Project 1 Module 5

0.1.3

Generated by Doxygen 1.9.8

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 src/DS1631.cpp File Reference

```
#include "mbed.h"
#include "DS1631.h"
```

Variables

- char cmd [ ] = {0x51, 0xAA}
- char read_temp [2]

2.1.1 Variable Documentation

2.1.1.1 cmd

```
char cmd[] = {0x51, 0xAA}
```

2.1.1.2 read_temp

```
char read_temp[2]
```

2.2 src/main.cpp File Reference

```
#include "shared.h"
#include "NHD_0216HZ.h"
#include "DS1631.h"
#include "pindef.h"
```

Macros

- #define MAIN

Functions

- DigitalOut greenLED (LED2)
- void flip (void)
- Serial pc (USBTX, USBRX)
- int main ()

Variables

- volatile uint16_t SwTimerIsrCounter
- Ticker tick
- Timer custom_timer2

2.2.1 Detailed Description

– ECEN 5803 Mastering Embedded System Architecture – – Project 1 Module 4 – – Microcontroller Firmware – – main.cpp –

2.2.1.1 – –

–

2.2.1.2 – Designed for: University of Colorado at Boulder

–
– Designed by: Tim Scherr

2.2.1.3 – Revised by: Student's name

– Version: 3.0 – Date of current revision: 2022-06-20
– Target Microcontroller: ST STM32F401RE – Tools used: ARM mbed compiler – ARM mbed SDK

2.2.1.4 – ST Nucleo STM32F401RE Board

– – Functional Description: Main code file generated by mbed, and then

2.2.1.5 – modified to implement a super loop bare metal OS.

2.2.1.6 – Copyright (c) 2015, 2016, 2022 Tim Scherr All rights reserved.

2.2.2 Macro Definition Documentation

2.2.2.1 MAIN

```
#define MAIN
```

2.2.3 Function Documentation

2.2.3.1 flip()

```
void flip (
            void  )
```

2.2.3.2 greenLED()

```
DigitalOut greenLED (
            LED2  )
```

2.2.3.3 main()

```
int main ( )
```

Cyclical Executive Loop
End while(1) loop

2.2.3.4 pc()

```
Serial pc (
            USBTX ,
            USBRX  )
```

2.2.4 Variable Documentation

2.2.4.1 custom_timer2

```
Timer custom_timer2
```

2.2.4.2 SwTimerIsrCounter

```
volatile uint16_t SwTimerIsrCounter  [extern]
```

2.2.4.3 tick

```
Ticker tick
```

2.3 src/memory.cpp File Reference

Functions to interact with memory, registers, and the UART interface.

```
#include "memory.h"
#include "mbed.h"
#include "stdio.h"
```

Functions

- void read_serial_input (char *buffer, int length)
  Reads input from the serial interface.
- __asm uint32_t get_r0 ()
  Fetch value of R1 register.
- __asm uint32_t get_r1 ()
  Fetch value of R2 register.
- __asm uint32_t get_r2 ()
  Fetch value of R3 register.
- __asm uint32_t get_r3 ()
  Fetch value of R4 register.
- __asm uint32_t get_r4 ()
  Fetch value of R5 register.
- __asm uint32_t get_r5 ()
  Fetch value of R6 register.
- __asm uint32_t get_r6 ()
  Fetch value of R7 register.

- __asm uint32_t get_r7 ()

  Fetch value of R8 register.
- __asm uint32_t get_r8 ()

  Fetch value of R9 register.
- __asm uint32_t get_r9 ()

  Fetch value of R10 register.
- __asm uint32_t get_r10 ()

  Fetch value of R11 register.
- __asm uint32_t get_r11 ()

  Fetch value of R12 register.
- __asm uint32_t get_r12 ()

  Fetch value of R13 register.
- __asm uint32_t get_sp ()

  Fetch value of Stack Pointer register.
- __asm uint32_t get_lr ()

  Fetch value of Link Register.
- __asm uint32_t get_pc ()

  Fetch value of Program Counter register.
- void print_registers ()

  Prints the values of the registers.
- void dump_memory (void)

  Dumps memory content from a given address for a given length.
- void display_last_16_stack_words ()

  Displays the last 16 words in the stack.

Variables

- Serial pc

2.3.1 Detailed Description

Functions to interact with memory, registers, and the UART interface.

Author

Kiran Jojare, Viraj Patel

2.3.2 Function Documentation

2.3.2.1 display_last_16_stack_words()

```
void display_last_16_stack_words (
            void  )
```

Displays the last 16 words in the stack.
Displays the last 16 words of the stack.
This function sends the last 16 words of the current stack over UART.

