

Brookstone Alarm Clock Prototype

Tuesday, May 13

Spring 2008

University of Southern California
Embedded Systems Design Laboratory
EE 459

Dr. Allan Weber
Mark Redekopp

Team 3A

Omar Khan
Anthony Visessonchok
Robert Wright



Figure 1: Final Product Concept Drawing

Table of Contents

Table of Figures	iv
Introduction	1
Project Outline	2
User Interface	3
System Overview	5
Hardware Block Diagrams	6
Main Clock	6
Wireless Temperature Module	6
Parts / Hardware	7
Main Clock	7
Microprocessor Subsystem	7
Oscillator	7
IIC Requirements	8
Baud Rate	8
Operation	9
Real Time Clock Subsystem	10
Operation	10
Battery Backup	10
Non-Volatile RAM	10
Alarm / Buzzer	11
Temperature Sensor Subsystem	12
Temperature VS Voltage Relationship	12
JL16 ADC Setup and Performance	14

Calculation Optimizations	15
LCD Subsystem.....	16
Operation	16
Large Character Generation.....	17
Contrast Voltage	18
Input Buttons & Wireless Receiver Subsystems	20
SCI & Baud Rate Setup	20
Data Acquisition	21
Night Light and Switch Subsystem	24
LM2674-ADJ Required Parts Calculations.....	24
Problems	25
Power Subsystem.....	26
Radio Subsystem	26
Wireless Temperature Sensor Module	27
Microprocessor Subsystem.....	27
Serial Communication.....	27
Temperature Sensor Subsystem	27
Wireless Transmitter & Power Subsystem	28
Software.....	29
Software System Diagram.....	29
Software Organization and Memory Usage.....	31
24LC256.h	31
DS1307.h.....	31
HG25504.h	31

[Embedded Systems Design Laboratory: Spring 2008: Team 3A: Brookstone]

AD22100.h	31
Cost Analysis	32
Conclusion.....	35
Appendix I: Power Consumption Analysis	I
Appendix II: Project Screenshots & Pictures.....	III
Appendix III: Main Clock Schematics	X
Appendix IV: Wireless Temperature Module Schematics	XII
Appendix V: CD	XIV
Appendix VI: Marketing Team	XV
Appendix VII: Member Contributions.....	XVI

Table of Figures

FIGURE 1: FINAL PRODUCT CONCEPT DRAWING	B
FIGURE 2: MAIN CLOCK DIAGRAM	6
FIGURE 3: WIRELESS TEMPERATURE MODULE DIAGRAM.....	6
FIGURE 4: DS1307 NVRAM ADDRESS SPACE TABLE	11
FIGURE 5: AD22100 VOUT VS TEMPERATURE RELATIONSHIP GRAPH	12
FIGURE 6: HC08 ADC DATA OUT VS TEMPERATURE RELATIONSHIP	13
FIGURE 7: TEMPERATURE VS HC08 ADC DATA OUT RELATIONSHIP	13
FIGURE 8: INTERNAL CHARACTER ROM TABLE.....	17
FIGURE 9: LARGE CHARACTER BITMAP SIZE.....	18
FIGURE 10: EEPROM ADDRESS SPACE ALLOCATION TABLE	19
FIGURE 11: KEYPAD AND WIRELESS RX MUXING	21
FIGURE 12: KEYPAD INTERRUPT WAVEFORM	22
FIGURE 13: KEYPAD INTERRUPT FLIP-FLOP	22
FIGURE 14: KEYPAD INTERRUPT WITH FLIP-FLOP WAVEFORM.....	23
FIGURE 15: SI473X SITTING ON TOP OF A \$0.25 COIN	26
FIGURE 16: MAIN SOFTWARE DIAGRAM.....	30
FIGURE 17: COST ANALYSIS TABLE	32
FIGURE 18: POWER CONSUMPTION ANALYSIS TABLE	I
FIGURE 19: WELCOME SCREEN	III
FIGURE 20: MAIN SCREEN WITH ALARMS OFF AND CELSIUS TEMP.....	III
FIGURE 21: MENU SCREEN	IV
FIGURE 22: SET SCREEN TIME.....	IV
FIGURE 23: SET DATE SCREEN	V
FIGURE 24: SET ALARMS SCREEN	V
FIGURE 25: SETTINGS SCREEN	VI
FIGURE 26: SAVED SCREEN.....	VI
FIGURE 27: MAIN SCREEN WITH ALARM ON AND FAHRENHEIT TEMP	VII
FIGURE 28: MAIN SCREEN WITH WIRELESS MODULE TURNED OFF.....	VII
FIGURE 29: MAIN PROJECT BOARD	VIII
FIGURE 30: MAIN PROJECT BOARD WITH NIGHT LIGHT ON.....	VIII
FIGURE 31: WIRELESS TEMPERATURE MODULE.....	IX

