ENGI E1102 Departmental Project Report: Computer Science/Computer Engineering

Stephen A. Edwards

September, 2013

Abstract

This template attempts to illustrate both how to structure and write the final report for ENGI E1112 as well as serve as a very basic tutorial for the LaTeX typesetting system.

An abstract like this should summarize a document in a way that just about anybody can understand. It should be roughly 200 words of pure text and not contain images, undefined terms, or bibliographic references. Address why you did what you did, what you actually did, and, space permitting, how you did it.

My goal is to show you a document that is representative of good style in both English and LaTeX.

1 Introduction

This section should orient the reader to the document. Unlike the abstract, which should stand on its own, the introduction is an integrated part of the document and may contain figures, references to later sections, and bibliographic references.

Most paragraphs in technical writing follow the "topic sentence style" I am using for this paragraph. The initial sentence summarizes the main point of the paragraph; the remaining sentences provide additional details. Following this structure should make it easy to skim the document by reading the first sentence of every paragraph.

Be concise. This is the central point of Strunk and White [3], perhaps the best English style guide ever. I think of technical writing as an engineering challenge: how to clearly express ideas using as few words as possible. You will never be paid (or graded) by the word in technical writing, so be conservative. Your readers will thank you.

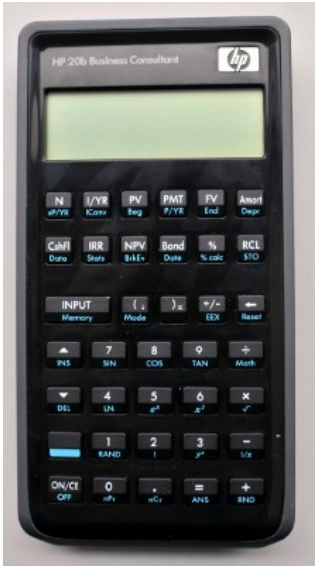

Figure 1: The HP 20b

2 User Guide

You built a system that people are intended to use; tell them how to use it in this section. Explain your calculator's display and keyboard, and how to use them to perform computations. Don't include any details about how you implemented these things; those are for later sections.

Follow a tutorial style: use plenty of examples, start simply and end with more complex examples. Feel free to look over another calculator's instruction manual for inspiration, but be sure to cite everything you use heavily.

3 The Platform

In this section, describe the hardware platform that you used to implement your project—the HP 20b calculator. The first lab writeup describes some aspects of it; more information can be found online.

I first learned about hacking the HP 20b (Figure 1) from de Brébisson's article in *Circuit Cellar* [1], and eventually took some more information from the HP *20b repurposing project* website [2].

3.1 The Processor

The HP 20b is little more than a keyboard and liquid crystal display (LCD) connected to an Atmel AT91SAM7L128 processor. This mouthful of a name, which I

will abbreviate to SAM7L, was given to it because it is part of Atmel's AT91SAM series of chips, which are all built around an ARM processor core ("AT" is for Atmel; "SAM" is "smart ARM core;" 91 appears to be arbitrary). The 7L series of microcontrollers are designed for low power (hence the L), and the final 128 is a reminder that it includes 128K of flash program memory.

Figure 2 shows a block diagram of the SAM7L chip. It looks complicated, but is essentially a single standard processor surrounded by memory and a wide variety of peripherals, most of which we will not use. You should be aware of the system controller, which, through software, controls the clock and power supply of each peripheral. This makes it possible to save energy by not powering on unneeded peripherals, but can also make a peripheral appear not to work if you neglect to turn on its power.

3.2 The LCD Display

Describe the layout of the LCD. The HP 20b SDK, which you can find on HP's website, includes a technical drawing of the LCD if you want to include it. Here would be a good place to describe some of the library functions I provided for you (e.g., *lcd_put_char7*).

3.3 The Keyboard

We tore apart some keyboards earlier in the term; here would be a good place to describe what you found along with details of the HP 20b's keyboard.

4 Software Architecture

In this section, explain how the various pieces of software in your system work together. Describe what they do, but not how each piece works; leave that for the next section.

