Repurposing an HP Calculator
Lab 1: Hello World
CS & CE Art of Engineering Project

Stephen A. Edwards

Fall 2015

Abstract

In this project, you will write new firmware for an HP 20b calculator. This is an example of embedded programming—coding software for something that does not appear to be a computer, yet is one at its core. The plummeting cost of integrated circuits has made such embedded systems ubiquitous. The challenges of designing such systems run the gamut from traditional electrical issues such as sensor noise and power consumption all the way to high-level computer science problems such as efficient algorithm design to human factors engineering. You will experience all of these, and learn some standard solutions, while performing this project.

1 Introduction

In 2008, over ten billion processor chips were manufactured and sold[1]—more than one for every human on earth. The heavily marketed high-end Intel and AMD processors residing within our desktop and notebook computers represent only about two percent of the total; the rest are so-called embedded processors residing in familiar objects such as cell phones, but many end up in less obvious places such as cars, televisions, Blu-ray players, and toys. Once, I even found a microprocessor in my breakfast cereal.[2] These things are everywhere; somebody has to program them.

In this project, you will do some embedded programming by creating new firmware for the HP 20b calculator (Figure 1). Unlike most consumer products, this one was "opened" by HP: they provide schematics and a software development kit, so it is fairly straightforward to turn this calculator into something it wasn't originally designed to be. I turned one into a remote control for third-world power distribution systems.[3]

[1]Michael Barr. Real men program in C. *Embedded System Design*, August 1st, 2009. http://www.embedded.com/columns/barrcode/218600142.

[2]Xbox 360 "mini games" in Apple Jacks cereal: http://www.youtube.com/watch?v=jUNsQFG_5Pk

[3]With Prof. Vijay Modi of the mechanical engineering department http://modi.mech.columbia.edu/.

