Report

Laboratory 6

December 7, 2017

Author:
Caroline Nilsson *(cn222nd)*
Daniel Alm Grundström *(dg222dw)*
Term: HT 2017
Course: 1DT301 - Computer Technology I

Contents

1 Assignment 1

```c
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//    1DT301, Computer Technology I
//    Date: 2017-12-07
//    Author:
//                          Caroline Nilsson            (cn222nd)
//                          Daniel Alm Grundström       (dg222dw)
//
//    Lab number:           6
//    Title:                CyberTech Wall Display
//
//    Hardware:             STK600, CPU ATmega2560, CyberTech Wall Display
//
//    Function:             Writes the character '#' to the CyberTech Display
//
//    Input ports:          N/A
//
//    Output ports:         USART1:
//                              RXD1/TXD1 connected to PD2/PD3
//                              RS232 connected to CyberTech Display
//
//    Functions:            main
//    Included files:       display_utils.h
//                          avr/io.h
//                          string.h
//
//    Other information:    More information about used functions available
//                          in 'display_utils.c'
//
//    Changes in program:   2017-12-07:
//                          Adds headers and additional comments.
//
//                          2017-10-17:
//                          Removes debug code.
//
//                          2017-10-10:
//                          Fix bugs, restructures code and adds comments.
//
//                          2017-10-09:
//                          Adds implementation.
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#include <string.h>     // strncpy

#include <avr/io.h>     // TXEN1

#include "display_utils.h"

int main() {
    init_serial_comm(1 << TXEN1);           // Enable USART for transmitting

    Frame info_frame = create_frame(Information);
    Frame image_frame = create_frame(Image);
    info_frame.address = 'Z';               // 'Z' is used instead of 'A' so display is cleared

    strncpy(info_frame.line_1, "#", 1);     // info_frame.line_1 = "#"

    set_checksum(&info_frame, calculate_checksum(&info_frame, 1));
    set_checksum(&image_frame, calculate_checksum(&image_frame, 1));

    send_frame(&info_frame, 1);
    send_frame(&image_frame, 1);

    return 0;
}
```

2 Assignment 2

```c
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//    1DT301, Computer Technology I
//    Date: 2017-12-07
//    Author:
//                          Caroline Nilsson           (cn222nd)
//                          Daniel Alm Grundström      (dg222dw)
//
//    Lab number:           6
//    Title:                CyberTech Wall Display
//
//    Hardware:             STK600, CPU ATmega2560, CyberTech Wall Display
//
//    Function:             Writes three lines of text to the CyberTech Display
//
//    Input ports:          N/A
//
//    Output ports:         USART1:
//                              RXD1/TXD1 connected to PD2/PD3
//                              RS232 connected to CyberTech Display
//
//    Functions:            main
//
//    Included files:       display_utils.h
//                          avr/io.h
//                          string.h
//
//    Other information:    More information about used functions available
//                          in 'display_utils.c'
//
//    Changes in program:   2017-12-07:
//                          Adds headers and comments.
//
//                          2017-10-17:
//                          Removes debug code.
//
//                          2017-10-10:
//                          Adds implementation.
//
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#include <string.h>      // strncpy, strlen

#include <avr/io.h>      // TXEN1

#include "display_utils.h"

#define LINE_1 "Rad 1"
#define LINE_2 "Rad 2"
#define LINE_3 "Rad 3"

int main() {
    init_serial_comm(1 << TXEN1);

    Frame image_frame = create_frame(Image);
    set_checksum(&image_frame, calculate_checksum(&image_frame, 1));

    // Initializes the two information frames (one for the first two lines
    // and the other for the last line)
    Frame first_lines = create_frame(Information);
    Frame last_line = create_frame(Information);
    first_lines.address = 'A';
    last_line.address = 'B';

    strncpy(first_lines.line_1, LINE_1, strlen(LINE_1));
    strncpy(first_lines.line_2, LINE_2, strlen(LINE_2));
    set_checksum(&first_lines, calculate_checksum(&first_lines, 1));

    send_frame(&first_lines, 1);
    send_frame(&image_frame, 1);

    strncpy(last_line.line_1, LINE_3, strlen(LINE_3));
    set_checksum(&last_line, calculate_checksum(&last_line, 3));

    send_frame(&last_line, 3);
    send_frame(&image_frame, 3);

    return 0;
}
```