2.3.2.2 dump_memory()

```
void dump_memory (
            void  )
```

Dumps memory content from a given address for a given length.
Dumps the memory contents in hex format.
This function sends the content of the memory from a specified address and for a given length over UART.

2.3.2.3 get_lr()

```
__asm uint32_t get_lr ( )
```
Fetch value of Link Register.

2.3.2.4 get_pc()

```
__asm uint32_t get_pc ( )
```
Fetch value of Program Counter register.

2.3.2.5 get_r0()

```
__asm uint32_t get_r0 ( )
```
Fetch value of R1 register.

2.3.2.6 get_r1()

```
__asm uint32_t get_r1 ( )
```
Fetch value of R2 register.

2.3.2.7 get_r10()

```
__asm uint32_t get_r10 ( )
```
Fetch value of R11 register.

2.3.2.8 get_r11()

```
__asm uint32_t get_r11 ( )
```
Fetch value of R12 register.

2.3.2.9 get_r12()

```
__asm uint32_t get_r12 ( )
```
Fetch value of R13 register.

2.3.2.10 get_r2()

```
__asm uint32_t get_r2 ( )
```
Fetch value of R3 register.

2.3.2.11 get_r3()

```
__asm uint32_t get_r3 ( )
```
Fetch value of R4 register.

2.3.2.12 get_r4()

```
__asm uint32_t get_r4 ( )
```
Fetch value of R5 register.

2.3.2.13 get_r5()

```
__asm uint32_t get_r5 ( )
```
Fetch value of R6 register.

2.3.2.14 get_r6()

```
__asm uint32_t get_r6 ( )
```
Fetch value of R7 register.

2.3.2.15 get_r7()

```
__asm uint32_t get_r7 ( )
```
Fetch value of R8 register.

2.3.2.16 get_r8()

```
__asm uint32_t get_r8 ( )
```
Fetch value of R9 register.

2.3.2.17 get_r9()

```
__asm uint32_t get_r9 ( )
```
Fetch value of R10 register.

2.3.2.18 get_sp()

```
__asm uint32_t get_sp ( )
```
Fetch value of Stack Pointer register.

2.3.2.19 print_registers()

```
void print_registers (
            void  )
```
Prints the values of the registers.
Prints the values of the core registers to the UART interface.
This function sends the current values of the processor's registers over UART.

2.3.2.20 read_serial_input()

```
void read_serial_input (
            char * buffer,
            int length )
```
Reads input from the serial interface.
Reads serial input into a buffer.
This function captures characters from the UART until a newline or a carriage return character is detected. Additionally, it handles backspaces by erasing the previously entered character.

2.3.3 Variable Documentation

2.3.3.1 pc

```
Serial pc  [extern]
```

2.4 src/memory.h File Reference

Header file to interact with memory, registers, and the UART interface.
`#include <stdint.h>`

Functions

- void read_serial_input (char *buffer, int length)

  Reads serial input into a buffer.
- void print_registers ()

  Prints the values of the core registers to the UART interface.
- void dump_memory (void)

  Dumps the memory contents in hex format.

- void display_last_16_stack_words ()

  Displays the last 16 words of the stack.

2.4.1 Detailed Description

Header file to interact with memory, registers, and the UART interface.

Author

Kiran Jojare, Viraj Patel

2.4.2 Function Documentation

2.4.2.1 display_last_16_stack_words()

```
void display_last_16_stack_words (
            void  )
```

Displays the last 16 words of the stack.
Displays the last 16 words of the stack.
This function sends the last 16 words of the current stack over UART.

2.4.2.2 dump_memory()

```
void dump_memory (
            void  )
```

Dumps the memory contents in hex format.
Dumps the memory contents in hex format.
This function sends the content of the memory from a specified address and for a given length over UART.

2.4.2.3 print_registers()

```
void print_registers (
            void  )
```

Prints the values of the core registers to the UART interface.
Prints the values of the core registers to the UART interface.
This function sends the current values of the processor's registers over UART.

2.4.2.4 read_serial_input()

```
void read_serial_input (
            char * buffer,
            int length )
```

Reads serial input into a buffer.

Parameters

buffer	The buffer to store the read data.
length	The maximum number of characters to read.

Reads serial input into a buffer.
This function captures characters from the UART until a newline or a carriage return character is detected. Additionally, it handles backspaces by erasing the previously entered character.

