EE445M/EE380L Lab 7 Documentation

Generated by Doxygen 1.8.11

Contents

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:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

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

path	Path to file to load
env	Pointer to environment struct

Return values

0	On successful
-1	On fail

Todo Error information

5.2 Can_api

Data Structures

- struct tCANMsgObject
- struct tCANBitClkParms

Macros

- #define MSG_OBJ_TX_INT_ENABLE 0x00000001
- #define MSG_OBJ_RX_INT_ENABLE 0x00000002
- #define MSG_OBJ_EXTENDED_ID 0x00000004
- #define MSG_OBJ_USE_ID_FILTER 0x00000008
- #define MSG_OBJ_NEW_DATA 0x00000080

  This indicates that new data was available in the message object.
- #define MSG_OBJ_DATA_LOST 0x00000100
- #define MSG_OBJ_USE_DIR_FILTER (0x00000010 | MSG_OBJ_USE_ID_FILTER)
- #define MSG_OBJ_USE_EXT_FILTER (0x00000020 | MSG_OBJ_USE_ID_FILTER)
- #define MSG_OBJ_REMOTE_FRAME 0x00000040

  This indicates that a message object is a remote frame.
- #define MSG_OBJ_FIFO 0x00000200
- #define MSG_OBJ_NO_FLAGS 0x00000000

  This indicates that a message object has no flags set.
- #define MSG_OBJ_STATUS_MASK (MSG_OBJ_NEW_DATA | MSG_OBJ_DATA_LOST)
- #define CAN_INT_ERROR 0x00000008
- #define CAN_INT_STATUS 0x00000004
- #define CAN_INT_MASTER 0x00000002
- #define CAN_STATUS_BUS_OFF 0x00000080

  CAN controller has entered a Bus Off state.
- #define CAN_STATUS_EWARN 0x00000040

  CAN controller error level has reached warning level.
- #define CAN_STATUS_EPASS 0x00000020

  CAN controller error level has reached error passive level.
- #define CAN_STATUS_RXOK 0x00000010

  A message was received successfully since the last read of this status.
- #define CAN_STATUS_TXOK 0x00000008
- #define CAN_STATUS_LEC_MSK 0x00000007

  This is the mask for the last error code field.
- #define CAN_STATUS_LEC_NONE 0x00000000

  There was no error.
- #define CAN_STATUS_LEC_STUFF 0x00000001

  A bit stuffing error has occurred.
- #define CAN_STATUS_LEC_FORM 0x00000002

  A formatting error has occurred.
- #define CAN_STATUS_LEC_ACK 0x00000003

  An acknowledge error has occurred.
- #define CAN_STATUS_LEC_BIT1 0x00000004

  The bus remained a bit level of 1 for longer than is allowed.
- #define CAN_STATUS_LEC_BIT0 0x00000005

  The bus remained a bit level of 0 for longer than is allowed.
- #define CAN_STATUS_LEC_CRC 0x00000006

  A CRC error has occurred.
- #define CAN_STATUS_LEC_MASK 0x00000007

  This is the mask for the CAN Last Error Code (LEC).
- #define **CANSetBitTiming**(a, b) CANBitTimingSet(a, b)
- #define **CANGetBitTiming**(a, b) CANBitTimingGet(a, b)

Enumerations

- enum tCANIntStsReg { CAN_INT_STS_CAUSE, CAN_INT_STS_OBJECT }
- enum tCANStsReg { CAN_STS_CONTROL, CAN_STS_TXREQUEST, CAN_STS_NEWDAT, CAN_STS_↩ MSGVAL }
- enum tMsgObjType { MSG_OBJ_TYPE_TX, MSG_OBJ_TYPE_TX_REMOTE, MSG_OBJ_TYPE_RX, MSG_OBJ_TYPE_RX_↩ REMOTE, MSG_OBJ_TYPE_RXTX_REMOTE }

Functions

- void **CANBitTimingGet** (uint32_t ulBase, tCANBitClkParms *pClkParms)
- void **CANBitTimingSet** (uint32_t ulBase, tCANBitClkParms *pClkParms)
- uint32_t **CANBitRateSet** (uint32_t ulBase, uint32_t ulSourceClock, uint32_t ulBitRate)
- void **CANDisable** (uint32_t ulBase)
- void **CANEnable** (uint32_t ulBase)
- tBoolean **CANErrCntrGet** (uint32_t ulBase, uint32_t *pulRxCount, uint32_t *pulTxCount)
- void **CANInit** (uint32_t ulBase)
- void **CANIntClear** (uint32_t ulBase, uint32_t ulIntClr)
- void **CANIntDisable** (uint32_t ulBase, uint32_t ulIntFlags)
- void **CANIntEnable** (uint32_t ulBase, uint32_t ulIntFlags)
- void **CANIntRegister** (uint32_t ulBase, void(*pfnHandler)(void))
- uint32_t **CANIntStatus** (uint32_t ulBase, tCANIntStsReg eIntStsReg)
- void **CANIntUnregister** (uint32_t ulBase)
- void **CANMessageClear** (uint32_t ulBase, uint32_t ulObjID)
- void **CANMessageGet** (uint32_t ulBase, uint32_t ulObjID, tCANMsgObject *pMsgObject, tBoolean bClr↩ PendingInt)
- void **CANMessageSet** (uint32_t ulBase, uint32_t ulObjID, tCANMsgObject *pMsgObject, tMsgObjType e↩ MsgType)
- tBoolean **CANRetryGet** (uint32_t ulBase)
- void **CANRetrySet** (uint32_t ulBase, tBoolean bAutoRetry)
- uint32_t **CANStatusGet** (uint32_t ulBase, tCANStsReg eStatusReg)

5.2.1 Detailed Description

5.2.2 Macro Definition Documentation

5.2.2.1 #define CAN_INT_ERROR 0x00000008

This flag is used to allow a CAN controller to generate error interrupts.