3 Assignment 3

```c
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//  1DT301, Computer Technology I
//  Date: 2017-12-07
//  Author:
//                      Caroline Nilsson          (cn222nd)
//                      Daniel Alm Grundström     (dg222dw)
//
//  Lab number:         6
//  Title:              CyberTech Wall Display
//
//  Hardware:           STK600, CPU ATmega2560, CyberTech Wall Display
//
//  Function:           Scrolls 5 lines of text on the CyberTech Display, with 5 seconds between each scroll
//
//  Input ports:        N/A
//
//  Output ports:
//                      - RXD1/TXD1 connected to PD2/PD3
//                      - RS232 connected to CyberTech Display
//
//  Functions:          main,
//                      scroll_text - Runs loop which scrolls the text in 'text' on the CyberTech Display
//
//  Included files:     display_utils.h
//                      avr/io.h
//                      util/delay.h
//                      string.h
//
//  Other information:  More information about used functions available
//                      in 'display_utils.c'
//
//                      Lines in 'text' need to be exactly 24 characters long
//
//  Changes in program: 2017-12-07:
//                      Adds headers and comments.
//
//                      2017-10-17:
//                      Removes debug code.
//
//                      2017-10-11:
//                      Adds implementation.
//
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#include <string.h>         // strncpy

#include <avr/io.h>         // TXEN1

#define F_CPU 1000000UL     // 1MHz
#include <util/delay.h>     // _delay_ms

#include "display_utils.h"

static const uint8_t SCROLL_LINES = 5;
static const char *text[] = {
    "Assignment #6           ",
    "Computer Technology     ",
    "Computer Science 2017   ",
    "Daniel Alm Grundstrom   ",
    "Caroline Nilsson        "
};

void scroll_text();

int main() {
    init_serial_comm(1 << TXEN1);
    scroll_text();

    return 0;
}

void scroll_text() {
    Frame first_lines = create_frame(Information);
    Frame last_line = create_frame(Information);
    Frame image_frame = create_frame(Image);
    uint8_t next_line = 0;

    first_lines.address = 'A';
    last_line.address = 'B';

    // Image frame does not change, so checksum can be calculated directly
    set_checksum(&image_frame, calculate_checksum(&image_frame, 1));

    for (;;) {
        strncpy(last_line.line_1, first_lines.line_2, INFO_FRAME_LINE_LEN);
        strncpy(first_lines.line_2, first_lines.line_1, INFO_FRAME_LINE_LEN);
        strncpy(first_lines.line_1, text[next_line], INFO_FRAME_LINE_LEN);

        // update checksum
        set_checksum(&first_lines, calculate_checksum(&first_lines, 1));

        // write first two lines to display
        send_frame(&first_lines, 1);
        send_frame(&image_frame, 1);

        _delay_ms(10);  // Short delay to make sure display has time to receive and act on command

        // update checksum
        set_checksum(&last_line, calculate_checksum(&last_line, 3));
```

```
 99
100             // write last line to display
101             send_frame(&last_line, 3);
102             send_frame(&image_frame, 3);
103
104             next_line = next_line + 1;
105
106             if (next_line >= SCROLL_LINES) {
107                 next_line = 0;
108             }
109
110             _delay_ms(5000);
111         }
112     }
```