2.5 memory.h

Go to the documentation of this file.

```
00001 //*****************************************************************************/
00009 //*****************************************************************************/
00010 #ifndef MEMORY_H
00011 #define MEMORY_H
```

```
00012
00013 #include <stdint.h>
00014
00015 //*****************************************************************************/
00020 void read_serial_input(char *buffer, int length);
00021
00022 //*****************************************************************************/
00024 void print_registers();
00025
00026 //*****************************************************************************/
00028 void dump_memory(void);
00029
00030 //*****************************************************************************/
00032 void display_last_16_stack_words();
00033
00034 #endif // MEMORY_H
```

2.6 src/Monitor.cpp File Reference

```
#include <stdio.h>
#include "shared.h"
#include "memory.h"
```

Functions

- void set_display_mode (void)

  Displays a selection menu over UART.
- void chk_UART_msg (void)
  - *fills a message buffer until return is encountered, then calls message processing*
- void UART_msg_process (void)
- uchar8_t is_hex (uchar8_t c)
- void monitor (void)

2.6.1 Detailed Description

– ECEN 5003 Mastering Embedded System Architecture – – Project 1 Module 4 – – Microcontroller Firmware – – Monitor.cpp –

2.6.1.1 – –

–

2.6.1.2 – Designed for: University of Colorado at Boulder

–
– Designed by: Tim Scherr

2.6.1.3 – Revised by: Student's name

– Version: 2.0 – Date of current revision: 2022-06-20
– Target Microcontroller: ST STM32F401RE – Tools used: ARM mbed compiler – ARM mbed SDK

2.6.1.4 – ST Nucleo STM32F401RE Board

–

2.6.1.5 Functional Description: See below

2.6.1.6 – Copyright (c) 2015, 2022 Tim Scherr All rights reserved.

2.6.2 Function Documentation

2.6.2.1 chk_UART_msg()

```
void chk_UART_msg (
            void  )
```

- fills a message buffer until return is encountered, then calls message processing

2.6.2.2 is_hex()

```
uchar8_t is_hex (
            uchar8_t c )
```

2.6.2.3 monitor()

```
void monitor (
            void  )
```

2.6.2.4 set_display_mode()

```
void set_display_mode (
            void  )
```

Displays a selection menu over UART.
This function sends a series of messages over UART to display a menu allowing the user to choose between different modes. Each mode is highlighted with a different color.

2.6.2.5 UART_msg_process()

```
void UART_msg_process (
            void  )
```

UART Input Message Processing

2.7 src/NHD_0216HZ.cpp File Reference

```
#include "mbed.h"
#include <stdarg.h>
#include "NHD_0216HZ.h"
```

2.8 src/shared.h File Reference

```
#include "mbed.h"
```

Macros

- #define OFF 0 /∗ used for readability ∗/
- #define ON 1 /∗ used for readability ∗/
- #define NO 0 /∗ used for readability ∗/
- #define YES 1 /∗ used for readability ∗/
- #define TEN 10
- #define TIMER0 TMR0
- #define SEC 10000 /∗ 10000 timer0 interrupts per second (100 usec.) ∗/

- #define T100MS 0.1∗SEC
- #define T2S 2∗SEC
- #define LED_FLASH_PERIOD .5 /∗ in seconds ∗/
- #define CLOCK_FREQUENCY_MHZ 8
- #define CODE_VERSION "2.0 2016/09/29" /∗ YYYY/MM/DD ∗/
- #define COPYRIGHT "Copyright (c) University of Colorado"
- #define RX_BUF_SIZE 10 /∗ size of receive buffer in bytes ∗/
- #define TX_BUF_SIZE 40 /∗ size of transmit buffer in bytes ∗/
- #define MSG_BUF_SIZE 10

Typedefs

- typedef unsigned char uchar8_t
- typedef unsigned char bit
- typedef unsigned int uint32_t
- typedef unsigned short uint16_t

Enumerations

- enum boolean { FALSE , TRUE }

- enum dmode {
  QUIET , NORMAL , DEBUG , VERSION ,
  REGISTERS , MEMORY , STACK }

Functions

- void monitor (void)
- void timer0 (void)
- void serial (void)
- void UART_put (uchar8_t)
- uchar8_t UART_get (void)
- uchar8_t UART_input (void)
- void UART_direct_msg_put (const char ∗)
- void UART_msg_put (const char ∗)
- void UART_direct_hex_put (uchar8_t)
- void UART_direct_put (uchar8_t)
- void UART_hex_put (uchar8_t)
- void UART_low_nibble_put (uchar8_t)
- void UART_high_nibble_put (uchar8_t)
- void chk_UART_msg (void)
  - *fills a message buffer until return is encountered, then calls message processing*
- void UART_msg_process (void)
- void status_report (void)
- void set_display_mode (void)

  Displays a selection menu over UART.