5.2.2.2 #define CAN_INT_MASTER 0x00000002

This flag is used to allow a CAN controller to generate any CAN interrupts. If this is not set, then no interrupts will be generated by the CAN controller.

5.2.2.3 #define CAN_INT_STATUS 0x00000004

This flag is used to allow a CAN controller to generate status interrupts.

5.2.2.4 #define CAN_STATUS_TXOK 0x00000008

A message was transmitted successfully since the last read of this status.

5.2.2.5 #define MSG_OBJ_DATA_LOST 0x00000100

This indicates that data was lost since this message object was last read.

5.2.2.6 #define MSG_OBJ_EXTENDED_ID 0x00000004

This indicates that a message object will use or is using an extended identifier.

5.2.2.7 #define MSG_OBJ_FIFO 0x00000200

This indicates that this message object is part of a FIFO structure and not the final message object in a FIFO.

5.2.2.8 #define MSG_OBJ_RX_INT_ENABLE 0x00000002

This indicates that receive interrupts should be enabled, or are enabled.

5.2.2.9 #define MSG_OBJ_STATUS_MASK (MSG_OBJ_NEW_DATA | MSG_OBJ_DATA_LOST)

This define is used with the flag values to allow checking only status flags and not configuration flags.

5.2.2.10 #define MSG_OBJ_TX_INT_ENABLE 0x00000001

This definition is used with the tCANMsgObject ulFlags value and indicates that transmit interrupts should be enabled, or are enabled.

5.2.2.11 #define MSG_OBJ_USE_DIR_FILTER (0x00000010 | MSG_OBJ_USE_ID_FILTER)

This indicates that a message object will use or is using filtering based on the direction of the transfer. If the direction filtering is used, then ID filtering must also be enabled.

5.2.2.12 #define MSG_OBJ_USE_EXT_FILTER (0x00000020 | MSG_OBJ_USE_ID_FILTER)

This indicates that a message object will use or is using message identifier filtering based on the extended identifier. If the extended identifier filtering is used, then ID filtering must also be enabled.

5.2.2.13 #define MSG_OBJ_USE_ID_FILTER 0x00000008

This indicates that a message object will use or is using filtering based on the object's message identifier.

5.2.3 Enumeration Type Documentation

5.2.3.1 enum tCANIntStsReg

This data type is used to identify the interrupt status register. This is used when calling the CANIntStatus() function.

Enumerator

CAN_INT_STS_CAUSE Read the CAN interrupt status information.

CAN_INT_STS_OBJECT Read a message object's interrupt status.

5.2.3.2 enum tCANStsReg

This data type is used to identify which of several status registers to read when calling the CANStatusGet() function.

Enumerator

CAN_STS_CONTROL Read the full CAN controller status.

CAN_STS_TXREQUEST Read the full 32-bit mask of message objects with a transmit request set.

CAN_STS_NEWDAT Read the full 32-bit mask of message objects with new data available.

CAN_STS_MSGVAL Read the full 32-bit mask of message objects that are enabled.

5.2.3.3 enum tMsgObjType

This definition is used to determine the type of message object that will be set up via a call to the CANMessageSet() API.

Enumerator

MSG_OBJ_TYPE_TX Transmit message object.

MSG_OBJ_TYPE_TX_REMOTE Transmit remote request message object.

MSG_OBJ_TYPE_RX Receive message object.

MSG_OBJ_TYPE_RX_REMOTE Receive remote request message object.

MSG_OBJ_TYPE_RXTX_REMOTE Remote frame receive remote, with auto-transmit message object.

Chapter 6

Data Structure Documentation

6.1 _pcb_s Struct Reference

Data Fields

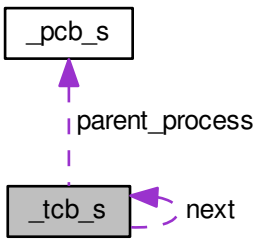
- unsigned long **num_threads**
- void * **text**
- void * **data**

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

- inc/OS.h

6.2 _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**
- pcb_t * **parent_process**

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

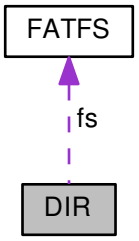
- inc/OS.h

6.3 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**

6.3.1 Detailed Description

Directory object structure (DIR)

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

- inc/ff.h

6.4 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

6.5 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

6.6 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

6.7 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

6.8 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

6.9 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

6.10 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**

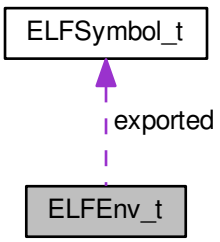
The documentation for this struct was generated from the following file:

- inc/elf.h

6.11 ELFEnv_t Struct Reference

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

Collaboration diagram for ELFEnv_t:



Data Fields

- const ELFSymbol_t * exported
- unsigned int exported_size

6.11.1 Detailed Description

Environment for execution

6.11.2 Field Documentation

6.11.2.1 const ELFSymbol_t* ELFEnv_t::exported

Pointer to exported symbols array

6.11.2.2 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

6.12 ELFSymbol_t Struct Reference

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

Data Fields

- const char * name
- void * ptr

6.12.1 Detailed Description

Exported symbol struct

6.12.2 Field Documentation

6.12.2.1 const char* ELFSymbol_t::name

Name of symbol

6.12.2.2 void* ELFSymbol_t::ptr

Pointer of symbol in memory

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

- inc/loader.h

6.13 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

6.14 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]

