1 Introduction

This project report describes the motivation and decisions behind the design and implementation of a clock, as well as the results. It is a clock based on an ARM7 microcontroller (Atmel AT91SAM7S64) using a $128x128$ pixel full-color organic LED display (Univision UG-2828GFEFF01) as the output. The clock face is a raster image of an analog clock face and time is kept by a dedicated real-time clock IC (a Dallas DS3234S).

1.1 Motivation

The main reason this project is being undertaken is in order to implement (and then gain experience with) several of the components that will be part of a more ambitious project for PHYS476. When compared to this semester's clock project, next semester's project will use an identical organic LED display as well as the same family of microcontroller that uses the same syntax assembly language.

At the start of this project, the designer was inexperienced with designing DC-DC converters, designing printed circuit boards and was even inexperienced soldering surface-mount components. These three things are necessary parts of next semester's project, so it was decided to tackle them with this project.

2 Design criteria & decisions

2.1 Display

The display will be a full-color graphic organic LED module with integrated controller and memory. The Univision UG-2828GFEFF01 is a fairly low-cost display ($37 in single quantities), it has a simple parallel interface, has 18 bit color and is low power.

An organic LED display is well suited to battery-powered applications when compared to an equivalent LCD because an OLED display only powers the pixels that are lit (versus an LCD where all pixels are backlit and light is selectively blocked for some of them). This is only important for the the project for next semester, which will be battery-powered.

Another feature of OLED displays (and electroluminescent displays in general) is that since each pixel emits its own light, the display is viewable from any angle equally well.

When all pixels are illuminated (white), the Univision UG-2828GFEFF01 draws $22.48mA@14V$ which is $315mW$. This is $19.84\mu W$ per pixel.

While the datasheet says the display is 128 columns and 128 rows, it's only possible to illuminate 124 of the rows. It is unclear why this limitation exists.

2.2 Microcontroller

An ARM7 was chosen partly so that the designer could gain experience with an unfamiliar microcontroller that had higher performance than those from his previous experience (Motorola HC05, Zilog Z8). An Atmel AT91SAM7S64 can execute about 55 MIPS (although there's a mandatory wait state if running code from flash instead of from SRAM, so it's somewhere around 30 MIPS in that case), compared with about 1 or 2 MIPS for the HC05 or Z8.

This microcontroller has several features built-in that make this project simpler when compared to having to implement them manually. Specifically, it has an SPI (serial peripheral interface) port and a 10 bit A/D (analog-to-digital) converter. The included SPI port means the timing of the signals going between the RTC and the microcontroller is handled entirely by the microcontroller itself and no special software need be developed to directly manipulate the signals. Having an on-board A/D converter permits the ambient light measurement to require only a single external component; namely the three-terminal light-to-voltage sensor.

All of this makes an Atmel ARM7 a good choice for this project. However, the ARM7 lacks a divide instruction, which presents a challenge when trying to generate a sine/cosine lookup table.

2.3 Real-time clock

The choice to use an RTC (real-time clock) IC over a software clock was one of power requirements. An RTC can keep time given only a microamp or so, versus at least a couple of milliamps to get a microcontroller to do it.

The DS3234S has an integrated crystal and only needs a small external battery to keep time for years when main power is absent. It has a switched capacitor network so the capacitance that shunts the crystal's terminals can be altered to modify the frequency of oscillation to compensate for imperfect crystal trimming and crystal aging. In addition, it has a temperature sensor and it periodically measures its internal temperature and uses an internal lookup table to compensate for the temperature drift of the crystal-capacitor network's resonant frequency.

2.4 Light-to-voltage

In order to make the display's brightness proportional to the ambient light level in the room, a light-to-voltage converter is used. The result is that, in darkness, the display will be dimmed so as not to illuminate the otherwise dark room. Whereas in a bright room or in daylight, the display is brightened so that it is readable.

The TSL250 is a light sensor whose output voltage is proportional to the ambient light level. The maximum voltage output is $\approx V_{cc} - 1V = 3.3V - 1V = 2.3V$; some precision is lost because the upper $\frac{1}{3.3}$ of the input range is never used. However, this is immaterial because the OLED's controller only allows 16 levels of brightness, so only $log_2\left(16\frac{3.3V}{2.3V}\right) = 4 + log_2\left(\frac{3.3V}{2.3V}\right) = 4.52$ bits of A/D conversion are required and 10 are available.

2.5 Software tools

A relatively simple objective was chosen for this project so that it could be determined whether it was straightforward to get a development platform setup on linux to assemble, compile, link and program the ARM7 using freely available software. In this regard, it has turned out well.

Here is a list of the tools used in this project to get the prototype working:

- gnu assembler
- gnu c compiler
- gnu linker
- gnu make
- openocd (to program the device over jtag as well as for simple debugging)

And here's a list of tools used to design the final version of the project:

- Eagle (schematic and PCB design tools).