Variables

- unsigned char Error_status
- uchar8_t display_timer
- uchar8_t display_flag
- uchar8_t tx_in_progress
- uchar8_t ∗ rx_in_ptr
- uchar8_t ∗ rx_out_ptr

- uchar8_t * tx_in_ptr
- uchar8_t * tx_out_ptr
- volatile uchar8_t swtimer0
- volatile uchar8_t swtimer1
- volatile uchar8_t swtimer2
- volatile uchar8_t swtimer3
- volatile uchar8_t swtimer4
- volatile uchar8_t swtimer5
- volatile uchar8_t swtimer6
- volatile uchar8_t swtimer7
- uchar8_t serial_flag
- enum dmode display_mode
- uchar8_t rx_buf [ ]
- uchar8_t tx_buf [ ]
- uchar8_t msg_buf [MSG_BUF_SIZE]
- uchar8_t msg_buf_idx

2.8.1 Detailed Description

– ECEN 5803 Mastering Embedded System Architecture – – Project 1 Module 4 – – Microcontroller Firmware – – shared.h –

2.8.1.1 – –

–

2.8.1.2 – Designed for: University of Colorado at Boulder

–
– Designed by: Tim Scherr

2.8.1.3 – Revised by: Student's name

– Version: 3.0 – Date of current revision: 2022-06-20
– Target Microcontroller: ST STM32F401RE – Tools used: ARM mbed compiler – ARM mbed SDK

2.8.1.4 – ST Nucleo STM32F401RE Board

–

2.8.1.5 – Functional Description: Header file for all globals

2.8.1.6 – Copyright (c) 2015, 2022 Tim Scherr All rights reserved.

2.8.2 Macro Definition Documentation

2.8.2.1 CLOCK_FREQUENCY_MHZ

```
#define CLOCK_FREQUENCY_MHZ 8
```

2.8.2.2 CODE_VERSION

```
#define CODE_VERSION "2.0 2016/09/29" /* YYYY/MM/DD */
```

2.8.2.3 COPYRIGHT

```
#define COPYRIGHT "Copyright (c) University of Colorado"
```

2.8.2.4 LED_FLASH_PERIOD

```
#define LED_FLASH_PERIOD .5 /* in seconds */
```

2.8.2.5 MSG_BUF_SIZE

```
#define MSG_BUF_SIZE 10
```

2.8.2.6 NO

```
#define NO 0 /* used for readability */
```

2.8.2.7 OFF

```
#define OFF 0 /* used for readability */
```

2.8.2.8 ON

```
#define ON 1 /* used for readability */
```

2.8.2.9 RX_BUF_SIZE

```
#define RX_BUF_SIZE 10 /* size of receive buffer in bytes */
```

2.8.2.10 SEC

```
#define SEC 10000 /* 10000 timer0 interrupts per second (100 usec.)  */
```

2.8.2.11 T100MS

```
#define T100MS 0.1*SEC
```

2.8.2.12 T2S

```
#define T2S 2*SEC
```

2.8.2.13 TEN

```
#define TEN 10
```

2.8.2.14 TIMER0

```
#define TIMER0 TMR0
```

2.8.2.15 TX_BUF_SIZE

```
#define TX_BUF_SIZE 40 /* size of transmit buffer in bytes */
```

2.8.2.16 YES

```
#define YES 1 /* used for readability */
```

2.8.3 Typedef Documentation

2.8.3.1 bit

```
typedef unsigned char bit
```

2.8.3.2 uchar8_t

```
typedef unsigned char uchar8_t
```

2.8.3.3 uint16_t

```
typedef unsigned short uint16_t
```

2.8.3.4 uint32_t

```
typedef unsigned int uint32_t
```

2.8.4 Enumeration Type Documentation

2.8.4.1 boolean

```
enum boolean
```

Enumerator

FALSE	
TRUE	

2.8.4.2 dmode

```
enum dmode
```

Enumerator

QUIET	
NORMAL	
DEBUG	
VERSION	
REGISTERS	
MEMORY	
STACK	

2.8.5 Function Documentation

2.8.5.1 chk_UART_msg()

```
void chk_UART_msg (
            void  )  [extern]
```

- fills a message buffer until return is encountered, then calls message processing

2.8.5.2 monitor()

```
void monitor (
            void  )  [extern]
```

2.8.5.3 serial()

```
void serial (
            void  )  [extern]
```

function polls the serial port for Rx or Tx data

2.8.5.4 set_display_mode()

```
void set_display_mode (
            void  )  [extern]
```

Displays a selection menu over UART.

This function sends a series of messages over UART to display a menu allowing the user to choose between different modes. Each mode is highlighted with a different color.