6.14.1 Detailed Description

File system object structure (FATFS)

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

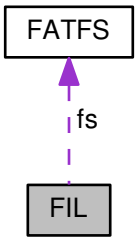
- inc/ff.h

6.15 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]

6.15.1 Detailed Description

File object structure (FIL)

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

- inc/ff.h

6.16 FILINFO Struct Reference

File status structure (FILINFO)

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

Data Fields

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

6.16.1 Detailed Description

File status structure (FILINFO)

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

- inc/ff.h

6.17 heap_stats Struct Reference

Data Fields

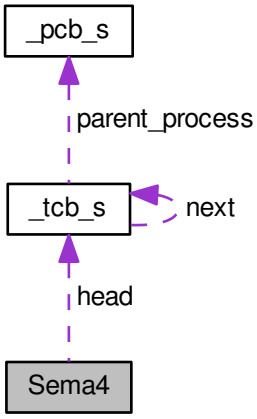
- int32_t **wordsAllocated**
- int32_t **wordsAvailable**
- int32_t **wordsOverhead**
- int32_t **blocksUsed**
- int32_t **blocksUnused**

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

- inc/heap.h

6.18 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

6.19 tCANBitClkParms Struct Reference

`#include <can.h>`

Data Fields

- unsigned int uSyncPropPhase1Seg
- unsigned int uPhase2Seg
- unsigned int uSJW
- unsigned int uQuantumPrescaler

6.19.1 Detailed Description

This structure is used for encapsulating the values associated with setting up the bit timing for a CAN controller. The structure is used when calling the CANGetBitTiming and CANSetBitTiming functions.

6.19.2 Field Documentation

6.19.2.1 unsigned int tCANBitClkParms::uPhase2Seg

This value holds the Phase Buffer 2 segment in time quanta. The valid values for this setting range from 1 to 8.

6.19.2.2 unsigned int tCANBitClkParms::uQuantumPrescaler

This value holds the CAN_CLK divider used to determine time quanta. The valid values for this setting range from 1 to 1023.

6.19.2.3 unsigned int tCANBitClkParms::uSJW

This value holds the Resynchronization Jump Width in time quanta. The valid values for this setting range from 1 to 4.

6.19.2.4 unsigned int tCANBitClkParms::uSyncPropPhase1Seg

This value holds the sum of the Synchronization, Propagation, and Phase Buffer 1 segments, measured in time quanta. The valid values for this setting range from 2 to 16.

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

- inc/can.h

6.20 tCANMsgObject Struct Reference

`#include <can.h>`

Data Fields

- uint32_t ulMsgID
  The CAN message identifier used for 11 or 29 bit identifiers.
- uint32_t ulMsgIDMask
  The message identifier mask used when identifier filtering is enabled.
- uint32_t ulFlags
- uint32_t ulMsgLen
  This value is the number of bytes of data in the message object.
- uint8_t * pucMsgData
  This is a pointer to the message object's data.

6.20.1 Detailed Description

The structure used for encapsulating all the items associated with a CAN message object in the CAN controller.

6.20.2 Field Documentation

6.20.2.1 uint32_t tCANMsgObject::ulFlags

This value holds various status flags and settings specified by tCANObjFlags.

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

- inc/can.h

Chapter 7

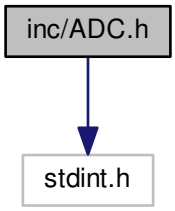
File Documentation

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.
- int ADC_Collect4Chan (uint32_t channels[ ], uint32_t fs, void(∗handler)(unsigned long))

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

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

Returns

int 0 on success, -1 on failure.

7.1.2.2 int ADC_Collect4Chan (uint32_t *channels[ ],* uint32_t *fs,* void(*)(unsigned long) *handler*)

channels: 4 element array with each element indiciating one channel to open

7.1.2.3 uint16_t ADC_In (void)

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

This function uses busy-wait for the ADC sampling to be done

Returns

uint16_t The conversion result

7.1.2.4 int ADC_Init (uint32_t *channelNum*)

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

Parameters

channelNum	The channel to set up

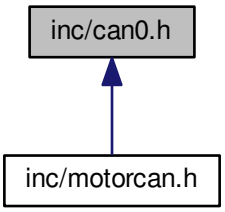
Returns

int 0 on success, -1 on failure.

7.2 inc/can0.h File Reference

Use CAN0 to communicate on CAN bus Conventions:

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



Macros

- #define **CAN_BITRATE** 1000000
- #define **RCV_ID** 2
- #define **XMT_ID** 4

Functions

- void CAN0_GetMailwithIdx (uint8_t data[4], int rxidx)

  If receive data is ready, gets the data.
- void CAN0_Open (void)

  Initialize CAN0 port.
- int CAN0_SetRecv (int idx)

  Set handler work for specific recieve ID.
- void CAN0_SendDatawithIdx (uint8_t data[4], int idx)

  Send 4 bytes of data to other microcontroller.

7.2.1 Detailed Description

Use CAN0 to communicate on CAN bus Conventions:

Author

Jonathan Valvano and modified by Jeageun Jung

- How to use : When Initialize OS, CAN0_Open() one time. Every ID that want to use as an receive port, CAN0_SetRecv(receive idx) to make interrupt for specific index. CAN0_GetMailwithIdx(data, index) in the point that we want to get data. It automatically block code until data arrive using blocking semaphore. CA↩ N0_SendDatawithIdx(data, index) when we have data to send.
- Pin configurations: CAN0Rx PE4 (8) I TTL CAN module 0 receive. CAN0Tx PE5 (8) O TTL CAN module 0 transmit.

