ME 586: Lab 5 Report
Mixed-Language Programming

Harsh Savla & TJ Wiegman

2022-10-21

Abstract

In this lab, we created assembly and C programs for the STM32 microcontroller and tested them on a STM32F100RB. These programs aimed to test integration of C functions into assembly programs, as well as integration of assembly routines into C programs. Serial input, serial output, and digital output were all tested in both languages.

1 Lab Checkpoints

1.1 Using a C function in an assembly program

Overview

The first task was to create a function in C that replicated an assembly program that was written for a previous lab.

Procedure

This section tested the `counts` function given in Appendix A.2.1 with flow chart given in Figure 4. The code was compiled and flashed to the STM32F100RB board using Keil µVision, and the program's behavior was validated using a PC serial terminal.

Results and Discussion

This function performed much as expected. A sample serial exchange is shown in Figure 1. Thankfully, minimal code changes were necessary in order to make the function compile and run correctly, so there are no remarkable notes on the debugging process.

1.2 Using assembly routines in a C program

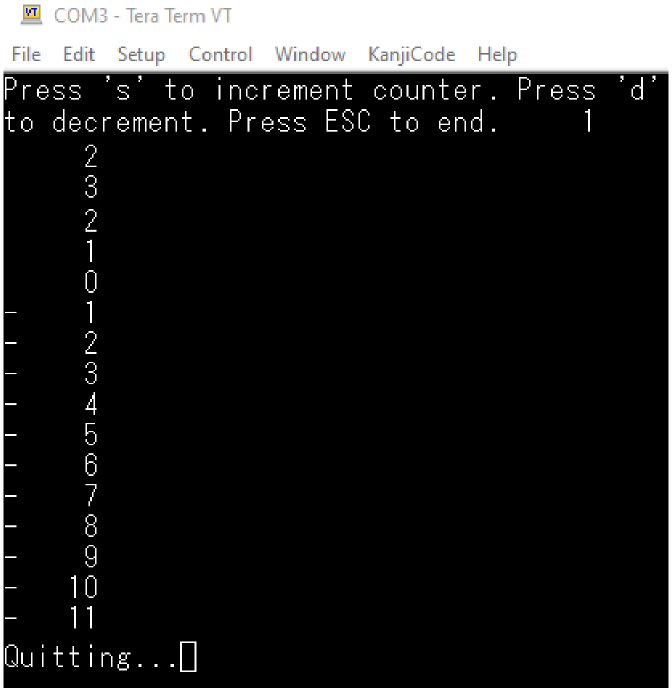
Overview

The next task was to compare and comment on how the assembly code generated by the assembler is different from the hand-written code used in Lab 3. Additionally, a C program to take serial input and output it to digital pins was created using both C functions and assembly subroutines.

Procedure

First, the memory map was compared between this program, written in C, and an equivalent that was written directly in assembly for a previous lab. The total memory usage for each was recorded in Table 1.

Figure 1: Serial terminal showing the input to and output from the `counts` program given in Appendix A.2.1.



```
Press ’s’ to increment counter. Press ’d’
to decrement. Press ESC to end.     1
     2
     3
     2
     1
     0
-    1
-    2
-    3
-    4
-    5
-    6
-    7
-    8
-    9
-   10
-   11
Quitting...
```

Table 1: Compiler optimizes C code for smaller memory usage than hand-written assembly code.

	Assembly	**C**
Total RO + Code Data	1.15 kB	1.09 kB
Total RW Data + ZI Data	1.53 kB	1.53 kB
Total ROM (Code + RO + RW)	1.18 kB	1.11 kB