Introduction

The task for this year's EE459 embedded system capstone design class offered a variety of different obstacles, both new and old. While our final object meant to be an alarm-clock radio, the opportunity to work cooperatively with a senior-level marketing team was also provided. This had never been done before in this class and it offered a new variable to an engineer's systematic progression. As the teams were chosen, many had not worked with each other, and that in it-self, was a challenge both in scheduling and in the division of labor. The added members of the marketing team meant additional scheduling conflicts and personalities. The marketing aspect also added the role of a brand-name, which would ultimately shape the final design and functionality of the project.

The marketing team was composed of four members¹. While the engineering team held a great deal of technical control over the design and implementation of the product, the marketing team brought direction and focus to the overall design of the project. Although a great deal of their research was spent on mainly the needs and desires of future customers, it also focused on keeping true to the brand name. Much of the give-and-take during the design phase revolved around the capabilities of the engineering group contrasting with the ambitions of the customers' wants.

Given the brand of Brookstone, as well as original ideas of implementing an alarm clock, the main objective was to develop a user-friendly, yet a technically savvy, design that would provide the user with a great deal of useful information. While one of the initial requirements set forth was to create an alarm for each of the seven days as well as a general alarm, a number of "brand" features were also added. These included a large display with night light, a customizable snooze ranging from one minute to 30 minutes, weather and temperature information, as well as radio functionality. The radio would be able to play music through in-house speakers, as well as display the station, artist name, and song name as well as a volume meter on the LCD.

In order to begin in a timely fashion, meetings were immediately organized with the marketing teams to discuss designs and get an understanding of the process they will be undertaking to complete their tasks. For a while, weekly meetings were held with solely our engineering group to establish a system design, as well as with the marketing team to determine logistics.

The following report outlines the resulting project, describing its features and functionality in detail.

¹ This list of members is available in *Appendix VI*

Project Outline

As mentioned earlier, the purpose of this project was to create an innovative alarm clock based on market research. In light of the Brookstone brand name, the goal of this particular project was to create a clock that could be placed in the bedroom on a night stand, or on an office desk, providing all the information one would need to know about the current day, while looking sleek at the same time.

Although the main response from marketing focus groups revolved around the customers' desires of iPod connectivity and other forms of music integration, a lot of favorable feedback was obtained about displaying temperature and weather information, as well as display formatting, and over all function-ability of our initial design considerations.

The final design features a large LCD screen adapted to be able to display a large amount of information to the user on a single screen which includes the current time, date, and temperature.

The clock includes a total of eight alarms. Seven of these alarms are day specific and will only ring at the set time on the specified day. The eighth alarm is a general alarm which will ring at the set time every day.

The snooze button is a user customizable feature which is initially set to a 10 minute interval. The user is able to customize the snooze to their desire ranging between 1 and 30 minutes.

A night light was incorporated into the alarm clock to provide a simple lighting solution. Like the snooze, the night light has an auto shut-off feature that is user customizable between 1 and 30 minutes.

An AM/FM radio was also included in the design. When activated, the seven day forecast portion is replaced with radio information such as current station and song information and volume control.

The user is also given the choice of selecting between a buzzer or the radio as their alarm.

Furthermore, the clock includes two points of temperature acquisition. The first is an on board temperature sensor which displays the reading from the clock's surrounding atmosphere. The second is from a Wireless Temperature Module which can be placed in another room, or outside on a window sill, up to 500 feet away. Both these temperatures can be selected to display in either Celsius or Fahrenheit.

Last but not least, the clock includes a battery backup which allows the time and date to be accurately maintained and all user customizable alarms and settings to be saved in the case of power loss.

User Interface

The user interface was designed to be simple and user friendly, but to also keep a technical prowess in line of the Brookstone brand name. With that said, we needed something that looked more complicated at first glance yet turned out to be fairly intuitive once the interface was tested. The entire interface revolves around two main components: the keypad and the LCD.