  Copyright

  Copyright (c) 2019

7.2.2 Function Documentation

7.2.2.1 void CAN0_GetMailwithIdx (uint8_t *data[4],* int *rxidx*)

If receive data is ready, gets the data.

Parameters

data	Pointer to get data
rxidx	Recieve ID which is previously opened

7.2.2.2 void CAN0_SendDatawithIdx (uint8_t *data[4],* int *idx*)

Send 4 bytes of data to other microcontroller.

Parameters

data	Pointer to send data
idx	ID to send the data

7.2.2.3 int CAN0_SetRecv (int *idx*)

Set handler work for specific recieve ID.

Parameters

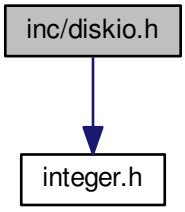
idx	ID to use for recieving data

7.3 inc/diskio.h File Reference

Low level disk interface modlue 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() fucntion */
- #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.

7.3.1 Detailed Description

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

7.3.2 Function Documentation

7.3.2.1 DSTATUS disk_initialize (BYTE *drv*)

Initialize disk drive.

Parameters

drv	Physical drive number, which must be 0

Returns

status (see DSTATUS)

7.3.2.2 DRESULT disk_ioctl (BYTE *drv,* BYTE *cmd,* void ∗ *buff*)

Miscellaneous drive controls.

Parameters

drv	Physical drive number (0)
cmd	Control command code
buff	Pointer to the control data

Returns

status (see DRESULT)

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

Read sector(s)

Parameters

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

Returns

status (see DRESULT)

7.3.2.4 DSTATUS disk_status (BYTE *drv*)

Get disk status.

Parameters

drv	Physical drive number, which must be 0

Returns

status (see DSTATUS)

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

Write sector(s)

Parameters

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

Returns

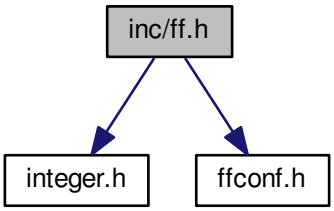
status (see DRESULT)

7.4 inc/ff.h File Reference

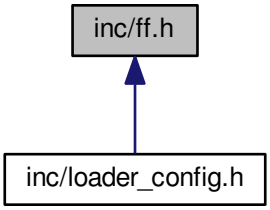
FatFs - FAT file system module include file R0.10c (C)ChaN, 2014 FatFs module is a generic FAT file system module for small embedded systems. This is a free software that opened for education, research and commercial developments under license policy of following terms. Copyright (C) 2014, , all right 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)|((W↩ ORD)*((BYTE*)(ptr)+1)<<8)|*(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 sfd, 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)

7.4.1 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

7.4.2 Function Documentation

7.4.2.1 FRESULT f_chdir (const TCHAR * path)

Change current directory

7.4.2.2 FRESULT f_chdrive (const TCHAR * path)

Change current drive

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

Change attribute of the file/dir

7.4.2.4 FRESULT f_close (FIL * fp)

Close an open file object

7.4.2.5 FRESULT f_closedir (DIR * dp)

Close an open directory

7.4.2.6 FRESULT f_fdisk (BYTE pdrv, const DWORD szt[ ], void * work)

Divide a physical drive into some partitions

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

Forward data to the stream

7.4.2.8 FRESULT f_getcwd (TCHAR * buff, UINT len)

Get current directory

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

Get number of free clusters on the drive

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

Get volume label

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

Get a string from the file

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

Move file pointer of a file object

7.4.2.13 FRESULT f_mkdir (const TCHAR * *path*)

Create a sub directory

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

Create a file system on the volume

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

Mount/Unmount a logical drive

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

Open or create a file

7.4.2.17 FRESULT f_opendir (DIR * *dp,* const TCHAR * *path*)

Open a directory

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

Put a formatted string to the file

7.4.2.19 int f_putc (TCHAR *c,* FIL * *fp*)

Put a character to the file

7.4.2.20 int f_puts (const TCHAR * *str,* FIL * *cp*)

Put a string to the file

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

Read data from a file

7.4.2.22 FRESULT f_readdir (DIR * *dp,* FILINFO * *fno*)

Read a directory item

7.4.2.23 FRESULT f_rename (const TCHAR * *path_old,* const TCHAR * *path_new*)

Rename/Move a file or directory

7.4.2.24 FRESULT f_setlabel (const TCHAR * *label*)

Set volume label

7.4.2.25 FRESULT f_stat (const TCHAR * *path,* FILINFO * *fno*)

Get file status

7.4.2.26 FRESULT f_sync (FIL * *fp*)

Flush cached data of a writing file

7.4.2.27 FRESULT f_truncate (FIL * *fp*)

Truncate file

7.4.2.28 FRESULT f_unlink (const TCHAR * *path*)

Delete an existing file or directory

7.4.2.29 FRESULT f_utime (const TCHAR * *path,* const FILINFO * *fno*)

Change times-tamp of the file/dir

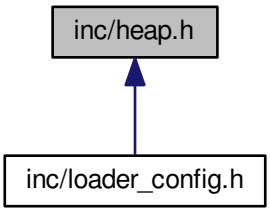
7.4.2.30 FRESULT f_write (FIL * *fp,* const void * *buff,* UINT *btw,* UINT * *bw*)

Write data to a file

7.5 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_↩ 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, W↩ HETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. VALV↩ ANO 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** (2048)
- #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

7.5.1 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_↩ 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, W↩ HETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. VALV↩ ANO 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 Egner

Date

2008-07-31

7.5.2 Function Documentation

7.5.2.1 void* Heap_Calloc (int32_t *desiredBytes*)

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

Parameters

desiredBytes	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

7.5.2.2 int32_t Heap_Free (void * *pointer*)

return a block to the heap

Parameters

pointer	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;

7.5.2.3 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

7.5.2.4 void* Heap_Malloc (int32_t *desiredBytes*)

Allocate memory, data not initialized.