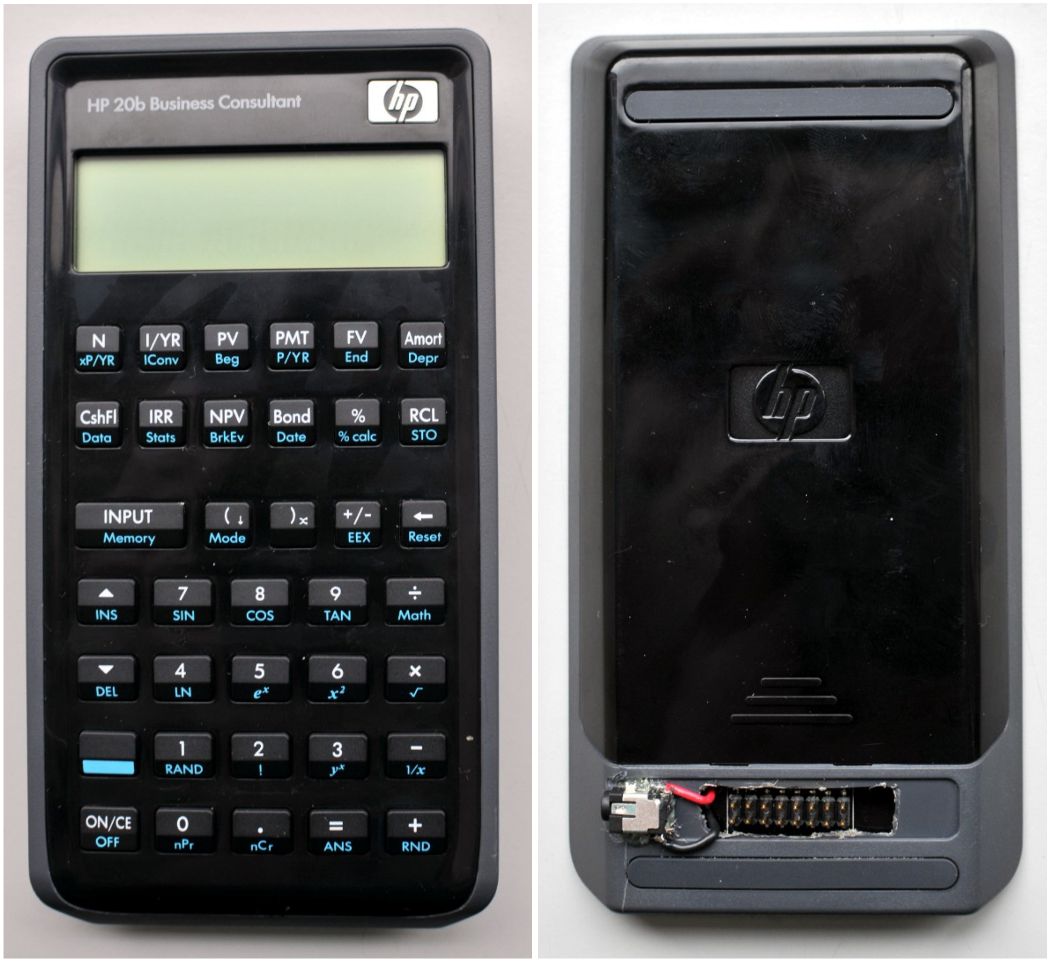

Figure 1: The front and back of the HP 20b calculator. I cut a hole and added the JTAG header so we could develop software on these.

2 The HP 20b

The HP 20b is essentially a keyboard and liquid crystal display (LCD) driven by 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 indicates 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.

For this project, the two most interesting peripherals are the LCD controller, which generates the complex AC waveforms necessary to drive the calculator's multiplexed LCD display; to software, the LCD appears as a series of memory locations whose bits control individual LCD segments.

3 What the heck is a JTAG?

We will communicate with the processor through a JTAG[4] port, which is built into the SAM7L. The 20b's circuit board has a convenient place to solder a connector that bring out the JTAG signals, which I have already done for you.

Our development environment consists of Windows workstations with JTAG adapters connected to the USB ports. On the workstations, we are using a software package called "OpenOCD"[5] that communicates to the SAM7L CPU through USB and JTAG.

Figure 3 show how to assemble the hardware. Plug the 20-pin connector onto the calculator's JTAG connector, making sure to have the red wire (pin 1) on the left when you are looking at the back of the calculator. The calculator's connector has only 16 pins; make sure the JTAG connector extends past the pins *on only the right side*. The unconnected four pins do not do anything we care about.

[4] The Joint Test Action Group originally developed this protocol for testing printed circuit boards. Today, virtually every microcontroller has a JTAG port for development and debugging.