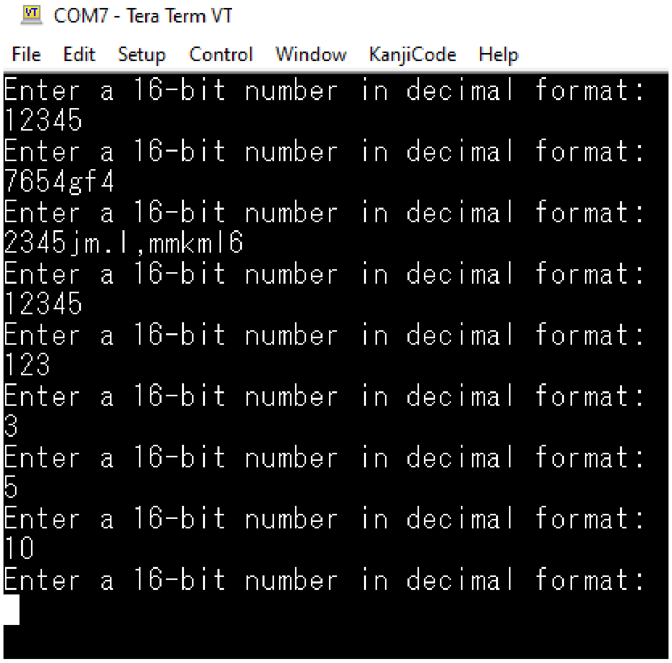
Afterwards, this section tested the `digout` program given in Appendix A.2.2 with flow chart given in Figure 5. The code was compiled and flashed to the board using Keil µVision. Sample data values were injected using a PC serial terminal and the output was compared with the expected behavior.

Results and Discussion

As shown in Table 1, the compiled C program took up less space than the hand-written assembly program. Examining the disassembled C code showed that the compiler made much heavier use of loops and branching to minimize code size.

For the following task, testing `digout`, the compiler initially threw some errors that were difficult to understand. After some debugging, it was discovered that this compiler requires any variables to be declared at the beginning of a new C function. Additionally, it was noticed that the function `initports` had not been called, which meant that the GPIO pins were not properly initialized before attempting to use them. After rectifying those errors, the program ran as expected. A sample serial exchange with the corrected program is shown in Figure 2. Interestingly, the `getanum` function (as given in Appendix A.2.3) ignores any non-numeric inputs, so an input such as 123abc45 would be interpreted as simply ”12345”, even though the alphabetic characters are still shown in the serial terminal.

Figure 2: Serial terminal showing the input to and output from the `digout` program given in Appendix A.2.2.



```
Enter a 16-bit number in decimal format:
12345
Enter a 16-bit number in decimal format:
7654gf4
Enter a 16-bit number in decimal format:
2345jm.l,mmkml6
Enter a 16-bit number in decimal format:
12345
Enter a 16-bit number in decimal format:
123
Enter a 16-bit number in decimal format:
3
Enter a 16-bit number in decimal format:
5
Enter a 16-bit number in decimal format:
10
Enter a 16-bit number in decimal format:
```

1.3 Serial input and digital output entirely in C

Overview

The final task was to use the various functions included in the C libraries to perform the same serial to digital data transmission that was done in the previous checkpoint.

Procedure

Assembly routines were removed from the project, and equivalents from the ME586 C library were used instead. Sample data values were injected using a PC serial terminal and the output was compared with the expected behavior.

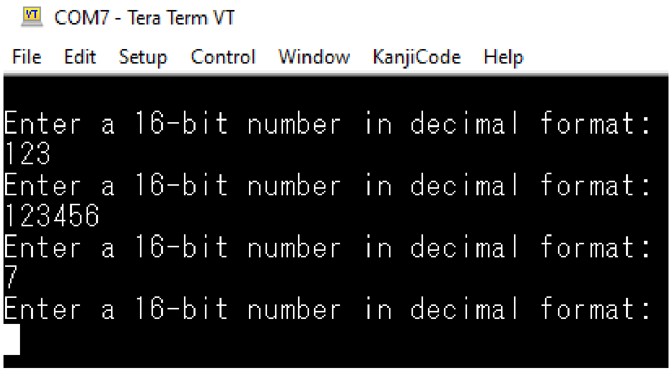
Results and Discussion

The code ran flawlessly and results matched expectations, as shown in Figure 3.

2 Conclusion

This lab allowed us to practice the fundamentals of mixed-language programming. We learned how to use C functions in assembly programs and assembly routines in C programs, and it provided a good transition from assembly-focused development to C-focused development. Being able to write in C has proven quite useful in that it allows much easier and more readable implementation of complicated control flow structures than assembly's branching system. One ”best practice” we learned in this lab was that including newline and carriage return characters after each serial output made the resulting serial exchanges much easier to read and understand.