Parameters

desiredBytes	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

7.5.2.5 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

oldBlock	pointer to a block
desiredBytes	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

7.5.2.6 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

7.5.2.7 int32_t Heap_Test (void)

Test the heap.

Returns

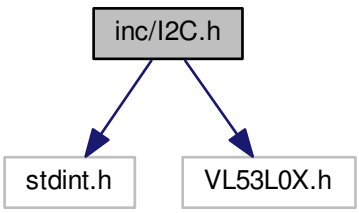
validity of the heap - either HEAP_OK or HEAP_ERROR_HEAP_CORRUPTED

7.6 inc/I2C.h File Reference

TM4C123G I2C APIs and some settings. For future development and updates, please follow this repository: If you find any bug or problem, please create new issue or a pull request with a fix in the repository. Or you can simply email me about the problem or bug at zeelivermorium@gmail.com Much Appreciated!

```
#include <stdint.h>
#include "VL53L0X.h"
```

Include dependency graph for I2C.h:



Macros

- #define **I2C_BUS1** I2C0
- #define **I2C_BUS2** I2C3
- #define **I2C0** 0xabcc
- #define **I2C1** 0xabcd
- #define **I2C2** 0xabce
- #define **I2C3** 0xabcf

Functions

- void I2C_Init (void)
  initialize a I2C module with corresponding setting parameters.
- int I2C_read (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, uint32_t count, int deviceIndex)
  read 1 or more bytes from slave device.
- int I2C_write (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, uint32_t count, int deviceIndex)
  write 1 or more bytes to slave device.
- int I2C_read_byte (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, int deviceIndex)
  read 1 byte from slave device.
- int I2C_write_byte (uint8_t deviceAddress, uint8_t targetRegister, uint8_t data, int deviceIndex)
  write 1 byte to slave device.
- int I2C_read_2_bytes (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, int deviceIndex)
  read 2 bytes from slave device.
- int I2C_write_2_bytes (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, int deviceIndex)
  write 2 bytes to slave device.
- int I2C_read_4_bytes (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, int deviceIndex)
  read 4 bytes from slave device.
- int I2C_write_4_bytes (uint8_t deviceAddress, uint8_t targetRegister, uint8_t *data, int deviceIndex)
  write 4 bytes to slave device.

7.6.1 Detailed Description

TM4C123G I2C APIs and some settings. For future development and updates, please follow this repository: If you find any bug or problem, please create new issue or a pull request with a fix in the repository. Or you can simply email me about the problem or bug at zeelivermorium@gmail.com Much Appreciated!

Author

Zee Livermorium

Date

Aug 4, 2018

7.6.2 Function Documentation

7.6.2.1 void I2C_Init (void)

initialize a I2C module with corresponding setting parameters.

I2C_Init

7.6.2.2 int I2C_read (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* uint32_t *count,* int *deviceIndex*)

read 1 or more bytes from slave device.

I2C_read

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device.
data	data address to store read data. **count number of bytes to be read.**

7.6.2.3 int I2C_read_2_bytes (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* int *deviceIndex*)

read 2 bytes from slave device.

I2C_read_2_bytes

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device. **data data address to store read data.**

7.6.2.4 int I2C_read_4_bytes (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* int *deviceIndex*)

read 4 bytes from slave device.

I2C_read_4_bytes

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device. **data data address to store read data.**

7.6.2.5 int I2C_read_byte (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* int *deviceIndex*)

read 1 byte from slave device.

I2C_read_byte

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device. **data data address to store read data.**

7.6.2.6 int I2C_write (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* uint32_t *count,* int *deviceIndex*)

write 1 or more bytes to slave device.

I2C_write

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device.
data	data address of data to be writen. **count number of bytes to be writen.**

7.6.2.7 int I2C_write_2_bytes (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* int *deviceIndex*)

write 2 bytes to slave device.

I2C_write_2_bytes

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device. **data data to be writen.**

7.6.2.8 int I2C_write_4_bytes (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t * *data,* int *deviceIndex*)

write 4 bytes to slave device.

I2C_write_4_bytes

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device. **data data to be writen.**

7.6.2.9 int I2C_write_byte (uint8_t *deviceAddress,* uint8_t *targetRegister,* uint8_t *data,* int *deviceIndex*)

write 1 byte to slave device.

I2C_write_byte

Parameters

deviceAddress	address of slave device.
targetRegister	target register of slave device. **data data to be writen.**

7.7 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.

7.7.1 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.

7.7.2 Function Documentation

7.7.2.1 void interpreter_cmd (char * *cmd_str*)

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

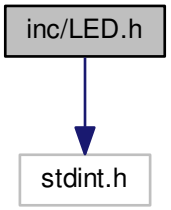
Parameters

cmd_str	String containing the entire user command.