The LCD normally displays the following to the user at a single glance:

- 1) Current time in 12 hour mode
- 2) Current Day, Month, Date, Year
- 3) Whether an alarm is activated
- 4) Whether the snooze is activated
- 5) Current temperature in the room
- 6) Current temperature from a wireless source
- 7) Seven day weather forecast with iconic weather representation

The keypad is composed of nine buttons, each with a specific or multiple functions:

- 1) *MENU* Button
- 2) *SELECT* Button
- 3) *RADIO* Button
- 4) *SNOOZE* Button
- 5) *NIGHT LIGHT / ALARM ON-OFF* Button
- 6) *Increment (+)* Button
- 7) *Decrement (-)* Button
- 8) *Scroll Left (←)* Button
- 9) *Scroll Right (→)* Button

The *MENU* button allows the user to access the sub-menus that control the various aspects of the clock interface. These sub-menus include adjusting the clock's time and date, which can be done by highlighting either option and pressing the *SELECT* button. Once within the sub-menu, the user changes each section by scrolling through and increasing or decreasing each digit using the *Scroll Left*, *Scroll Right*, *Increment* and *Decrement* buttons respectively. Once you have done so you can hit the *SELECT* button, and the data is saved as the LCD returns to the main display.

There is also a sub-menu that allows you to set the various alarm choices offered by the clock. The clock holds seven alarms, each tied to a specific day of the week. It also holds an eighth general alarm that, when set, will sound at the specified time every day. Once an alarm is ringing, the user can either hit the *SNOOZE* button which will start a counter and allow the alarm to sound again after the desired length, or the user can press the *NIGHT LIGHT / ALARM ON-OFF* button which will disable the alarm. When the alarm is not ringing the user can press the same button to activate the night light. Both the snooze and the light length can be set in the fourth menu entitled "settings." The time ranges from one to 30 minutes which should be an ample range for any user. Once each

setting is set to its desired value, the user hits the *SELECT* button which saves the settings and the LCD returns to the main display.

The main display is comprised of four main sections. When one first ponders upon the display, one quickly notices the large time and date displays. This is for easy viewing in low light situations. The time is displayed in standard 12-hour format with the AM/PM indicator on the side. Next to the time, there is a quadrant which displays what menu you are in as well as whether or not an alarm is set for that day. If the general alarm is set then the display always reads on, however, the display also reads on for the day that the day-specific alarms are set for as well. Beneath that quadrant, we see the temperature section, which can either be set to read in degrees Celsius or Fahrenheit (which is done in the “settings” menu), and displays the indoor temperature as well as the readings from the Wireless Temperature Module. If the wireless module is turned off the display reads an error and does not display a temperature.

The last section, if one notices, is blank and was meant to house the radio frequency information. This feature could not be implemented into the project as discussed in the *Radio Subsystem* section. Normally, the seven day forecast information obtained over the weather band frequencies would be displayed here. When the radio is turned on by pushing the *Radio* button, the forecast is replaced by information relating to the radio which includes:

- 1) Current band (AM/FM)
- 2) Current tuner frequency
- 3) Currently tuned station name
- 4) Currently playing song information
- 5) Volume information

In this mode, the *Scroll Left* and *Scroll Right* buttons function as the frequency tuner, the *Increment* and *Decrement* buttons function as the volume control, the *Select* button functions as the AM/FM button, and the *Snooze* button functions as the mute button if no alarm is ringing.

Appendix II shows screen shots and other pictures of the project.

System Overview

Although each part is discussed in more detail later in the *Parts / Hardware* section, an introduction on the overall system is important to discuss. As is evident by the block diagrams on the next page, the system is divided into various subsystems as is represented by the various color schemes. First and foremost is the LCD display whose contrast is controlled using a potentiometer. This is tied to the I/O port of the microcontroller. While the EEPROM is plugged into the IIC port, it is considered a part of the LCD subsystem since it is used to store the bitmaps of the large characters for the time and date display as well as other graphics displayed.

The Cold Cathode Florescent Tube (CCFT) night light is considered its own subsystem largely due to the switch that was constructed to allow it to be controlled by the microcontroller. It too is controlled through the I/O ports.

The clock chip, which contains a backup battery such that the real-time is stored even if the power goes out as well as the alarm buzzer, is connected to the IIC bus. The radio chip would have also been connected to this bus if it had been implemented into the system². The speaker volume would have been controlled through the radio subsystem as well.

The on-board temperature sensor is attached to the Analog-to-Digital Converter as is the temperature sensor on the wireless module which houses a smaller microcontroller of the same family.