Except for Eagle, everything in these lists is open source/free software. Eagle is a free trial version of closed-source software whose functionality is limited in certain ways but it is still usable for this project.

All of the aforementioned software runs under linux, so one of the goals of the project is already complete.

2.6 Power

Another constraint on this project is that the components have been chosen partly because they are low-power. This is not a strict requirement for a wall-powered clock, but it is a requirement for any battery-powered device.

The organic led display consumes power proportional to how many pixels are illuminated (plus a small offset to power the controller). For a simple clock face, only about 5% of the pixels are lit and the display only consumes $3.86mA@14V + 0.83mA@3.3V = 56.8mW$.

The microcontroller draws power proportional to the main oscillator frequency (plus some small offset for other things). For the prototype, this frequency is $18.432MHz$ and it draws $12.5mA@3.3V = 41.3mW$. The final clock will run at a slower clock rate and the power draw will be proportionally less.

When main power is absent, the RTC keeps time via the $12mm$ lithium battery. According to the datasheet, it draws somewhere between $0.5\mu A$ and $2\mu A$, which is between $4.5mAh$ and $18mAh$ per year. For a $30mAh$ battery, it should keep time for a couple of years with the clock unplugged.

2.7 Cost

As the design for this clock has matured, the cost of the prototype has been kept in mind, as well as how much a production run would cost, per clock. The OLED display is by far the most expensive single component at $37, followed by the circuit board, the real-time clock and then the microcontroller. Table 1 is a list of the required components as a function of the cost for buying them in single quantities (but including data for a production run).

2.8 Schematic

The current version of the schematic is pictured in Figure 1.

table 1: How component cost varies with quantity purchased

part	cost per clock	
	single quantity	100
OLED display	$36.95	$29.56
printed circuit board	$17.30	$5.48
~~crystal~~ oscillator	$11.61	$7.74
RTC	$8.77	$4.68
microcontroller	$8.49	$4.94
DC-DC converters (2)	$6.71	$4.68
accelerometer	$6.10	$3.91
capacitors (25)	$3.39	$2.43
momentary pushbuttons (4)	$3.32	$2.40
usb connector	$2.19	$1.17
display connector	$1.22	$0.87
battery	$1.11	$1.11
resistors (23)	$1.02	$0.61
inductors	$0.86	$0.66
mosfet	$0.54	$0.31
battery holder	$0.53	$0.27
Zener diode	$0.39	$0.19
total	$110.50	$71.01

2.9 Prototype

The prototype is built on 0.1" spacing vectorboard. In order to speed up development of the prototype, it incorporates two third-party printed circuit boards (one for the microcontroller and one for the display). With these in place, the software could be written almost immediately to verify that the project's objectives could be fulfilled with the chosen hardware.

2.10 Printed circuit board

The PCB (Figure 2) has 0.1Ω resistors in series with the power supply lines (both before and after the DC-DC converters). So with a voltmeter, one can determine how much current is being drawn at any of the three power supply voltages ($3.3V$, $5V$, $17V$), as well as how efficiently the $5V$ input voltage is being converted to $3.3V$ and $17V$.

2.11 DC-DC converters

The clock is powered via USB (which is $\approx 5V$) and all the ICs run on $3.3V$, so a DC-DC converter is required. For efficiency, a linear regulator was ruled out and a LM3670 buck converter was chosen instead.

The display needs a higher voltage (between $7V$ and $18V$) to drive the leds, so another DC-DC converter was used to convert $5V$ to $17V$. A LT3461 boost converter with integrated schottky diode was chosen to accomplish this.

2.12 Extra features

Several features were added to the printed circuit board in the hopes that this project will be capable of doing more than just being a simple clock.

First, a resistor/capacitor network was added to enable the PLL in the microcontroller so that it can run at $> 20MHz$ (which is the limit if it just uses an external crystal). The clock software runs well at $\frac{18.432MHz}{64} = 288kHz$, so something greater than 20MHz is overkill. But in the case of the successor project, it is desirable to have a dynamic clock rate, to allow for both quick processing power in a short time when needed as well as long battery life.

Second, the data part of the usb port is wired and series resistors are included as well as a p-channel FET (field-effect transistor) and pull-up resistor. None of these components are necessary to have it behave as a simple clock, but if appropriate software is written, the clock can be set from a network time protocol server via a personal computer's usb port.

Third, a three-axis accelerometer is included so that the device's orientation relative to the Earth's gravitational field can be determined. This is primarily for mode selection; tip the clock on its side to put it in a calendar mode (for instance). Since the display controller allows mirroring in both the horizontal and the vertical, only a minimal software change will be required to get the clock face to still be upright after being tipped over.