5 Software Details

In this section, include cleaned-up listings of every bit of code you wrote for this lab (only include the declarations for the library code I wrote) and explain how they work, your motivations for them, whether you would do anything differently, etc.

5.1 Lab 1: Display a Number

In this section, describe what you were asked to implement for this lab (display a single integer) and how you implemented it.

Below, I describe my own code for a different problem (making a scrolling display); use it as a starting point for describing your own solution.

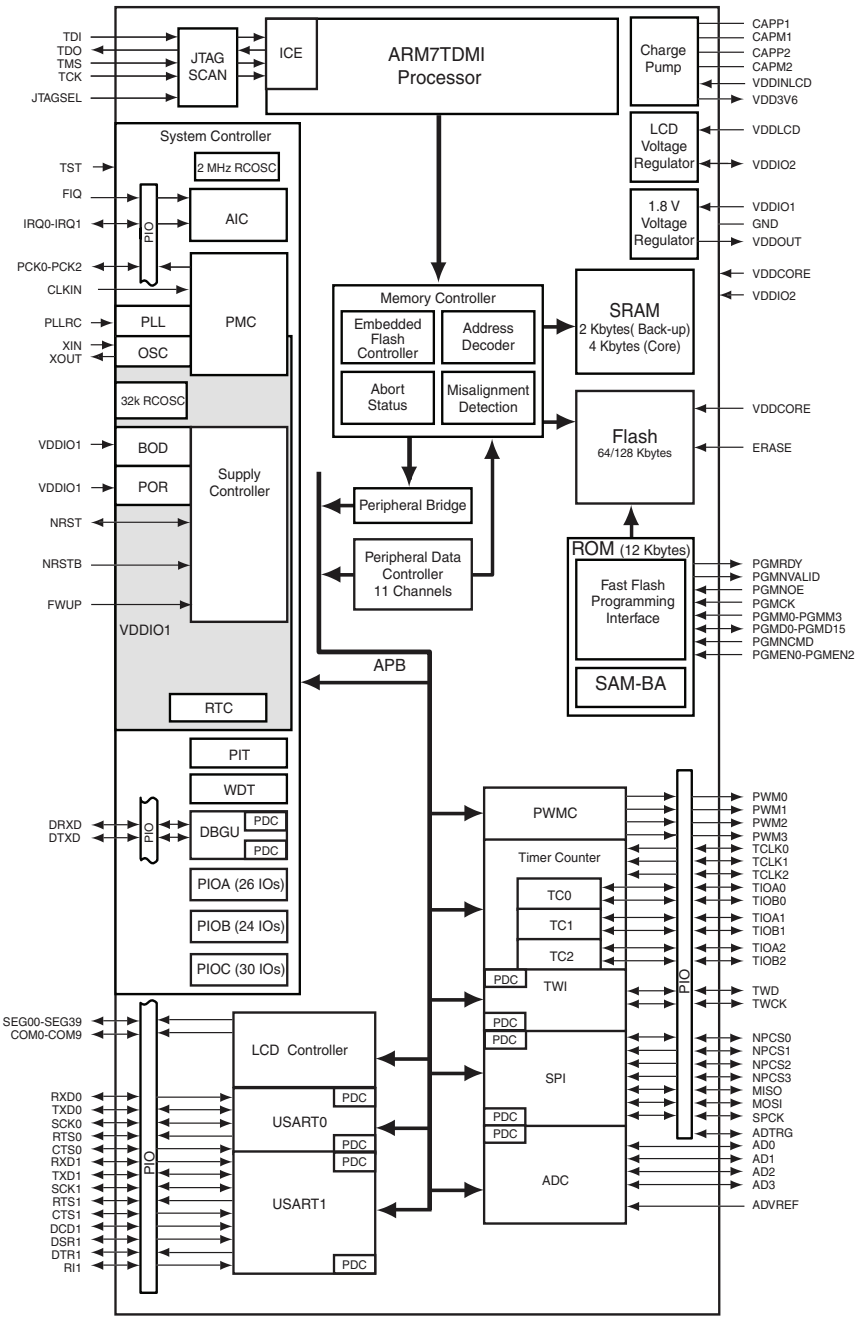

Figure 2: A block diagram of the AT91SAM7L microcontroller that is at the heart of the HP 20b

```c
#include "AT91SAM7L128.h"
#include "lcd.h"

#define DELAY 50000
#define COLUMNS 12

int main()
{
  char message[] = "CS AND CE ARE FUN ";
  char *start, *cptr;
  int col, i;

  lcd_init();
  // Turn off the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  for (;;) {
    for (start = message ; *start != '\0' ; start++) {
      cptr = start;
      for (col = 0 ; col < COLUMNS ; col++) {
        lcd_put_char7(*cptr++, col);
        if (*cptr == '\0') cptr = message; // wrap around
      }
      for (i = 0 ; i < DELAY ; i++) {}
    }
  }
  return 0;
}
```

Figure 3: My code for displaying a scrolling message

Figure 3 shows my code for a scrolling display. It consists of two loops inside an infinite loop. The innermost loop steps through each character of the display, fetching characters from the message string (the *cptr* pointer) and wrapping around to the first character of the message. The wrap-around makes the message restart after it ends, e.g., the display may read "ARE FUN CS AND." When this loop terminates, a do-nothing *for* inserts a delay before the display is redrawn to make it scroll at a reasonable speed.

The middle loop steps the *start* pointer through each character of the message, starting *cptr* at the beginning of the message, the second character, the third, etc. When this loop terminates, the outer *for* loop restarts it.

5.2 Lab 2: Scanning the Keyboard

Describe how you wrote the software that scanned the keyboard. Mention how you actually did the scanning, how you encoded the result, and how you tested it.

5.3 Lab 3: Entering and Displaying Numbers