Connected to the SCI port are the keypad encoder and the wireless receiver. The keypad was designed from scratch and utilized a three-by-three matrix encoded to create a distinct character signal for each button. This was done to provide easy customized button placements in the final design while still making use of the powerful keypad encoder.

² The radio feature was not included due to various reasons discussed in the *Radio Subsystem* section.

Hardware Block Diagrams

Main Clock

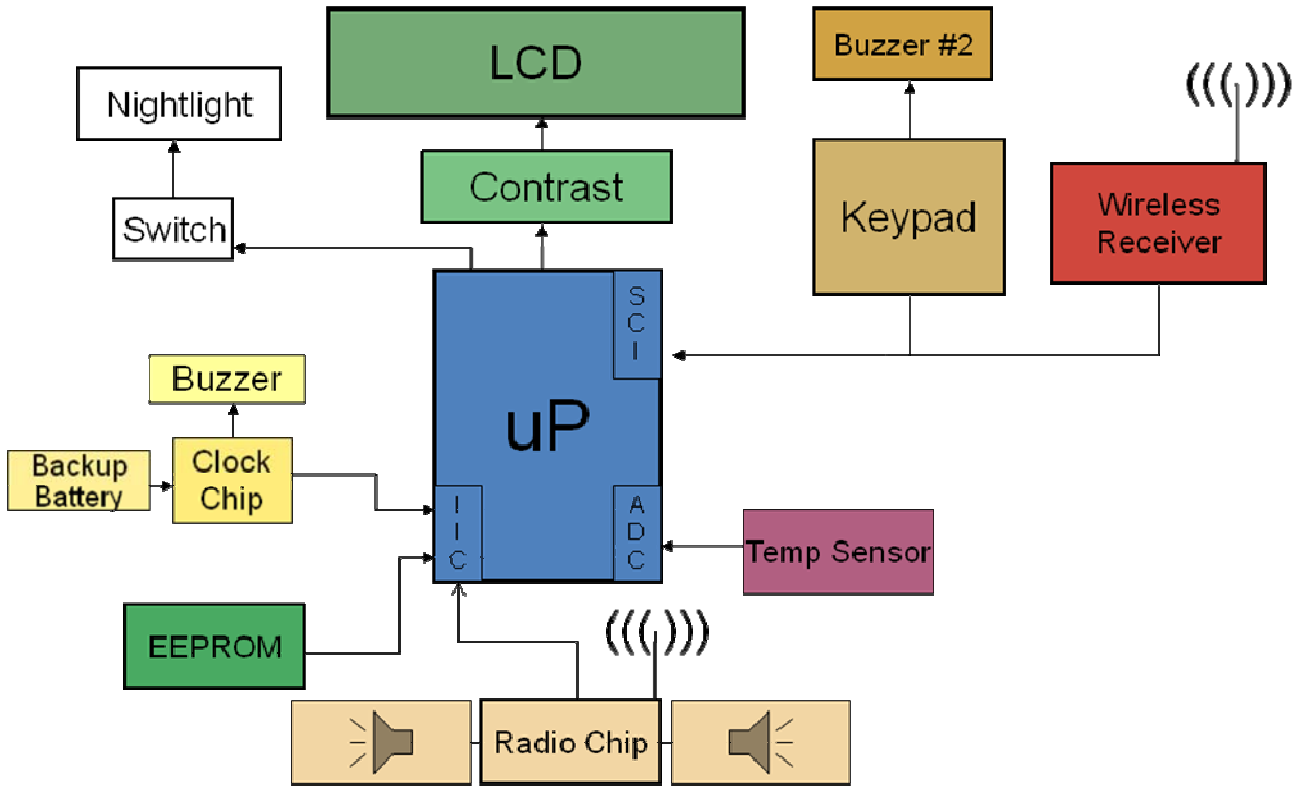


Figure 2: Main Clock Diagram

Wireless Temperature Module

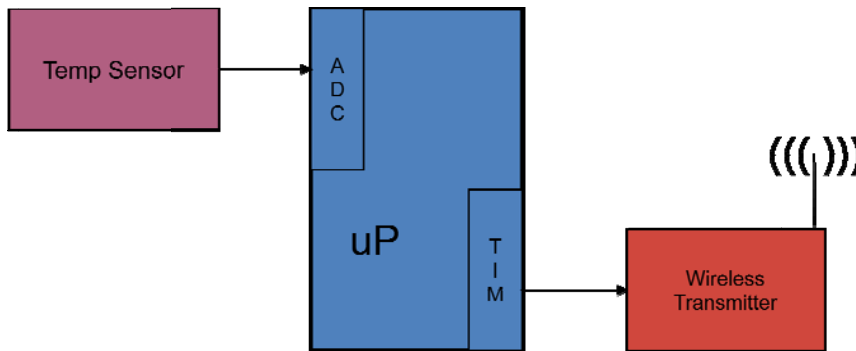


Figure 3: Wireless Temperature Module Diagram

Parts / Hardware

Main Clock

Microprocessor Subsystem

Parts Referenced: MC68HC908JL16

10MHZ Oscillator

10K Resistors (x2)