Figure 3: Serial terminal showing the input to and output from the `digout` program given in Appendix A.2.4.



```
Enter a 16-bit number in decimal format:
123
Enter a 16-bit number in decimal format:
123456
Enter a 16-bit number in decimal format:
7
Enter a 16-bit number in decimal format:
```

A Appendix

A.1 Flow Charts

Figure 4: Flowchart for the `counts` function given in Appendix A.2.1.

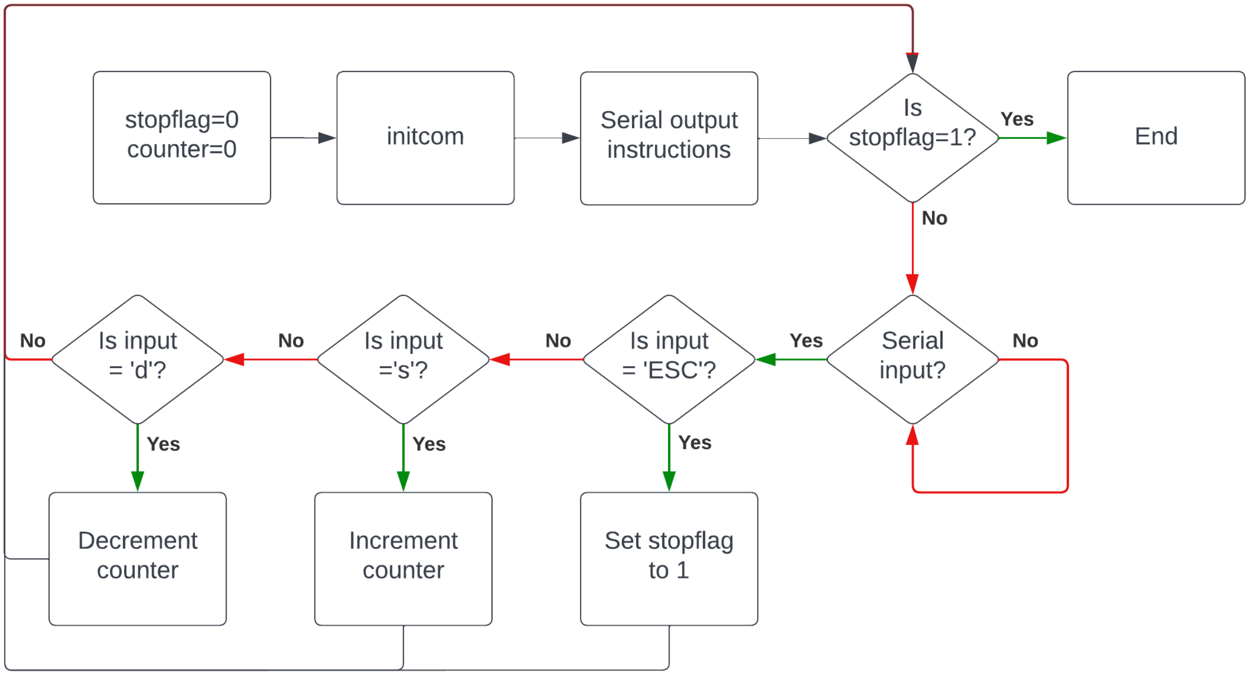
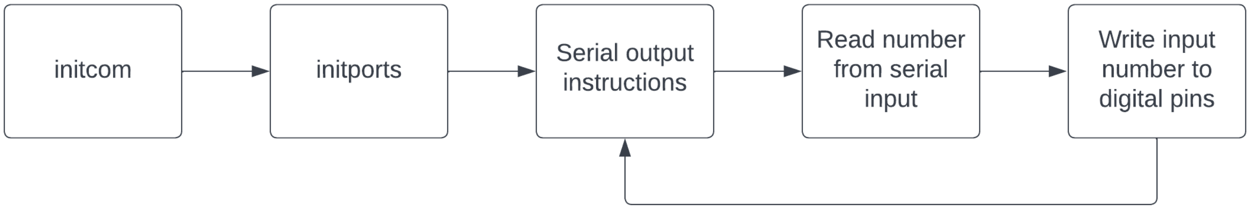


Figure 5: Flowchart for the `digout` function given in Appendices A.2.2 and A.2.4.


initcom
initports
Serial output instructions
Read number from serial input
Write input number to digital pins