4 Assignment 4

```
1  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
2  //    1DT301, Computer Technology I
3  //    Date: 2017-12-07
4  //    Author:
5  //                          Caroline Nilsson            (cn222nd)
6  //                          Daniel Alm Grundström       (dg222dw)
7  //
8  //    Lab number:           6
9  //    Title:                CyberTech Wall Display
10 //
11 //    Hardware:             STK600, CPU ATmega2560, CyberTech Wall Display
12 //
13 //    Function:             Reads 3 lines of text from a PuTTY terminal and
14 //                          outputs them on the CyberTech Display.
15 //
16 //                          First waits for the address for the first two lines ('A'-'Z'),
17 //                          then allows two lines of max 24 characters to be entered. After
18 //                          each line a '#' terminating character needs to be entered.
19 //
20 //                          The program then waits for the user to input the address of the last
21 //                          line ('A'-'Z') and then reads one additional line, which is also terminated
22 //                          with a '#'. After this terminating '#', the text is displayed on the
23 //                          CyberTech Display.
24 //
25 //                          For example, entering 'A', then "Rad 1#Rad 2#", then 'B' and "Rad 3#"
26 //                          outputs:
27 //
28 //                          [Rad 1                  ]
29 //                          [Rad 2                  ]
30 //                          [Rad 3                  ]
31 //
32 //    Input ports:          RS232 connected to Computer serial port
33 //
34 //    Output ports:
35 //                          - RXD1/TXD1 connected to PD2/PD3
36 //                          - RS232 connected to CyberTech Display
37 //
38 //    Functions:            main,
39 //                          get_line - Reads a line from of text from PuTTY
40 //
41 //    Included files:       display_utils.h
42 //                          avr/io.h
43 //                          string.h
44 //
45 //    Other information:    More information about used functions available
46 //                          in 'display_utils.c'
47 //
48 //    Changes in program:   2017-12-07:
49 //                          Adds headers and comments.
50 //
51 //                          2017-10-17:
52 //                          Removes debug code.
53 //
54 //                          2017-10-11:
55 //                          Adds implementation.
56 //
57 //<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
58 #include <string.h>
59
60 #include <avr/io.h>
61
62 #include "display_utils.h"
63
64 #define NEW_LINE '#'
65
66 void get_line(char line[]);
67
68 int main() {
69     init_serial_comm((1 << TXEN1) | (1 << RXEN1));       // Enable USART for transmitting and receiving
70
71     Frame image_frame = create_frame(Image);
72     set_checksum(&image_frame, calculate_checksum(&image_frame, 1));
73
74     Frame first_lines = create_frame(Information);
75     Frame last_line = create_frame(Information);
76
77     first_lines.address = uart_receive();
78     get_line(first_lines.line_1);
79     get_line(first_lines.line_2);
80     last_line.address = uart_receive();
81     get_line(last_line.line_1);
82
83     set_checksum(&first_lines, calculate_checksum(&first_lines, 1));
84     set_checksum(&last_line, calculate_checksum(&last_line, 3));
85
86     send_frame(&first_lines, 1);
87     send_frame(&image_frame, 1);
88
89     send_frame(&last_line, 3);
90     send_frame(&image_frame, 3);
91
92     return 0;
93 }
94
95 void get_line(char line[]) {
96     char received;
97     uint8_t line_index = 0;
98
```

```
 99      do {
100          received = (char)uart_receive();
101
102          if (line_index < INFO_FRAME_LINE_LEN && received != NEW_LINE) {
103              line[line_index++] = received;
104          }
105
106      } while (received != NEW_LINE);
107  }
```

5 Assignment 5

We were told this assignment didn't need to be completed and that it was sufficient to complete assignment 4.

A display_utils.h

```c
#ifndef DISPLAY_UTILS_H
#define DISPLAY_UTILS_H

#include <stdint.h>     // uint8_t

// Constants
#define TRANSFER_RATE 6                 // = 2400 bps (for 1MHz)

#define INFO_FRAME_COMMAND_LEN  5
#define IMG_FRAME_COMMAND_LEN  4
#define INFO_FRAME_LINE_LEN 24
#define FRAME_CHECKSUM_LEN  2

enum FrameType {
    Information,
    Image
};

typedef enum FrameType FrameType;

/*
 * Structure for display protocol frame. Used both for the Information frame
 * and the Image frame.
 */
struct Frame {
    FrameType type;
    uint8_t start;
    char address;
    char command[INFO_FRAME_COMMAND_LEN];
    char line_1[INFO_FRAME_LINE_LEN];
    char line_2[INFO_FRAME_LINE_LEN];
    char checksum[FRAME_CHECKSUM_LEN];
    uint8_t end;
};

typedef struct Frame Frame;

// Function prototypes
void init_serial_comm(uint8_t ucsr1b_flags);
Frame create_frame(FrameType type);
void send_frame(const Frame *frame, int line);
void uart_transmit(unsigned char data);
unsigned char uart_receive();
void clear_array(char arr[], uint8_t length, unsigned char c);
uint8_t calculate_checksum(const Frame *frame, int line);
void set_checksum(Frame *frame, uint8_t checksum);

#endif /* DISPLAY_UTILS_H */
```