The JL16 Microprocessor from Freescale's 8-bit HC08 family line was used as the main controller unit for the project due to its low cost, high performance, and in house features. The JL16 provides various useful features such as: hardware implemented IIC and SCI communication protocols, Timer Interface Modules, Analog-to-Digital Converters, External Keypad Interrupts, and 26 General Purpose I/O pins (some with internal pull-up resistors).

Although the processor features only 16,384 bytes of user programmable Flash Memory and 512 bytes of RAM, this memory space is more than adequate for our application needs³.

Furthermore, Freescale provides a simple, low cost solution to program its line of microprocessors. The P&E Micro USB MONO8 Multilink Programmer⁴ can be purchased for \$99 and interfaces nicely with the CodeWarrior® Microcontroller Edition IDE downloadable for free from the Freescale website. For debugging purposes, the CodeWarrior IDE provides an easy to use Full Chip Simulator which allows the programmer to step through source code and see how the microcontroller's registers and I/O pins are affected. The MONO8 Programmer can also be used with P&E's in-circuit debugger for visual hardware debugging.

Oscillator

The JL16 requires an external oscillator connected to the **OSC1** pin or an RC oscillator circuit connected to the **RCCLK** pin. A 10 MHz Oscillator was used to run the project. Thus, the internally dividing clock gives an internal bus frequency of $\frac{10MHz}{4} = 2.5MHz$. The **OSCSEL** bit in the **OSC** register must be set to enable the Crystal Oscillator circuit. This action also activates the **OSC2** output and other functions of the pin, such as **PTA6** and **KBI6** are not available for use.

³ Memory usage explained in the *Software* section

⁴ Part# USB-ML-MONO8 available from <http://www.pemicro.com>

IIC Requirements

The IIC module is a multi-master bus capable of interfacing multiple devices over the same bus. Each device on the bus has a unique address which it responds to. This address is hardcoded into most devices, however, some allow partial customization of the address bits. The IIC standard requires external pull-up resistors on the open-collector bus of the IIC module to function⁵. 10K resistors to 5 volts are adequate for this application. Please refer to the project schematic in *Appendix III* for details.

The JL16 provides access to the IIC module through two different ports **PTA2/KBI2/SDA** and **PTA3/KBI3/SCL** or **PTD7/RxD/SDA** and **PTD6/TxD/SCL**. For this application, IIC was operated on **PTA2** and **PTA3** due to the fact that the SCI port also lies on **PTD6** and **PTD7**. This is accomplished by setting the **CONFIG2_IICSEL** bit during microprocessor initialization. The IIC module must also be enabled by setting the **MMCR_MMEN** bit at this time.

Baud Rate

The IIC standard has many modes of operation:

- | | | |
|--------------------|---|---------------|
| 1) Standard Mode | - | 100 kbits/sec |
| 2) Fast Mode | - | 400kbits/sec |
| 3) Fast Mode Plus | - | 1 Mbit/sec |
| 4) High Speed Mode | - | 3.2 Mbits/sec |

Each IIC devices has specific supported modes of operation. The datasheets of each device must be checked to ensure that all devices support the same mode of operation. Standard Mode was used for this application.

The IIC baud rate can be easily calculated using the following equation:

$$\text{baud rate} = \frac{\left(\frac{\text{Oscillator frequency}}{4}\right)}{\text{dividing factor}}$$

The *dividing factor* is determined from *Table 8-2* on page 118 of the JL16 datasheet

Since Standard Mode is used in this application, the baud rate needs to be 100kbps. Thus:

$$\text{baud rate} = \frac{\left(\frac{10\text{MHZ}}{4}\right)}{\text{dividing factor}} = 100\text{kbps}$$

$$\therefore \text{dividing factor} = 25$$

However, since 25 is not a possible choice, either 16 or 32 must be chosen for the *dividing factor*. Since using 16 will give a baud rate of 156kbps, which is over the maximum acceptable rate of 100kbps for Standard Mode, 32

⁵ Wikipedia: I²C <http://en.wikipedia.org/wiki/I%C2%B2C>

was used giving a baud rate of 78kbps. To accomplish this, **MIMCR_MMBR** must be set to 2 before the IIC module is enabled during the microprocessor initialization.