2.8.5.5 status_report()

```
void status_report (
            void  )  [extern]
```

2.8.5.6 timer0()

```
void timer0 (
            void  )  [extern]
```

2.8.5.7 UART_direct_hex_put()

```
void UART_direct_hex_put (
            uchar8_t c )  [extern]
```

2.8.5.8 UART_direct_msg_put()

```
void UART_direct_msg_put (
            const char * str )  [extern]
```

2.8.5.9 UART_direct_put()

```
void UART_direct_put (
            uchar8_t  )  [extern]
```

2.8.5.10 UART_get()

```
uchar8_t UART_get (
            void  )  [extern]
```

2.8.5.11 UART_hex_put()

```
void UART_hex_put (
            uchar8_t c )  [extern]
```

2.8.5.12 UART_high_nibble_put()

```
void UART_high_nibble_put (
            uchar8_t  )  [extern]
```

2.8.5.13 UART_input()

```
uchar8_t UART_input (
            void  )  [extern]
```

2.8.5.14 UART_low_nibble_put()

```
void UART_low_nibble_put (
            uchar8_t  )  [extern]
```

2.8.5.15 UART_msg_process()

```
void UART_msg_process (
            void  )  [extern]
```

UART Input Message Processing

2.8.5.16 UART_msg_put()

```
void UART_msg_put (
            const char * str )  [extern]
```

2.8.5.17 UART_put()

```
void UART_put (
            uchar8_t c )  [extern]
```

2.8.6 Variable Documentation

2.8.6.1 display_flag

```
uchar8_t display_flag  [extern]
```

2.8.6.2 display_mode

```
enum dmode display_mode  [extern]
```

2.8.6.3 display_timer

```
uchar8_t display_timer  [extern]
```

2.8.6.4 Error_status

```
unsigned char Error_status  [extern]
```

2.8.6.5 msg_buf

```
uchar8_t msg_buf[MSG_BUF_SIZE]  [extern]
```

2.8.6.6 msg_buf_idx

```
uchar8_t msg_buf_idx  [extern]
```

2.8.6.7 rx_buf

```
uchar8_t rx_buf[]  [extern]
```

2.8.6.8 rx_in_ptr

```
uchar8_t* rx_in_ptr  [extern]
```

2.8.6.9 rx_out_ptr

```
uchar8_t* rx_out_ptr  [extern]
```

2.8.6.10 serial_flag

```
uchar8_t serial_flag  [extern]
```

2.8.6.11 swtimer0

```
volatile uchar8_t swtimer0  [extern]
```

2.8.6.12 swtimer1

```
volatile uchar8_t swtimer1  [extern]
```

2.8.6.13 swtimer2

```
volatile uchar8_t swtimer2  [extern]
```

2.8.6.14 swtimer3

```
volatile uchar8_t swtimer3  [extern]
```

2.8.6.15 swtimer4

```
volatile uchar8_t swtimer4  [extern]
```

2.8.6.16 swtimer5

```
volatile uchar8_t swtimer5  [extern]
```

2.8.6.17 swtimer6

```
volatile uchar8_t swtimer6  [extern]
```

2.8.6.18 swtimer7

```
volatile uchar8_t swtimer7  [extern]
```

2.8.6.19 tx_buf

```
uchar8_t tx_buf[]  [extern]
```

2.8.6.20 tx_in_progress

```
uchar8_t tx_in_progress  [extern]
```

2.8.6.21 tx_in_ptr

```
uchar8_t* tx_in_ptr  [extern]
```

2.8.6.22 tx_out_ptr

```
uchar8_t* tx_out_ptr  [extern]
```

2.9 shared.h

Go to the documentation of this file.

```
00001
00031  #include "mbed.h"
00032
00033  /*****************************************************************************
00034 * #defines available to all modules included here
00035 *****************************************************************************/
00036 #define OFF 0                       /* used for readability */
00037 #define ON 1                        /* used for readability */
00038 #define NO 0                        /* used for readability */
00039 #define YES 1                       /* used for readability */
00040 #define TEN 10
00041
00042 #define TIMER0 TMR0
00043 #define SEC 10000           /* 10000 timer0 interrupts per second (100 usec.) */
00044
00045 #define T100MS 0.1*SEC
00046 #define T2S    2*SEC
00047
00048 #define LED_FLASH_PERIOD .5   /* in seconds */
```

```
00049
00050 #define CLOCK_FREQUENCY_MHZ 8
00051 #define CODE_VERSION "2.0 2016/09/29"   /*   YYYY/MM/DD  */
00052 #define COPYRIGHT "Copyright (c) University of Colorado"
00053
00054  enum boolean { FALSE, TRUE };
00055  enum dmode {QUIET, NORMAL, DEBUG, VERSION, REGISTERS, MEMORY, STACK};
00056
00057  typedef unsigned char uchar8_t;
00058  typedef unsigned char bit;
00059  typedef unsigned int uint32_t;
00060  typedef unsigned short uint16_t;
00061
00062 #ifdef __cplusplus
00063 extern "C" {
00064 #endif
00065
00066 /*************************************************************************/
00067 /*             Global Variable declarations                              */
00068 /*************************************************************************/
00069
00070  extern unsigned char Error_status;          // Variable for debugging use
00071  extern uchar8_t  display_timer;  // \var 1 second software timer for display
00072  extern uchar8_t  display_flag;   // flag between timer interrupt and monitor.c,
00073                         // like a binary semaphore
00074  extern uchar8_t tx_in_progress;
00075  extern uchar8_t *rx_in_ptr; /* pointer to the receive in data */