2.13 Software

The software for the microcontroller is written almost entirely in assembly language. Writing everything in assembly language is partly for efficiency. The output of the c compiler for a simple function was inspected and found to be too inefficient to use. Another reason for writing the entire clock program in assembly language is programmer familiarity. The programmer is used to setting bits using assembly language and is not as comfortable doing so in c.

Programming of the device's flash memory is done over a JTAG (joint test action group) connection to a personal computer.

3 Conclusion

The goals set out at the beginning of the start of work on this project have been met. It was found that it is possible to develop c or assembly programs, compile or assemble them and then link them and program them to the device's flash memory easily in a freely available development environment. A printed circuit board was built and populated and then programmed and tested and functions well. The ARM7 microcontroller uses a modest quantity of power to run the clock program. The display was found to be adequate for showing text and graphics while using a reasonable amount of power to do so. The clock keeps time while main power is not present. The DC-DC converters behaved as expected: $\approx 5V$ went in and $3.31V$ came out from one and $16.35V$ came out from the other without any noticable ripple on either.

All (or at least most) of these things can be directly applied to the project for next semester, which is to be a battery powered MP3 player which decodes the compressed files in software. Only time will tell whether that project turns out as well as this one.

3.1 Addendum

The software written for this project is available at http://arm7-oled-clock.googlecode.com/svn/trunk/. It is licensed under the GNU General Public License v3 or higher. More information is available at http://fsf.org.

The schematic and circuit board design will soon be available at http://arm7-oled-clock.googlecode.com/. These are licensed under the Creative Commons attribution non-commercial share-alike v3.0 license. More information is available at http://creativecommons.org.

Photographs and more information about this project are availble at http://ohmslog.wordpress.com.