B display_utils.c

```c
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//    1DT301, Computer Technology I
//    Date: 2017-12-07
//    Author:
//                       Caroline Nilsson            (cn222nd)
//                       Daniel Alm Grundström       (dg222dw)
//
//    Lab number:        6
//    Title:             CyberTech Wall Display
//
//    Hardware:          STK600, CPU ATmega2560, CyberTech Wall Display
//
//    Function:          Utility functions to help with interfacing with the
//                       CyberTech Wall Display
//
//    Input ports:       N/A
//
//    Output ports:      N/A
//
//    Functions:         init_serial_comm - Initializes USART
//                       create_frame - Creates and initializes a new Frame of type Information or Image
//                       send_frame - Sends a Frame through the serial port to the display
//                       uart_transmit - Transmits a single byte to the display
//                       uart_receive - Receives a single byte from the PuTTY terminal
//                       set_checksum - Sets the checksum of a Frame
//                       calculate_checksum - Calculates the checksum of a Frame
//                       clear_array - Sets all positions of an array to a specified character
//
//    Included files:    avr/io.h
//                       stdio.h
//                       string.h
//
//    Other information: Uses USART1
//
//                       See header file 'display_utils.h' for constants and
//                       declarations of structs FrameType and Frame
//
//    Changes in program: 2017-12-07:
//                       Adds headers and comments.
//
//                       2017-10-31:
//                       Changes BAUD rate from 4800 to 2400
//
//                       2017-10-23:
//                       Updates USART initialization code.
//
//                       2017-10-17:
//                       Removes debug code.
//
//                       2017-10-10:
//                       Moves code from assignment 1.
//
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#include <stdio.h>      // snprintf
#include <string.h>     // strncpy

#include <avr/io.h>     // UBRR1H, UBRR1L, UCSR1A, UCSR1B, UCSR1C, UCSZ10, UDRE1, UDR1, RXC1

#define FOSC 1000000UL // Clock Speed (1 MHz)
#define BAUD 2400
#define BAUD_PRESCALE (FOSC/16/BAUD-1)

#include "display_utils.h"

/*
 * Sets the USART transfer rate and enables specified usart control register
 * flags.
 */
void init_serial_comm(uint8_t ucsr1b_flags) {
    UBRR1H = (unsigned char)(BAUD_PRESCALE >> 8);
    UBRR1L = (unsigned char)BAUD_PRESCALE;
    UCSR1B = ucsr1b_flags;
    UCSR1C = (3 << UCSZ10); // asynchronous mode, 8-bit data length,
                            // no parity bit, 1 stop bit
}

/*
 * Creates an empty frame of the specified type.
 * Sets frame start, address, command and end.
 */
Frame create_frame(FrameType type) {
    Frame frame;
    clear_array(frame.line_1, INFO_FRAME_LINE_LEN, ' ');
    clear_array(frame.line_2, INFO_FRAME_LINE_LEN, ' ');
    clear_array(frame.command, INFO_FRAME_COMMAND_LEN, 0);

    frame.type = type;
    frame.start = 0x0D;
    frame.end = 0x0A;

    if (type == Information) {
        strncpy(frame.command, "O0001",
                INFO_FRAME_COMMAND_LEN);        // frame.command = "O0001"
    } else if (type == Image) {
        strncpy(frame.command, "D001",
                IMG_FRAME_COMMAND_LEN);         // frame.command = "D001"
        frame.address = 'Z';                    // Image frame address should always be 'Z'