A.2 Code

A.2.1 counts.c

```c
/**
  ******************************************************************************
  * @file counts.c for ME586 Homework 5
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-10
  * @brief Keycount program for HW 5 question 2
  ******************************************************************************
  */

extern void showmsg(char*);
extern void initcom();
extern char checkcom();
extern char getchar();
extern void shownum(short);
void keycount() {
  // setup
  char stopflag = 0;
  char readyflag = 0x00;
  char input = 0x00;
  short int counter = 0; // counter is a 16 bit number
  initcom();
  showmsg("Press 's' to increment counter. Press 'd' to decrement. Press ESC to end.");

  // loop
  while (stopflag != 1) {
    // wait to receive serial input
    while (readyflag != 0xFF) {
      readyflag = checkcom();
    }
    readyflag = 0x00; // reset flag for next loop

    // get input and react accordingly
    input = getchar();
    if (input == 0x1B) { // if receive 'ESC'
      stopflag = 1;
           showmsg("Quitting...");
    } else if (input == 0x73) { // if receive 's'
      counter = counter + 1;
      shownum(counter);
    } else if (input == 0x64) { // if receive 'd'
      counter = counter - 1;
      shownum(counter);
    }
  }
}
```

A.2.2 digout.c

```c
/**
  ******************************************************************************
  * @file digout.c for ME586 Homework 5
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-10
  * @brief Main program body for HW 5 question 4
  ******************************************************************************
  */

#include "ME586.h"

void problem4() {
    short input;
    //char output;
  initcom();
    initports(0xF00);
  while (1) {
    showmsg("Enter a 16-bit number in decimal format:");
        putchar(0x0D);
        putchar(0x0A);
    input = getanum();
        putchar(0x0D);
        putchar(0x0A);
    digout(input);
  }
}
```

A.2.3 utility.c

```c
/**
  ******************************************************************************
  * @file utility.c for ME586 Homework 5
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-10
  * @brief Utility programs for HW 5 question 1 & 3
  ******************************************************************************
  */

extern void initcom();
extern char checkcom();
extern char getchar();
extern void showchar(char); // prototype for assembly routine
void showmsg(char* msg) {
  int i = 0;
  int stopflag = 0;
  while (stopflag != 1) {
    char outchar = *(msg+i);
    if (outchar == 0x00) { // check for null ASCII to end
      stopflag = 1;
    } else { // otherwise send character
      showchar(outchar);
      i = i+1;
    }
  }
}

```

```
short sci4(char base, char power10) {
  // Input number in scientific notation (maximum 10^4)
  // Output is number in decimal form
  short output = 0;
  if (power10 == 0) {
    output = base;
  } else if (power10 == 1) {
    output = base*10;
  } else if (power10 == 2) {
    output = base*100;
  } else if (power10 == 3) {
    output = base*1000;
  } else if (power10 == 4) {
    output = base*10000;
  }
  return(output);
}

short getanum() {
  // setup
  char stopflag = 0;
  char readyflag = 0x00;
  char input = 0x00;

  // default input
  char num[5] = {0,0,0,0,0};
  char digits = 0;
    short output = 0;
    char i = 0;
    initcom();

  // input loop
  while (stopflag != 1) {
    // wait to receive serial input
    while (readyflag != 0xFF) {
      readyflag = checkcom();
    }
    readyflag = 0x00; // reset flag for next loop

    // get input and react accordingly
    input = getchar();
    if ((input >= 0x30) && (input <= 0x39)) { // if input is a number
      num[digits] = input - 0x30; // convert from ASCII to decimal
      digits = digits + 1;
    }
    if ((input == 0x0D) || (digits >= 4)) { // if input is CR or 5th digit
      stopflag = 1;
    }
  }

  // convert num array to a single short

  for (i = 0; i < digits; i++) {
    output = output + sci4(num[i], digits - i);
  }
  return(output);
}
```

A.2.4 digout2.c

```c
/**
  ******************************************************************************
  * @file digout2.c for ME586 Lab 5
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-11
  * @brief Main program body for HW 5 question 4
  ******************************************************************************
  */

#include "ME586.h"

void problem4() {
    short input;
    //char output;
  initcom();
    initports(0xF00);
  while (1) {
    printf("Enter a 16-bit number in decimal format:");
        putchar(0x0D);
        putchar(0x0A);
    input = getnum();
        putchar(0x0D);
        putchar(0x0A);
    digout(input);
  }
}
```