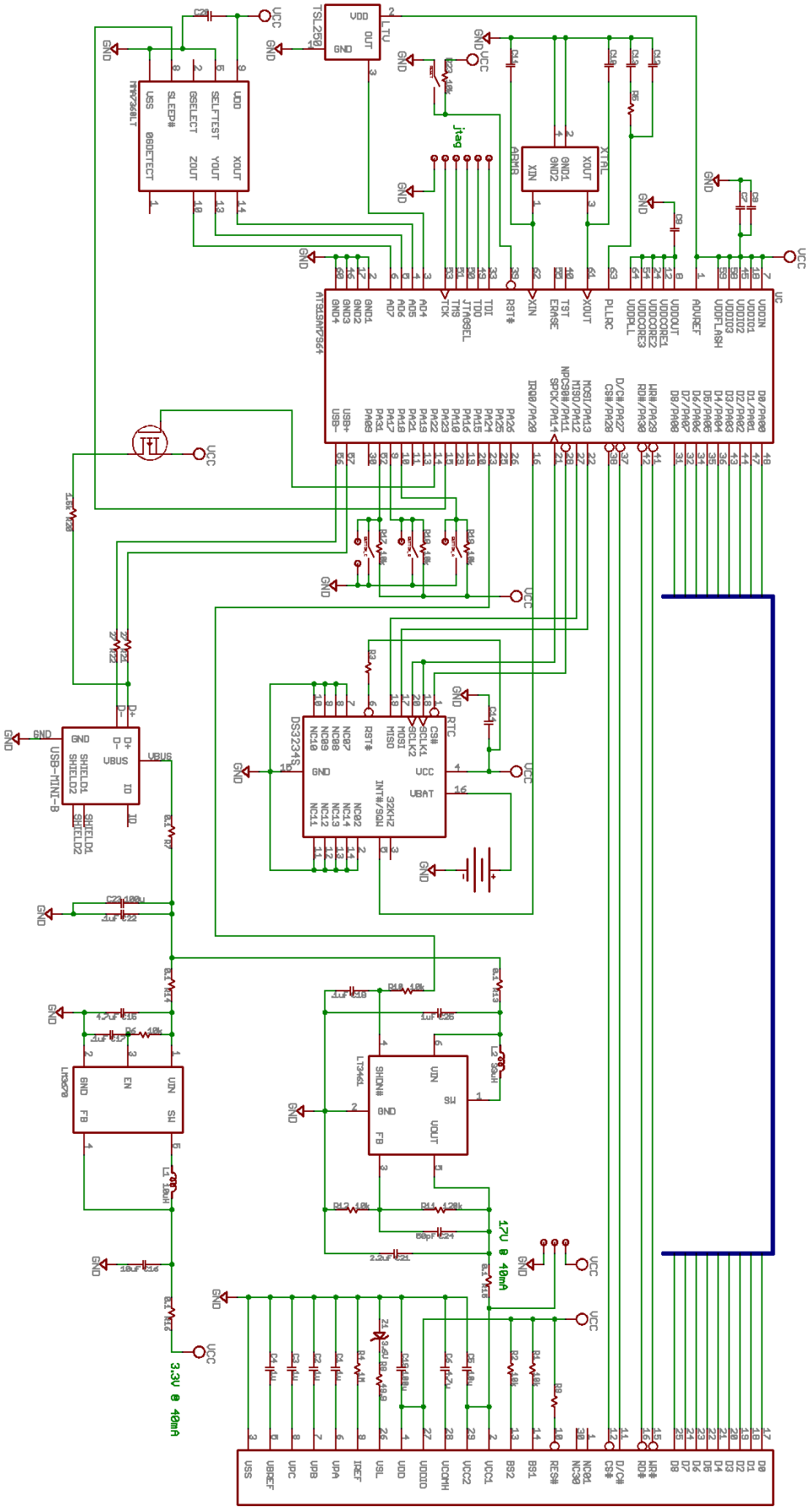

Figure 1: Schematic for arm7-oled-clock

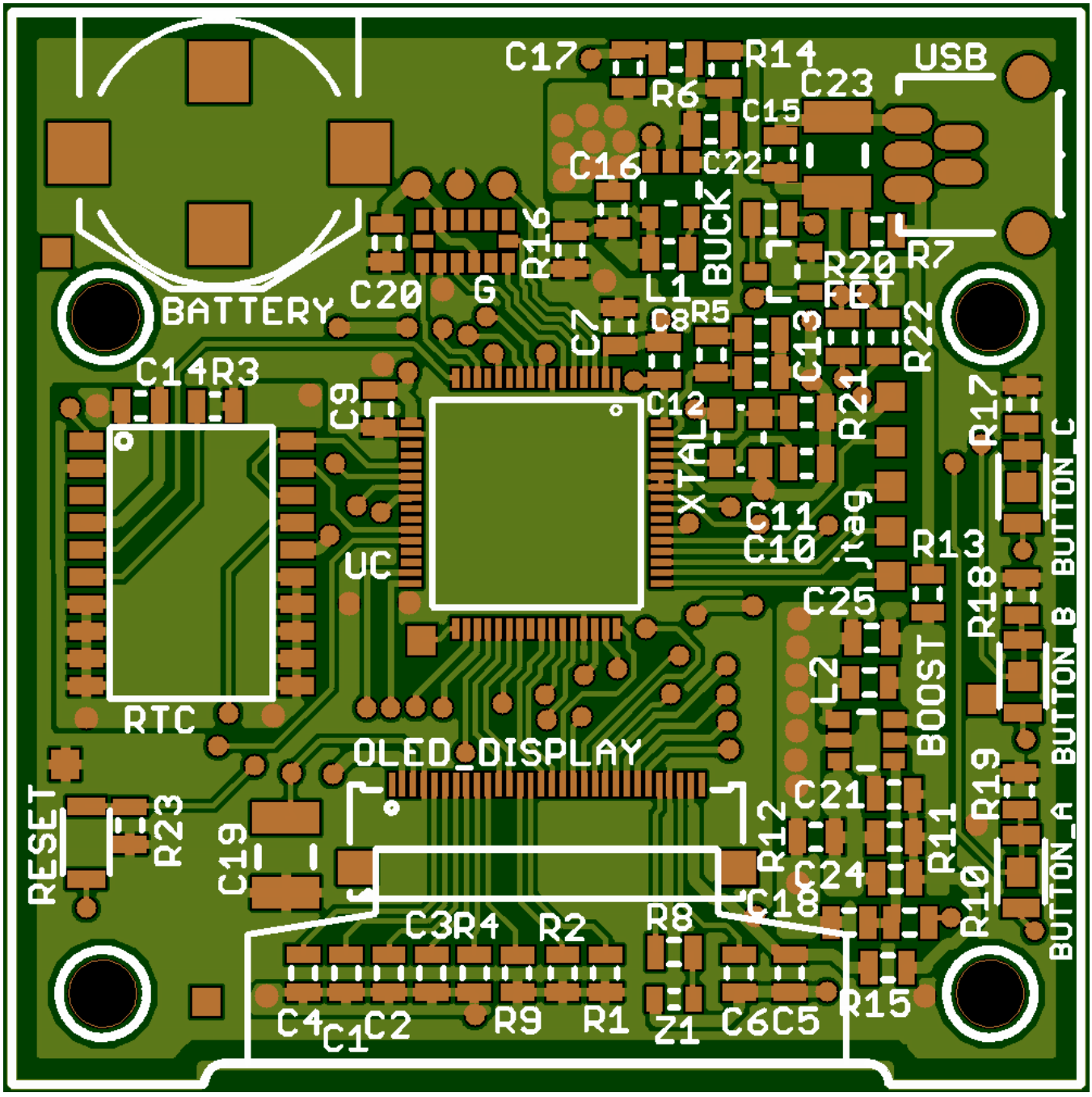

Figure 2: Rendering of front of rev00 of the PCB for arm7-oled-clock