[5] Open on-chip debugger: http://openocd.berlios.de/web/

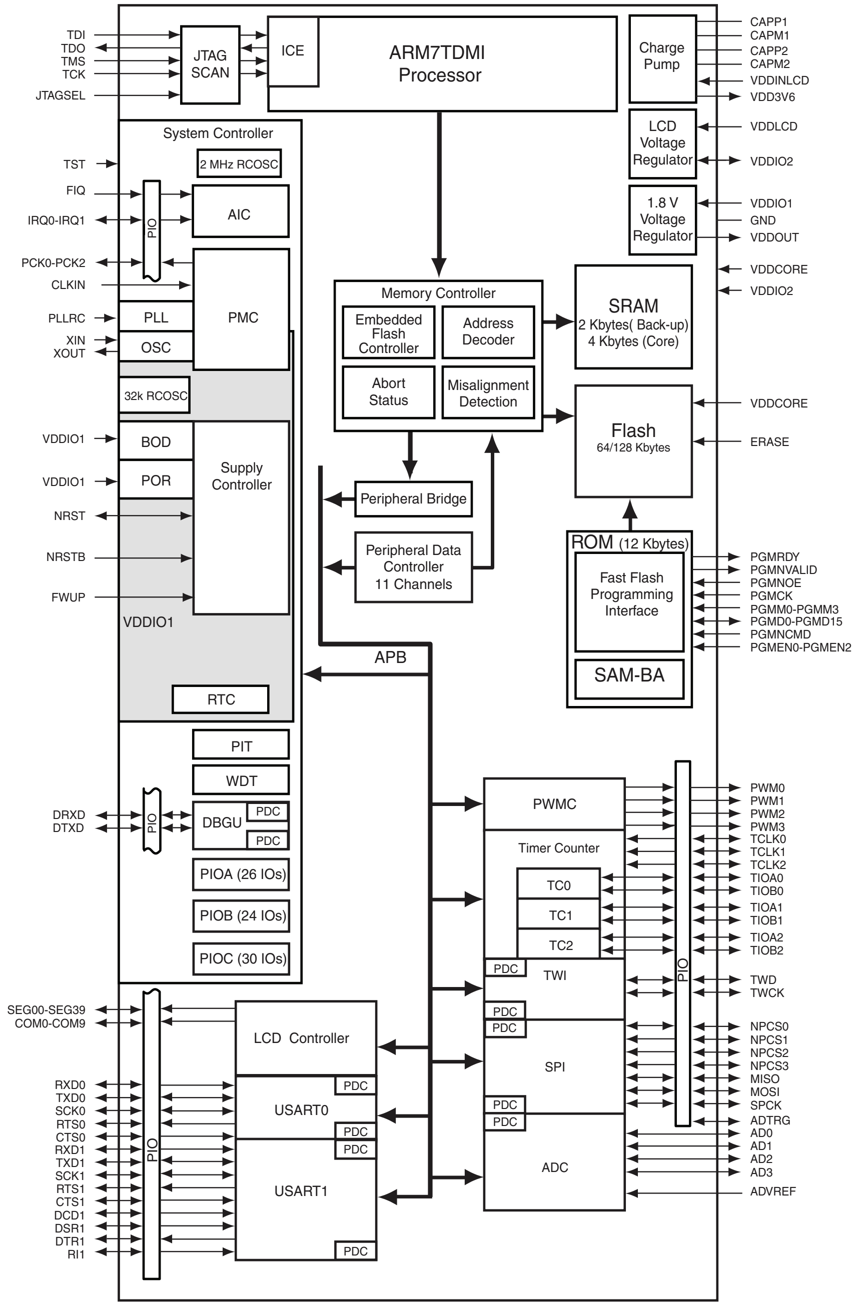

Figure 2: Block diagram of the AT91SAM7L chip. This consists of an ARM7TDMI processor core surrounded by memory, a system (clock) controller, an LCD controller, and a variety of other peripherals. Source: Atmel.