7.8 inc/LED.h File Reference

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

Include dependency graph for LED.h:



Functions

- void **LED_Init** (void)
- void **LED_RED_ON** (void)
- void **LED_RED_OFF** (void)
- void **LED_RED_TOGGLE** (void)
- void **LED_BLUE_ON** (void)
- void **LED_BLUE_OFF** (void)
- void **LED_BLUE_TOGGLE** (void)
- void **LED_GREEN_ON** (void)
- void **LED_GREEN_OFF** (void)
- void **LED_GREEN_TOGGLE** (void)

7.8.1 Detailed Description

Adapted code from ValvanoWareTM4C123 by Dr. Jonathan Valvano You can find ValvanoWareTM4C123 at http://edx-org-utaustinx.s3.amazonaws.com/UT601x/ValvanoWareTM4C123.zip?dl=1

You can find more of his work at http://users.ece.utexas.edu/~valvano/

Author

Zee Livermorium

Date

Apr 12, 2018

7.9 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.

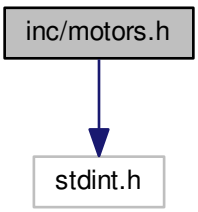
7.9.1 Detailed Description

Some helper macros.

7.10 inc/motors.h File Reference

Interface to two DC motors controlled by PWM. Allows differential driving.

`#include <stdint.h>`
Include dependency graph for motors.h:



Functions

- void Motors_Init (void)
  Initialize the robot motors.
- void Motors_SetTorque (int16_t left_trq, int16_t right_trq)
  Set the torque for each of the two motors atomically.
- void Motors_SetTorque_Left (int16_t left_trq)
  Set the torque of the left motor individually.
- void Motors_SetTorque_Right (int16_t right_trq)
  Set the torque of the left motor individually.
- void Motors_Brake (void)
  Brake both motors (tie both motors' leads to ground)
- void Motors_Brake_Left (void)
  Brake left motor (tie motor leads to ground)
- void Motors_Brake_Right (void)
  Brake right motor (tie motor leads to ground)

7.10.1 Detailed Description

Interface to two DC motors controlled by PWM. Allows differential driving.

Author

Riley Wood (riley.wood@utexas.edu)

This library makes use of PWM module 0, outputs 0 and 3. These are produced by PWM module 0's generators numbered 0 and 1.

Conventions:

- Definition of "left" versus "right":
  - The "left motor" is the motor on your left when the robot is on the ground with the servo pointed away from you.
  - The "right motor" is the motor on your right when the robot is on the ground with the servo pointed away from you.
- How to connect motors to motor board:
  - The left motor must be connected to motor port A.
  - The right motor must be connected to motor port B.
- Motor pin assignments:
  - The red wire of the left motor will be connected to A- and is controlled by PB6
  - The black wire of the left motor will be connected to A+ and is controlled by PB7
  - The red wire of right motor will be connected to B- and is controlled by PB4
  - The black wire of the right motor will be connected to B+ and is controlled by PB5
- Pin configurations:
  - A- (PB6) and B+ (PB5) will be configured as PWM outputs
  - A+ (PB7) and B- (PB4) will be configured as digital outputs.
  - We alternate + and - so that when both motors are driving forward (i.e. they are rotating OPPOSITE directions) with the same torque, their digital and PWM output configurations will be identical.
- H-Bridge convention:
  - A value of 1 (high) on any of PB4/5/6/7 will connect the corresponding motor terminal (A/B/+/-) to battery power.
  - A value of 0 (low) on any of PB4/5/6/7 will connect the corresponding motor terminal (A/B/+/-) to ground.

Copyright

7.10.2 Function Documentation

7.10.2.1 void Motors_SetTorque (int16_t *left_trq*, int16_t *right_trq*)

Set the torque for each of the two motors atomically.

Parameters

left_trq	Torque for left motor. Between -50 and 50. Positive argument indicates forward motion of robot, negative indicates backward. Zero indicates no rotation.
right_trq	Torque for right motor. Between -50 and 50. Positive argument indicates forward motion of robot, negative indicates backward. Zero indicates no rotation.

7.10.2.2 void Motors_SetTorque_Left (int16_t *left_trq*)

Set the torque of the left motor individually.

Parameters

left_trq	Torque for left motor. Between -50 and 50. Positive argument indicates forward motion of robot, negative indicates backward. Zero indicates no rotation.

7.10.2.3 void Motors_SetTorque_Right (int16_t *right_trq*)

Set the torque of the left motor individually.

Parameters

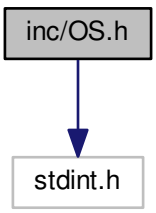
right_trq	Torque for right motor. Between -50 and 50. Positive argument indicates forward motion of robot, negative indicates backward. Zero indicates no rotation.

7.11 inc/OS.h File Reference

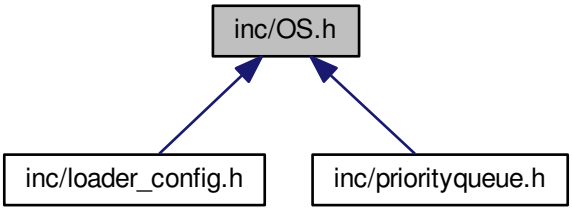
Real Time Operating System for Labs 2 and 3 EE445M/EE380L.12.