Operation

Custom functions were created to allow easy transactions over the IIC bus. The methods shown below must be followed to achieve successful IIC transactions.

An example of an IIC write transaction:

```
IIC_trans_start(<dev addr>,<first data>);
IIC_trans_ack(<second data>);
IIC_trans_ack(<third data>);
.
.
IIC_trans_ack(<last data>);
IIC_trans_ack(<dummy data>);
IIC_stop();
```

An example of an IIC read transaction:

```
IIC_rec_start(<dev addr>);
IIC_rec_wait();
temp_0=MMDRR;
IIC_rec_wait();
temp_1=MMDRR;
.
.
IIC_rec_disableAck();
IIC_rec_wait();
temp_last=MMDRR;
IIC_stop();
```

IMPORTANT: All interrupts must be disabled **PRIOR** to any IIC transaction. This can be accomplished by using the *DisableInterrupts; directive*.

For Example:

```
.
.
DisableInterrupts;
<<IIC transaction functions>>
EnableInterrupts;
.
.
```

Real Time Clock Subsystem

Parts Referenced: DS1307

DS32KHZ Oscillator

DBX05 Buzzer

Potentiometer

3V Lithium Battery

In order to keep time accurately, a Real Time Clock (RTC), the Maxim DS1307 was used. This chip uses the IIC interface to communicate with the microprocessor and requires a 32KHZ Oscillator to maintain time accuracy. The RTC maintains seconds, minutes, hours, day, date, month, and year information in Binary Coded Decimal (BCD) system with leap year compensation valid up to 2100. The RTC can either be set to 12 hour mode with an AM/PM indicator or 24 hour mode. In this application, 24 hour mode was not utilized.

Operation

The initial power-on state of the DS1307's registers are not defined. Therefore, the **CH** bit in the seconds register must be cleared on every system reset to enable the RTC. An initial power-on time and date can also be written to the registers at this time. Once the **CH** bit is cleared and valid data has been input to the time registers, the RTC will maintain itself and does not require any further attention from the microprocessor. The registers can simply be read at the software's discretion to display the current time and date. This is accomplished in the project by software polling.

Battery Backup

The RTC is capable of automatic power failure detection and continuous operation using a 3V Lithium Battery for up to 10 years. To make use of this feature, both the RTC and the 32 KHZ Oscillator must be connected to the battery via the **BAT** pin. If power failure is detected, the RTC will go into low power mode, cutting off all IIC communication. The **CH** bit must be cleared again if power is restored to enable the clock. However, time and date information do not need to be re-written as they are maintained accurately in low power mode.

IMPORTANT: The battery pin on the DS1307 must be connected to ground if there is no backup battery used. If this is not done, then the BAT pin will eventually float higher than the VDD pin causing the RTC to go into low power mode.

Non-Volatile RAM

The DS1307 also features 56 bytes of non-volatile RAM. As long as the backup battery is connected, the NVRAM will retain all of its information when main power is turned off. The NVRAM was used to store all user customizable settings including the 8 alarms, snooze duration, night light duration, and the temperature units.

Figure 4 shows a table of the NVRAM address space used.

Starting Address	Variable Name	Description
0x08	ALRM	General Alarm Day, HR,MIN
0x0B	ALRM0	Mon Alarm Day, HR,MIN
0x0E	ALRM1	Tue Alarm Day, HR,MIN
0x11	ALRM2	Wed Alarm Day, HR,MIN
0x14	ALRM3	Thu Alarm Day, HR,MIN
0x17	ALRM4	Fri Alarm Day, HR,MIN
0x1A	ALRM5	Sat Alarm Day, HR,MIN
0x1D	ALRM6	Sun Alarm Day, HR,MIN
0x20	CURR_SNOOZE	Snooze Duration (1 min – 30 min)
0x21	CURR_NL	Night Light Duration (1 min – 30 min)
0x22	TEMP_UNIT	Temperature Units (°C or °F)

Figure 4: DS1307 NVRAM Address Space Table

Alarm / Buzzer

The DBX05 buzzer, which was used as the alarm buzzer, requires a square wave to produce a sound. The frequency controls the pitch and the voltage offset controls the volume. The DS1307 RTC's programmable Square Wave Output was used to drive the buzzer at 4.069 KHZ by setting the **RS[1:0]** = {0,1} in the control register. The **SQWE** bit in the control register enables (set) or disables (clear) the square wave. If the alarm is ringing, the **SQWE** bit is set if the seconds are even and cleared if the seconds are odd. This creates a beeping sensation for the alarm.