Describe the structure of your software for entering numbers. Explain the interface to your function (i.e., how it is called and how to interpret what it returns), how you chose to represent intermediate results (e.g., did you keep an integer as the value and convert it for display, or did you store the results as a string and only convert it to an integer when you returned it?), and the various corner cases (e.g., what happens when someone presses $+/-$ without having entered any digits).

5.4 Lab 4: An RPN Calculator

Explain how you used the pieces you described in Sections 5.2 and 5.3 to create a real calculator. You should have explained how RPN works in Section 2; here you should explain how you implemented your stack and how your software interprets commands.

6 Lessons Learned

What did you learn from this lab? What advice do you have for future students? What do you know now that you wish you were told at the beginning of the class?

Using LaTeX

Get and install one of the many TeX distributions:

- under Windows, use MiKTeX;
- for Mac OS X, MacTeX may be the easiest; and

- for Linux, TEX Live is the way to go. Under Ubuntu, *sudo apt-get install texlive-latex-base* will install most of what you need.

Writing LATEX is a bit like writing a program. Once you've edited your *report.tex* and *report.bib* files, which contain text and bibliographic data, run *pdflatex* to generate a PDF file from them. Because of the way references are handled (e.g., to the bibliography and figures), you may actually have to run four programs (Figure 4). The *Makefile* I included with this report does this for you automatically; *make* should rebuild if anything changes.

References

[1] Cyrille de Brébisson. How to repurpose a development platform. *Circuit Cellar*, 232:44–49, November 2009.

[2] Hp-20b repurposing project. Online `http://www.wiki4hp.com/doku.php?id=20b:repurposing_project`.

[3] William Strunk Jr. and E. B. White. *The Elements of Style*. Longman, fourth edition, 1999.

```
$ pdflatex report.tex
...
LaTeX Warning: There were undefined references.

LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
...
Output written on report.pdf (7 pages, 296486 bytes).
Transcript written on report.log.
$ bibtex report.aux
...
Database file #1: report.bib
$ pdflatex report.tex
...
LaTeX Warning: There were undefined references.

LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
...
Output written on report.pdf (7 pages, 296486 bytes).
Transcript written on report.log.
$ pdflatex report.tex
...
Output written on report.pdf (8 pages, 298715 bytes).
Transcript written on report.log.
```

Figure 4: Running LaTeX. It is sometimes necessary to run *pdflatex* three times to resolve all the references.