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

Include dependency graph for OS.h:



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



Data Structures

- struct _pcb_s
- 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)
- #define OS_AddPeriodicThread(task, period, priority) OS_AddPeriodicThread_priv(task, period, priority, #task)

Typedefs

- typedef struct _pcb_s **pcb_t**
- 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** (void(*task)(void), unsigned long stackSize, unsigned long priority, char *task_name, pcb_t *parent_process)
- 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)

  Launch a process in the OS.
- long **StartCritical** (void)
- void **EndCritical** (long sr)
- void **DisableInterrupts** (void)
- void **EnableInterrupts** (void)

Variables

- tcb_t * **cur_tcb**

7.11.1 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

7.11.2 Macro Definition Documentation

7.11.2.1 #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

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

Returns

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

7.11.2.2 #define OS_AddThread(*task, stackSize, priority*)

Value:

```
__OS_AddThread(task,\
                                                    stackSize,\
                                                    priority,\
                                                    #task,\
                                                    cur_tcb ? cur_tcb->parent_process : 0)
```

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

Parameters

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

Returns

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

7.11.3 Function Documentation

7.11.3.1 int OS_AddProcess (void(∗)(void) *entry,* void ∗ *text,* void ∗ *data,* unsigned long *stackSize,* unsigned long *priority*)

Launch a process in the OS.

Parameters

entry	Entry point, usually main() of the process
text	Text (code) section start address
data	Data section start address
stackSize	Size of the stack for the first thread
priority	Priority for the first thread

Returns

int 0 on success, -1 on failure.

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

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

Parameters

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

Returns

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

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

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

Parameters

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

Returns

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

7.11.3.4 void OS_bSignal (Sema4Type * *semaPt*)

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

Parameters

sema↩ Pt	pointer to a binary semaphore

7.11.3.5 void OS_bWait (Sema4Type * *semaPt*)

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

Parameters

sema↩ Pt	pointer to a binary semaphore

7.11.3.6 void OS_ClearMsTime (void)

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

Returns

none

7.11.3.7 unsigned long OS_Fifo_Get (void)

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

Returns

data

7.11.3.8 void OS_Fifo_Init (unsigned long *size*)

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

Parameters

size	Size of the fifo

Returns

none

7.11.3.9 int OS_Fifo_Put (unsigned long *data*)

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

Parameters

data	Data to put in the FIFO

Returns

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

7.11.3.10 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

7.11.3.11 unsigned long OS_Id (void)

returns the thread ID for the currently running thread

Returns

Thread ID, number greater than zero

7.11.3.12 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

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

initialize semaphore

Parameters

sema↩ Pt	pointer to a semaphore

7.11.3.14 void OS_Kill (void)

kill the currently running thread, release its TCB and stack

7.11.3.15 void OS_Launch (unsigned long *theTimeSlice*)

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

Parameters

theTimeSlice	number of 12.5ns clock cycles for each time slice

Returns

none (does not return)

7.11.3.16 void OS_MailBox_Init (void)

Initialize communication channel

Returns

none

7.11.3.17 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

7.11.3.18 void OS_MailBox_Send (unsigned long *data*)

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

Parameters

data	to be sent

Returns

none

7.11.3.19 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

7.11.3.20 void OS_Signal (Sema4Type * *semaPt*)

increment semaphore Lab2 spinlock Lab3 wakeup blocked thread if appropriate

Parameters

semaPt	pointer to a counting semaphore

7.11.3.21 void OS_Sleep (unsigned long *sleepTime*)

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

Parameters

sleepTime	number of msec to sleep

7.11.3.22 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).

7.11.3.23 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

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

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

Parameters

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

Returns

time difference in 12.5ns units

7.11.3.25 void OS_Wait (Sema4Type * *semaPt*)

decrement semaphore Lab2 spinlock Lab3 block if less than zero

Parameters

sema↩ Pt	pointer to a counting semaphore

7.12 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

7.12.1 Detailed Description

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

Author

Daniel Valvano

7.12.2 Function Documentation

7.12.2.1 void PLL_Init (uint32_t *freq*)

configure the system to get its clock from the PLL

Parameters

freq	Macro defined in PLL.h to choose frequency

7.13 inc/profiler.h File Reference

Thread profiler utility.

Data Structures

- struct event_t

Macros

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

Enumerations

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

Functions

- void Profiler_Init (void)

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

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

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

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

7.13.1 Detailed Description

Thread profiler utility.

Author

Riley Wood (riley.wood@utexas.edu)

7.13.2 Function Documentation

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

Register an event has occurred in the profiler.

Parameters

event_id	ID of the event that occurred

Returns

-1 on error, 0 on success

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

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

Parameters

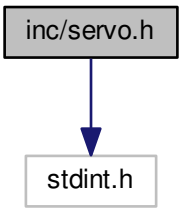
f	Function to execute on each event in the log.

7.14 inc/servo.h File Reference

Implements a driver for a single hitec HS-300 servo motor. PWM Module 1 is allocated to this driver.

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

Include dependency graph for servo.h:



Functions

- void Servo_Init (void)

  Initialize the servo module.
- void Servo_SetAngle (uint16_t degrees)

  Command the servo to hold a specific angular position.
- void Servo_Off (void)

  Stop driving the servo, allowing it to move freely. Call Servo_SetAngle to activate the servo again.

7.14.1 Detailed Description

Implements a driver for a single hitec HS-300 servo motor. PWM Module 1 is allocated to this driver.

Author

Riley Wood (riley.wood@utexas.edu)

Copyright

Copyright (c) 2019

7.14.2 Function Documentation

7.14.2.1 void Servo_SetAngle (uint16_t *degrees*)

Command the servo to hold a specific angular position.