Since the DS1307 is an open-collector device, a pull-up resistor to VCC must be connected between the buzzer and the **SQWOUT** pin. A potentiometer is used here to control the volume of the buzzer.

Temperature Sensor Subsystem

Parts Referenced: AD22100 Temperature Sensor

The AD22100 Temperature Sensor is an analog device which outputs a unique voltage for each temperature reading between 0.25V for -50°C and 4.75V for $+150^{\circ}\text{C}$ using a 5V reference voltage, making it ideal for many temperature sensing applications.

Temperature VS Voltage Relationship

Using the information provided in the datasheet, the following relationships were derived:

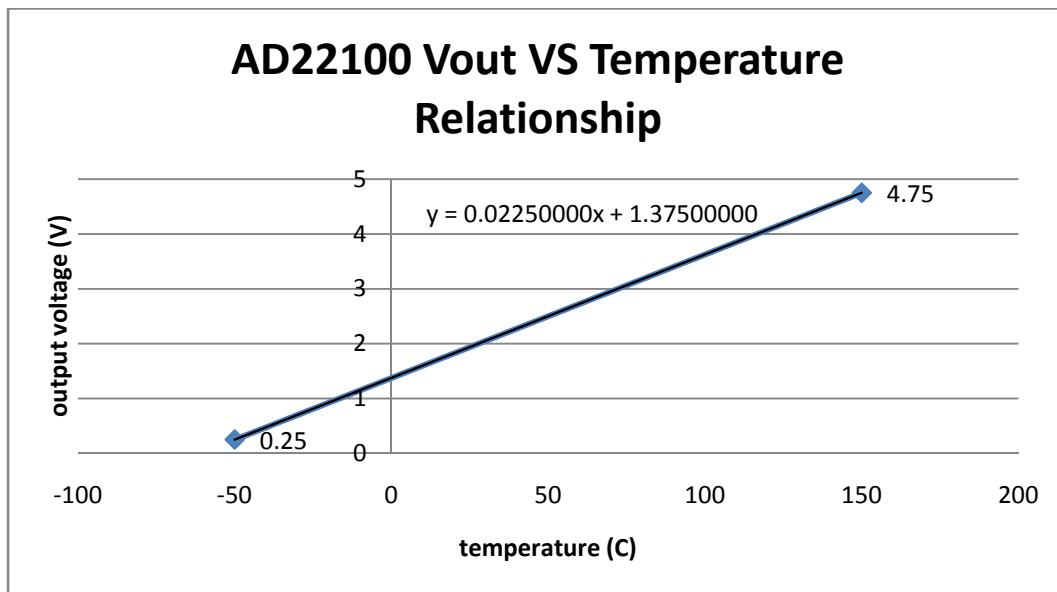


Figure 5: AD22100 Vout VS Temperature Relationship Graph

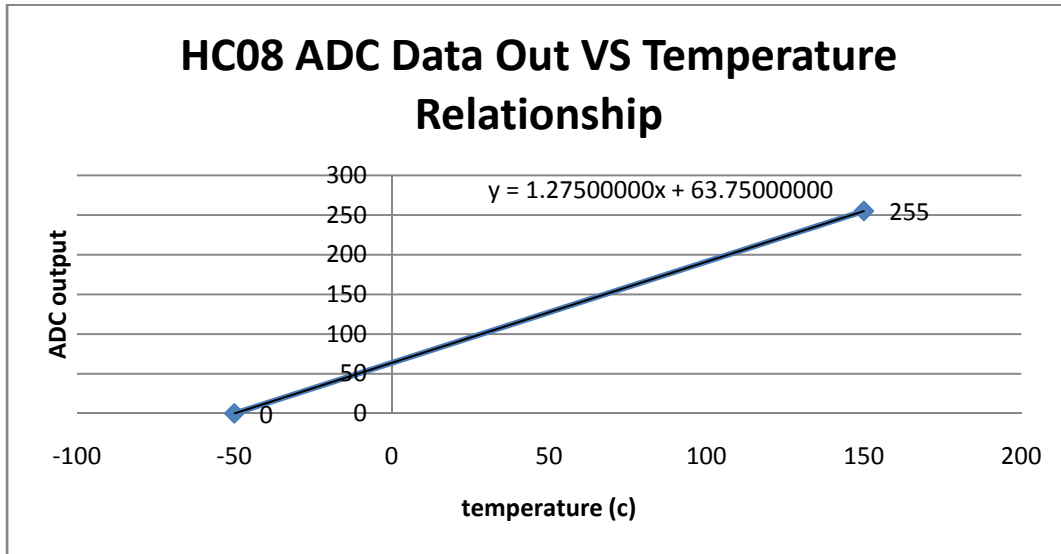


Figure 6: HC08 ADC Data Out VS Temperature Relationship

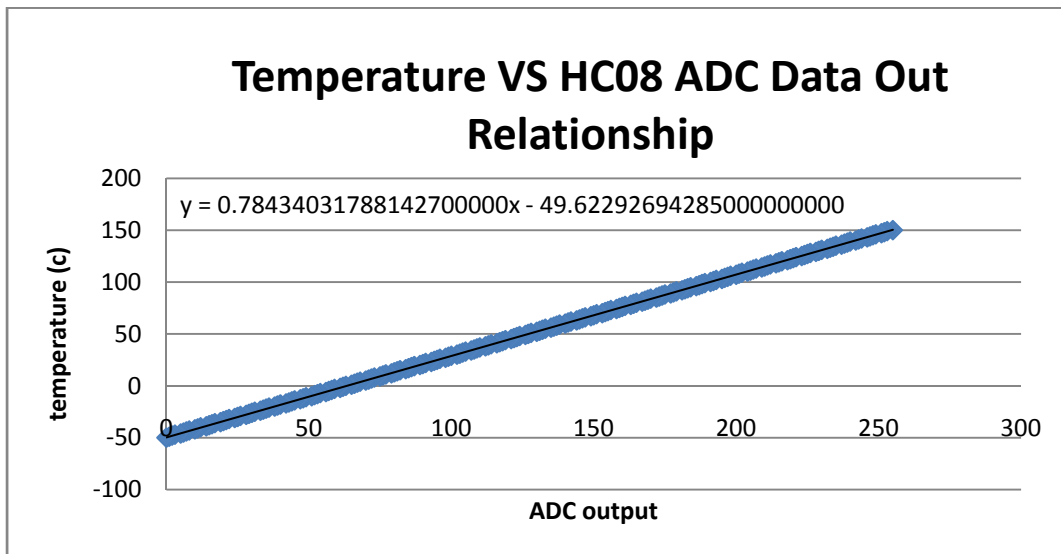


Figure 7: Temperature VS HC08 ADC Data Out Relationship

The equation in *Figure 7* is used to calculate the temperature (y) based on the ADC reading (x) in the software.

JL16 ADC Setup and Performance

The ADC here is used in high speed, short sample configuration, 8-bit, right-justified, **ADCSC** write-triggered mode. The internal bus clock is used with a dividing ratio of 1 (**ADIV=0**). Thus, the **ADCLK** register is set to 0x10. This gave the most accurate results during testing.