    }
```

```
99
100     return frame;
101 }
102
103 /*
104  * Send a information/image frame to the display through the serial port.
105  *
106  * 'line' represents the position of the line on the display and determines how
107  * many lines to transmit. When 'line' is 1, both 'line_1' and 'line_2' is sent,
108  * otherwise only 'line_1' is sent.
109  */
110 void send_frame(const Frame *frame, int line) {
111     uart_transmit((unsigned char)frame->start);
112     uart_transmit((unsigned char)frame->address);
113
114     for (uint8_t i = 0; i < INFO_FRAME_COMMAND_LEN; i++) {
115         if (frame->command[i] != 0) {
116             uart_transmit((unsigned char)frame->command[i]);
117         }
118     }
119
120     if (frame->type == Information) {
121         for (uint8_t i = 0; i < INFO_FRAME_LINE_LEN; i++) {
122             uart_transmit((unsigned char)frame->line_1[i]);
123         }
124
125         if (line == 1) {
126             for (uint8_t i = 0; i < INFO_FRAME_LINE_LEN; i++) {
127                 uart_transmit((unsigned char)frame->line_2[i]);
128             }
129         }
130     }
131
132     for (uint8_t i = 0; i < FRAME_CHECKSUM_LEN; i++) {
133         uart_transmit((unsigned char)frame->checksum[i]);
134     }
135
136     uart_transmit((unsigned char)frame->end);
137 }
138
139 /*
140  * Sends a byte of data through the serial port (USART).
141  */
142 void uart_transmit(unsigned char data) {
143     // Wait until transmit buffer is clear
144     while (! (UCSR1A & (1 << UDRE1))) {
145         ;
146     }
147
148     UDR1 = data; // transmit data
149 }
150
151 /*
152  * Reads a byte from the serial port (USART).
153  */
154 unsigned char uart_receive() {
155     // Wait until data received flag set
156     while ( !(UCSR1A & (1 << RXC1))) {
157         ;
158     }
159
160     return UDR1;
161 }
162
163 /*
164  * Sets all elements of a specified array to 'c'.
165  */
166 void clear_array(char arr[], uint8_t length, unsigned char c) {
167     for (uint8_t i = 0; i < length; i++) {
168         arr[i] = c;
169     }
170 }
171
172 /*
173  * Calculates the checksum of a frame using the formula:
174  *
175  * checksum = sum(start, address, command, [message]) mod 256
176  *
177  * 'line' represents the position of the line on the display and determines how
178  * many lines to include in the checksum calculation. When 'line' is 1, both
179  * 'line_1' and 'line_2' is calculated, otherwise only 'line_1' is calculated.
180  */
181 uint8_t calculate_checksum(const Frame *frame, int line) {
182     uint8_t sum = frame->start + (uint8_t)frame->address;
183
184     for (int8_t i = 0; i < INFO_FRAME_COMMAND_LEN; i++) {
185         sum = sum + (uint8_t)frame->command[i];
186     }
187
188     if (frame->type == Information) {
189         for (int8_t i = 0; i < INFO_FRAME_LINE_LEN; i++) {
190             sum = sum + (uint8_t)frame->line_1[i];
191
192             if (line == 1) {
193                 sum = sum + (uint8_t)frame->line_2[i];
194             }
195         }
196     }
197
198     return sum;
199 }
```

```
200
201 /*
202  * Translates the specified checksum to a hex string (E.g. 63 -> '3F') and
203  * sets the specified frame's checksum to the translated checksum.
204  */
205 void set_checksum(Frame *frame, uint8_t checksum) {
206     char buffer[FRAME_CHECKSUM_LEN + 1];
207     snprintf(buffer, FRAME_CHECKSUM_LEN + 1, "%02X", checksum);
208     frame->checksum[0] = buffer[0];
209     frame->checksum[1] = buffer[1];
210 }
```