Parameters

degrees	Desired position in degrees (0-180)

7.15 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, **INITR_GREENTAB**, **INITR_REDTAB**, **INITR_BLACKTAB**,
  none, **INITR_GREENTAB**, **INITR_REDTAB**, **INITR_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)

7.15.1 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

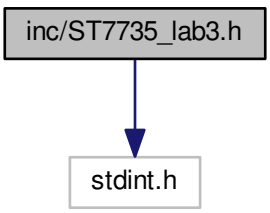
March 9, 2017

7.16 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, **INITR_GREENTAB**, **INITR_REDTAB**, **INITR_BLACKTAB**,
  none, **INITR_GREENTAB**, **INITR_REDTAB**, **INITR_BLACKTAB** }

  some flags for ST7735_InitR()

Functions

- void ST7735_InitB (void)

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

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

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

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

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

  Fill the screen with the given color. Requires 40,971 bytes of transmission.
- void ST7735_FillRect (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  Used in all the plots to step the X coordinate one pixel X steps from 0 to 127, then back to 0 again It does not output to display.
- void ST7735_PlotNextErase (void)

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

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

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

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

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

  Clear display.
- void Output_Off (void)

Turn off display (low power)

- void Output_On (void)

  Turn on display.

- void Output_Color (uint32_t newColor)

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

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

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

7.16.1 Detailed Description

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

7.16.2 Function Documentation

7.16.2.1 void Output_Color (uint32_t *newColor*)

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

Parameters

newColor	16-bit packed color

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

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

Parameters

r	red value
g	green value
b	blue value

Returns

uint16_t 16-bit color

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

Displays a 16-bit color BMP image. A bitmap file that is created by a PC image processing program has a header and may be padded with dummy columns so the data have four byte alignment. This function assumes that all of that has been stripped out, and the array image[] has one 16-bit halfword for each pixel to be displayed on the screen (encoded in reverse order, which is standard for bitmap files). An array can be created in this format from a

24-bit-per-pixel .bmp file using the associated converter program. (x,y) is the screen location of the lower left corner of BMP image Requires (11 + 2∗w∗h) bytes of transmission (assuming image fully on screen) Must be less than or equal to 128 pixels wide by 160 pixels high.

Parameters

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

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

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

Parameters

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

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

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

Parameters

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

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

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

Parameters

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

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

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

Parameters

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

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

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

Parameters

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

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

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

Parameters

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

Returns

uint32_t number of characters printed

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

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

Parameters

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

7.16.2.11 void ST7735_FillScreen (uint16_t *color*)

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

Parameters

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

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

Initialization for ST7735R screens (green or red tabs).

Parameters

initRFlags	one of the enumerated options depending on tabs

7.16.2.13 void ST7735_InvertDisplay (int *i*)

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

Parameters

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

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

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

Parameters

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

7.16.2.15 void ST7735_OutChar (char *ch*)

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

Parameters

ch	8-bit ASCII character

7.16.2.16 void ST7735_OutString (char * *ptr*)

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

Parameters

ptr	pointer to NULL-terminated ASCII string

7.16.2.17 void ST7735_OutUDec (uint32_t *n*)

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

Parameters

n	32-bit number to be transferred

7.16.2.18 void ST7735_PlotBar (int32_t *y*)

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

Parameters

y	the y coordinate of the bar plotted

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

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

Parameters

ymin	Lower bound of plot
ymax	Upper bound of plot

7.16.2.20 void ST7735_PlotdBfs (int32_t *y*)

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

Parameters

y	the y ADC value of the bar plotted

7.16.2.21 void ST7735_PlotLine (int32_t *y*)

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

Parameters

y	the y coordinate of the point plotted

7.16.2.22 void ST7735_PlotPoint (int32_t *y*)

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

Parameters

y	the y coordinate of the point plotted

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

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

Parameters

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

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

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

Parameters

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

7.16.2.25 void ST7735_SetRotation (uint8_t *m*)

Change the image rotation. Requires 2 bytes of transmission.

Parameters

m	new rotation value (0 to 3)

7.16.2.26 void ST7735_SetTextColor (uint16_t *color*)

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

Parameters

color	16-bit packed color

7.16.2.27 uint16_t ST7735_SwapColor (uint16_t *x*)

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

Parameters

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

Returns

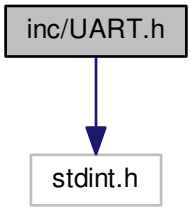
uint16_t 16-bit color in format R, G, B

7.17 inc/UART.h File Reference

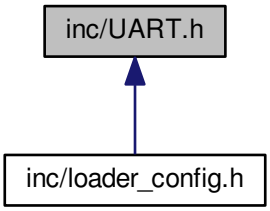
Runs on LM4F120/TM4C123 Use UART0 to implement bidirectional data transfer to and from a computer running HyperTerminal. This time, interrupts and FIFOs are used.

```
#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.

7.17.1 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

7.17.2 Function Documentation

7.17.2.1 char UART_InChar (void)

Wait for new serial port input.

Returns

char ASCII code for key typed

7.17.2.2 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

buf↩ Pt	pointer to empty buffer
max	size of buffer

7.17.2.3 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

7.17.2.4 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

7.17.2.5 void UART_OutChar (char *data*)

8-bit to serial port

Parameters

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

7.17.2.6 void UART_OutString (char * *pt*)

Output String (NULL termination)

Parameters

pt	pointer to a NULL-terminated string to be transferred

7.17.2.7 void UART_OutUDec (uint32_t *n*)

Output a 32-bit number in unsigned decimal format.

Parameters

n	32-bit number to be transferred

7.17.2.8 void UART_OutUHex (uint32_t *number*)

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

Parameters

number	32-bit number to be transferred

Index