 - ST7735_lab3.h, [60](#)
- ST7735_PlotBar

- ST7735_lab3.h, [60](#)
- ST7735_PlotClear
 - ST7735_lab3.h, [61](#)
- ST7735_PlotLine
 - ST7735_lab3.h, [61](#)
- ST7735_PlotPoint
 - ST7735_lab3.h, [61](#)
- ST7735_PlotPoints
 - ST7735_lab3.h, [61](#)
- ST7735_PlotdBfs
 - ST7735_lab3.h, [61](#)
- ST7735_SetCursor
 - ST7735_lab3.h, [62](#)
- ST7735_SetRotation
 - ST7735_lab3.h, [62](#)
- ST7735_SetTextColor
 - ST7735_lab3.h, [62](#)
- ST7735_SwapColor
 - ST7735_lab3.h, [62](#)
- ST7735_lab3.h
 - Output_Color, [55](#)
 - ST7735_Color565, [55](#)
 - ST7735_DrawBitmap, [55](#)
 - ST7735_DrawChar, [57](#)
 - ST7735_DrawCharS, [57](#)
 - ST7735_DrawFastHLine, [57](#)
 - ST7735_DrawFastVLine, [58](#)
 - ST7735_DrawPixel, [58](#)
 - ST7735_DrawString, [58](#)
 - ST7735_FillRect, [59](#)
 - ST7735_FillScreen, [59](#)
 - ST7735_InitR, [59](#)
 - ST7735_InvertDisplay, [59](#)
 - ST7735_Message, [59](#)
 - ST7735_OutChar, [60](#)
 - ST7735_OutString, [60](#)
 - ST7735_OutUDec, [60](#)
 - ST7735_PlotBar, [60](#)
 - ST7735_PlotClear, [61](#)
 - ST7735_PlotLine, [61](#)
 - ST7735_PlotPoint, [61](#)
 - ST7735_PlotPoints, [61](#)
 - ST7735_PlotdBfs, [61](#)
 - ST7735_SetCursor, [62](#)
 - ST7735_SetRotation, [62](#)
 - ST7735_SetTextColor, [62](#)
 - ST7735_SwapColor, [62](#)
- Sema4, [19](#)
- UART.h
 - UART_InChar, [64](#)
 - UART_InString, [64](#)
 - UART_InUDec, [65](#)
 - UART_InUHex, [65](#)
 - UART_OutChar, [65](#)
 - UART_OutString, [65](#)
 - UART_OutUDec, [65](#)
 - UART_OutUHex, [66](#)
- UART_InChar