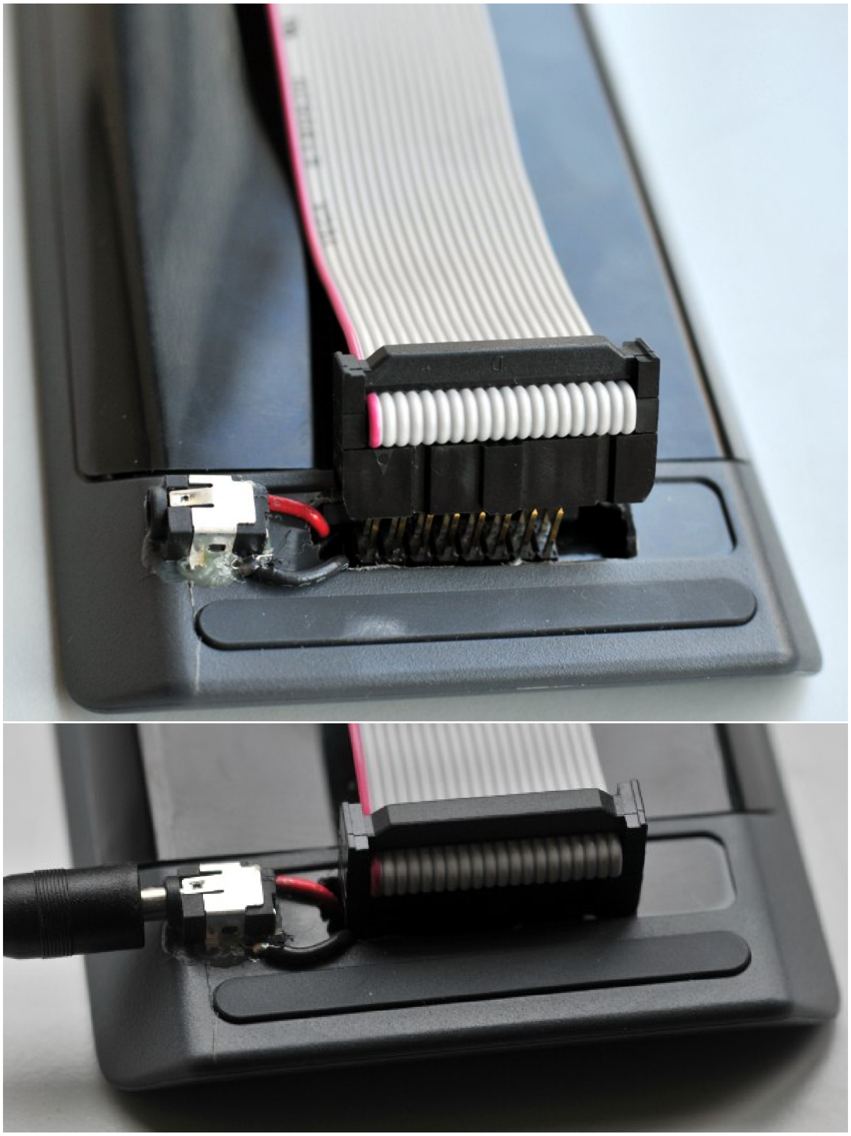

Figure 3: Connecting to the development ports on the back of the HP 20b calculator. Note that the JTAG connector extends beyond the header and we are not using the external power jack.

flash.bat	Windows batch file: compiles the program & flashes the 20b
flash-h.bat	Variant for the ARM-USB-TINY-H dongles
Makefile	Rules for compiling the program. Add source file names here.
main.c	The main program: initialize the LCD and display a string.
lcd.c	LCD-related functions and data.
lcd.h	Externally visible interface to *lcd.c*
AT91SAM7L128.h	Addresses of every peripheral in the SAM7L chip
crt0.S	ARM assembly code that initializes the C runtime environment
at91sam7l128.lds	Linker script: how and where to put the program in memory
flash.cfg	OpenOCD file: rules for writing to flash
hp-20b-calculator.cfg	Tells OpenOCD the HP 20b contains a SAM7L chip
at91sam7l128.cfg	Tells OpenOCD about the SAM7L chip

Figure 4: Files in *lab1.zip*

4 Compiling and Running "Hello"

Download the *lab1.zip* file from the class webpage and unpack it, e.g., in the *Downloads* folder. This contains all the rules and scripts to compile the program and download it to the calculator. See Figure 4 for a list of the files and what they do.

To compile the program and flash the calculator (load the program into the calculator), start a Windows command shell (press the Windows key and ender "cmd" followed by Enter).

In the command shell, change to the unpacked *lab1* directory and invoke the *flash* script:

```
C:\Users\sedwards> cd Downloads\lab1
C:\Users\sedwards\Downloads\lab1> flash
```

Many magic incantations are at work here, but broadly, this compiles the *crt0.S*, *main.c*, and *lcd.c* files into corresponding binary *.o* files, links them together to produce *main.elf*, transforms that into the loadable *main.hex* file, then downloads that to the calculator.

We have two varieties of JTAG dongles: ARM-USB-TINY and ARM-USB-TINY-H. They look almost identical but appear as different USB devices. If you are using the -H variant, run the flash-h script instead:

```
C:\Users\sedwards> cd Downloads\lab1
C:\Users\sedwards\Downloads\lab1> flash-h
```

If you see the following, the JTAG adapter either isn't properly connected to the workstation or you are using the -H variant but have it configured for the other one. Check that the JTAG adapter is attached to the USB cable, that the cable is plugged into the computer, and that you're running the right version of the *flash* script.

```
C:\Users\sedwards\Downloads\lab1> flash
...
Error: unable to open ftdi device: device not found
...
```

If you get the following error, the JTAG adapter wasn't able to establish communication with the SAM7L chip in the calculator.

```
C:\Users\sedwards\Downloads\lab1> flash
...
Error: JTAG scan chain interrogation failed: all ones
Error: Check JTAG interface, timings, target power, etc.
...
```

This error occurs if the calculator isn't properly connected to the JTAG adapter with the 20-pin ribbon cable, if the calculator doesn't have power (press the ON/CE button and check the battery), or if the SAM7L chip is sleeping.