00076  extern uchar8_t *rx_out_ptr; /* pointer to the receive out data*/
00077  extern uchar8_t *tx_in_ptr; /* pointer to the transmit in data*/
00078  extern uchar8_t *tx_out_ptr; /*pointer to the transmit out */
00079 #define RX_BUF_SIZE 10            /* size of receive buffer in bytes */
00080 #define TX_BUF_SIZE 40           /* size of transmit buffer in bytes */
00081
00082 /*************************************************************************
00083 * Some variable definitions are done in the module main.c and are externed in
00084 * all other modules.  The following section is visible to main.c only.
00085 *************************************************************************/
00086 #ifdef MAIN
00087
00088  enum dmode display_mode = QUIET;
00089
00090  uchar8_t serial_flag = 0;
00091
00092  uchar8_t tx_in_progress;
00093  uchar8_t *rx_in_ptr; /* pointer to the receive in data */
00094  uchar8_t *rx_out_ptr; /* pointer to the receive out data*/
00095  uchar8_t *tx_in_ptr; /* pointer to the transmit in data*/
00096  uchar8_t *tx_out_ptr; /*pointer to the transmit out */
00097
00098  uchar8_t  rx_buf[RX_BUF_SIZE];      /* define the storage */
00099  uchar8_t  tx_buf[TX_BUF_SIZE];      /* define the storage */
00100
00101 #define MSG_BUF_SIZE 10
00102  uchar8_t msg_buf[MSG_BUF_SIZE]; // define the storage for UART received messages
00103  uchar8_t msg_buf_idx = 0;    // index into the received message buffer
00104
00105 /*************************************************************************
00106 * Some variable definitions are done in the module main.c and are externed in
00107 * all other modules.  The following section is visible to all modules EXCEPT
00108 * main.c.
00109 *************************************************************************/
00110 #else
00111
00112 /**********************/
00113 /*   Declarations     */
00114 /**********************/
00115
00116    extern volatile     uchar8_t swtimer0;
00117    extern volatile     uchar8_t swtimer1;
00118    extern volatile     uchar8_t swtimer2;
00119    extern volatile     uchar8_t swtimer3;
00120    extern volatile     uchar8_t swtimer4;
00121    extern volatile     uchar8_t swtimer5;
00122    extern volatile     uchar8_t swtimer6;
00123    extern volatile     uchar8_t swtimer7;
00124
00125   extern uchar8_t serial_flag;
00126
00127   extern enum dmode display_mode;
00128
00129
00130   extern uchar8_t  rx_buf[];      /* declare the storage */
00131   extern uchar8_t  tx_buf[];      /* declare the storage */
00132
00133 #define MSG_BUF_SIZE 10
00134   extern  uchar8_t msg_buf[MSG_BUF_SIZE]; // declare the storage for UART received messages
00135   extern  uchar8_t msg_buf_idx;         // index into the received message buffer
```

```
00136
00137 #endif
00138
00139 /*****************************************************************************
00140 * All function prototypes are externed in all the modules.
00141 *****************************************************************************/
00142 extern void monitor(void);  /* located in module monitor.c */
00143 extern void timer0(void);   /* located in module timer0.c */
00144 extern void serial(void);   /* located in module UART_poll.c */
00145
00146 extern void UART_put(uchar8_t);                 /* located in module UART_poll.c */
00147 extern uchar8_t UART_get(void);                 /* located in module UART_poll.c */
00148 extern uchar8_t UART_input(void);               /* located in module UART_poll.c */
00149 extern void UART_direct_msg_put(const char *);
00150                                                 /* located in module UART_poll.c */
00151 extern void UART_msg_put(const char *);
00152                                                 /* located in module UART_poll.c */
00153 extern void UART_direct_hex_put(uchar8_t);      /* located in module UART_poll.c */
00154 extern void UART_direct_put(uchar8_t);          /* located in module UART_poll.c */
00155 extern void UART_hex_put(uchar8_t);             /* located in module UART_poll.c */
00156 extern void UART_low_nibble_put(uchar8_t);      /* located in module UART_poll.c */
00157 extern void UART_high_nibble_put(uchar8_t);     /* located in module UART_poll.c */
00158 extern void chk_UART_msg(void);                 /* located in module monitor.c */
00159 extern void UART_msg_process(void);             /* located in module monitors.c */
00160 extern void status_report(void);                /* located in module monitor.c */
00161 extern void set_display_mode(void);             /* located in module monitor.c */
00162
00163 #ifdef __cplusplus
00164 }
00165 #endif
00166
```