If all goes well, your calculator should look like Figure 5 and you should see a lengthy series of commands and comments that looks roughly like

```
C:\Users\sedwards\Downloads\lab1> flash
...
Info : JTAG tap: at91sam7l128.cpu tap/device found:
    0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x3)
...
target halted in ARM state due to debug-request,
...
wrote 1120 bytes from file lab1.hex
...
shutdown command invoked
```

Here I've excerpted the relevant lines. The "JTAG tap" line indicates it was able to communicate with and establish the identity of the processor. The "target halted" means the OpenOCD tool was able to instruct the processor to halt so it could download a file to it, which the "wrote" message indicates. Finally, we disconnected OpenOCD, indicated by "shutdown command invoked."

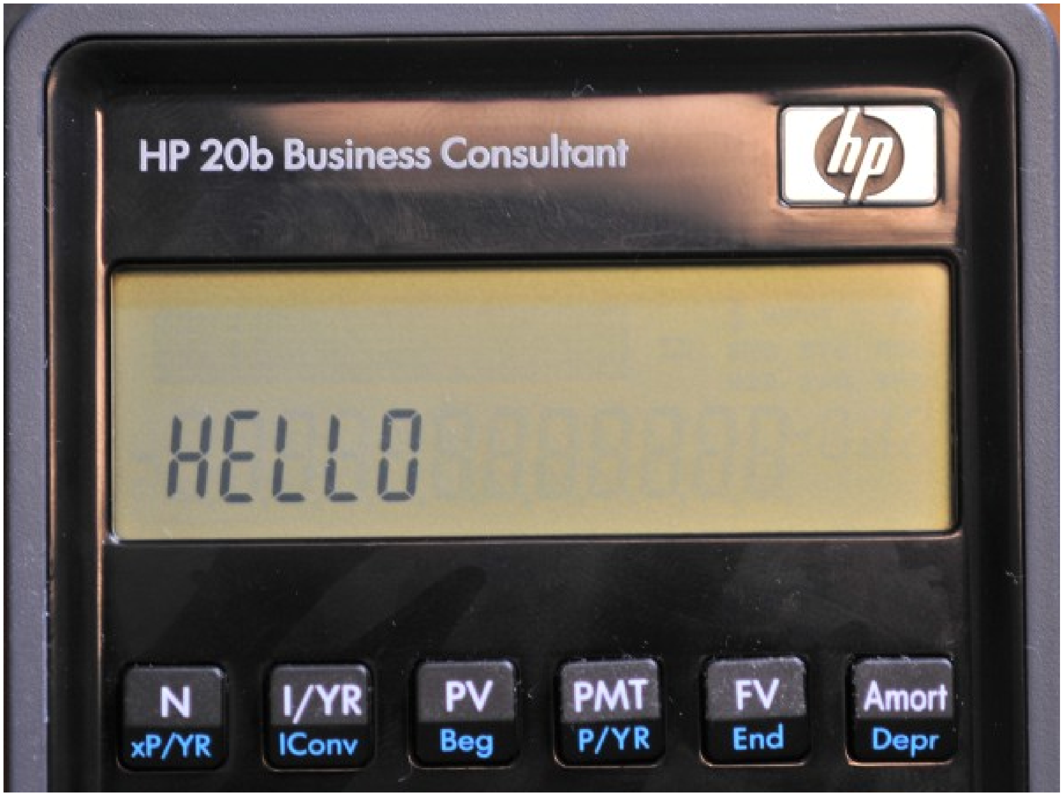

Figure 5: Running the “Hello” program

5 What To Do

1. Follow the instructions in the previous section to compile and download the sample program onto the calculator.

2. Edit *main.c* and create a function that takes an integer argument and displays it in decimal on the calculator. You have up to 12 digits to work with. Compile and download your program to the calculator and show us that it works. Test it with a variety of numbers, e.g., 0, −1, 1, 42, 12572, −123523.

3. When you're happy with your display-a-number function, make sure your names are in the main.c file and submit it via CourseWorks. Only one submission per group is necessary.