2.10 src/timer0.cpp File Reference

```
#include "shared.h"
#include "mbed.h"
```

Macros

- #define System Timer_INCREMENT_IN_US 1000

Typedefs

- typedef unsigned char uchar8_t
- typedef unsigned char bit
- typedef unsigned int uint32_t
- typedef unsigned short uint16_t

Functions

- DigitalOut redLED (PA_7)
- void toggleRedLED ()
- void timer0 (void)

Variables

- volatile uchar8_t swtimer0 = 0
- volatile uchar8_t swtimer1 = 0
- volatile uchar8_t swtimer2 = 0
- volatile uchar8_t swtimer3 = 0
- volatile uchar8_t swtimer4 = 0
- volatile uchar8_t swtimer5 = 0
- volatile uchar8_t swtimer6 = 0
- volatile uchar8_t swtimer7 = 0
- volatile uint16_t SwTimerIsrCounter = 0U
- uchar8_t display_timer = 0

- uchar8_t display_flag = 0
- int redLEDTickCounter = 0
- Serial pc
- Timer custom_timer
- int count30s = 0

2.10.1 Macro Definition Documentation

2.10.1.1 System

```
#define System Timer_INCREMENT_IN_US 1000
```

\file timer0.cpp

– ECEN 5803 Mastering Embedded System Architecture – – Project 1 Module 4 – – Microcontroller Firmware – – Timer0.cpp –

2.10.1.2 – –

–

2.10.1.3 – Designed for: University of Colorado at Boulder

–
– Designed by: Tim Scherr

2.10.1.4 – Revised by: Student's name

– Version: 3.0 – Date of current revision: 2022-06-20
– Target Microcontroller: ST STM32F401RE – Tools used: ARM mbed compiler – ARM mbed SDK

2.10.1.5 – ST Nucleo STM32F401RE Board

–
Functional Description:
This file contains code for the only interrupt routine, based on the System Timer.
The System Timer interrupt happens every 100 us as determined by mbed Component Configuration. The System Timer interrupt acts as the real time scheduler for the firmware. Each time the interrupt occurs, different tasks are done based on critical timing requirement for each task.
There are 256 timer states (an 8-bit counter that rolls over) so the period of the scheduler is 25.6 ms. However, some tasks are executed every other time (the 200 us group) and some every 4th time (the 400 us group) and so on. Some high priority tasks are executed every time. The code for the tasks is divided up into the groups which define how often the task is executed. The structure of the code is shown below:
I. Entry and timer state calculation II. 100 us group A. Fast Software timers B. Read Sensors C. Update III. 200 us group A. B. IV. 400 us group A. Medium Software timers B. V. 800 us group A. Set 420 PWM Period VI 1.6 ms group A. Display timer and flag B. Heartbeat/ LED outputs VII 3.2 ms group A. Slow Software Timers
VIII 6.4 ms group A A. Very Slow Software Timers IX. Long time group A. Determine Mode B. Heartbeat/ LED outputs
X. Exit
– Copyright (c) 2015, 2022 Tim Scherr All rights reserved.

2.10.2 Typedef Documentation

2.10.2.1 bit

```
typedef unsigned char bit
```

2.10.2.2 uchar8_t

```
typedef unsigned char uchar8_t
```

2.10.2.3 uint16_t

```
typedef unsigned short uint16_t
```

2.10.2.4 uint32_t

```
typedef unsigned int uint32_t
```

2.10.3 Function Documentation

2.10.3.1 redLED()

```
DigitalOut redLED (
            PA_7  )
```

2.10.3.2 timer0()

```
void timer0 (
            void  )
```

2.10.3.3 toggleRedLED()

```
void toggleRedLED ( )
```

2.10.4 Variable Documentation

2.10.4.1 count30s

```
int count30s = 0
```

2.10.4.2 custom_timer

```
Timer custom_timer
```

2.10.4.3 display_flag

```
uchar8_t display_flag = 0
```

2.10.4.4 display_timer

```
uchar8_t display_timer = 0
```

2.10.4.5 pc

```
Serial pc  [extern]
```

2.10.4.6 redLEDTickCounter

```
int redLEDTickCounter = 0
```

2.10.4.7 swtimer0

```
volatile uchar8_t swtimer0 = 0
```

2.10.4.8 swtimer1

```
volatile uchar8_t swtimer1 = 0
```

2.10.4.9 swtimer2

```
volatile uchar8_t swtimer2 = 0
```

2.10.4.10 swtimer3

```
volatile uchar8_t swtimer3 = 0
```

2.10.4.11 swtimer4

```
volatile uchar8_t swtimer4 = 0
```

2.10.4.12 swtimer5

```
volatile uchar8_t swtimer5 = 0
```

2.10.4.13 swtimer6

```
volatile uchar8_t swtimer6 = 0
```

2.10.4.14 swtimer7

```
volatile uchar8_t swtimer7 = 0
```

2.10.4.15 SwTimerIsrCounter

```
volatile uint16_t SwTimerIsrCounter = 0U
```

2.11 src/UART_poll.cpp File Reference

```
#include <stdio.h>
#include "shared.h"
```

Macros

- #define OERR (USART2->SR & USART_SR_ORE)
- #define CREN (USART2->CR1 & USART_CR1_RE)
- #define RCREG USART2->DR
- #define FERR (USART2->SR & USART_SR_FE)
- #define RCIF (USART2->SR & USART_SR_RXNE)
- #define TXIF (USART2->SR & USART_SR_TXE)
- #define TXREG USART2->DR
- #define TRMT (USART2->SR & USART_SR_TC)

Functions

- void serial (void)
- void UART_direct_msg_put (const char *str)
- void UART_put (uchar8_t c)
- uchar8_t UART_get (void)
- uchar8_t UART_input (void)
- void UART_msg_put (const char *str)
- uchar8_t hex_to_asc (uchar8_t c)
- uchar8_t asc_to_hex (uchar8_t c)
- void UART_hex_put (unsigned char c)
- void UART_direct_hex_put (unsigned char c)

Variables

- uchar8_t error_count = 0

2.11.1 Detailed Description

– ECEN 5803 Mastering Embedded System Architecture – – Project 1 Module 4 – – Microcontroller Firmware – – UART_poll.c –

2.11.1.1 – –

–

2.11.1.2 – Designed for: University of Colorado at Boulder

–
– Designed by: Tim Scherr

2.11.1.3 – Revised by: Student's name

– Version: 3.0 – Date of current revision: 2022-06-20
– Target Microcontroller: ST STM32F401RE – Tools used: ARM mbed compiler – ARM mbed SDK

2.11.1.4 – ST Nucleo STM32F401RE Board

– – Functional Description: This file contains routines that support messages – to and from the UART port. Included are: – Serial() - a routine to send/receive bytes on the UART port to – the transmit/receive buffers – UART_put() - a routine that puts a character in the transmit buffer – UART_get() - a routine that gets the next character from the receive – buffer – UART_msg_put() - a routine that puts a string in the transmit buffer – UART_direct_msg_put() - routine that sends a string out the UART port – UART_input() - determines if a character has been received

2.11.1.5 – UART_hex_put() - a routine that puts a hex byte in the transmit buffer

2.11.1.6 – Copyright (c) 2015, 2022 Tim Scherr All rights reserved.

2.11.2 Macro Definition Documentation

2.11.2.1 CREN

```
#define CREN (USART2->CR1 & USART_CR1_RE)
```

2.11.2.2 FERR

```
#define FERR (USART2->SR & USART_SR_FE)
```

2.11.2.3 OERR

```
#define OERR (USART2->SR & USART_SR_ORE)
```

2.11.2.4 RCIF

```
#define RCIF (USART2->SR & USART_SR_RXNE)
```

2.11.2.5 RCREG

```
#define RCREG USART2->DR
```

2.11.2.6 TRMT

```
#define TRMT (USART2->SR & USART_SR_TC)
```

2.11.2.7 TXIF

```
#define TXIF (USART2->SR & USART_SR_TXE)
```

2.11.2.8 TXREG

```
#define TXREG USART2->DR
```

2.11.3 Function Documentation

2.11.3.1 asc_to_hex()

```
uchar8_t asc_to_hex (
            uchar8_t c )
```

2.11.3.2 hex_to_asc()

```
uchar8_t hex_to_asc (
            uchar8_t c )
```

2.11.3.3 serial()

```
void serial (
            void  )
```

function polls the serial port for Rx or Tx data

2.11.3.4 UART_direct_hex_put()

```
void UART_direct_hex_put (
            unsigned char c )
```

2.11.3.5 UART_direct_msg_put()

```
void UART_direct_msg_put (
            const char * str )
```

2.11.3.6 UART_get()

```
uchar8_t UART_get (
            void  )
```

2.11.3.7 UART_hex_put()

```
void UART_hex_put (
            unsigned char c )
```

2.11.3.8 UART_input()

```
uchar8_t UART_input (
            void  )
```

2.11.3.9 UART_msg_put()

```
void UART_msg_put (
            const char * str )
```

2.11.3.10 UART_put()

```
void UART_put (
            uchar8_t c )
```

2.11.4 Variable Documentation

2.11.4.1 error_count

```
uchar8_t error_count = 0
```

Index