The following steps need to be taken to start an ADC conversion:

- 1) The **ADCLK** register must be set first as described above.
- 2) The `START_JL16_ADC(ADC#)`⁶ function must be called.
The **ADCSC_COCO** bit is set to indicate that a conversion is complete.
- 3) The ADC result register (**ADR**) must be read to clear the **COCO** bit to allow another conversion to take place.
- 4) When another conversion is needed, the `START_JL16_ADC(ADC#)` function is called again.

An example of ADC conversion using ADC port 9:

```
ADCLK=0x10;
START_JL16_ADC(0x09);
.
.
while(1){
    if(ADCSC_COCO==1){
        .
        .
        START_JL16_ADC(0x09);
    }
}
```

For this application, the ADC result needs to be subtracted by 7 in order to achieve the correct value based on the AD22100's output voltage as derived in *Figure 7*. Thus, the temperature calculation becomes:

$$TEMP_C = (0.784340317881427 * (ADC_{RESULT} - 7)) - 49.62292694285$$

After this result is calculated, the temperature can be easily converted to Fahrenheit using the following equation:

$$TEMP_F = (1.8 * TEMP_C) + 32$$

Since the temperature is not expected to change rapidly, new temperature values are calculated every 5 seconds. This value can be increased or decreased by changing the `UPDATE_TEMP_EVERY` definition in the Main Clock's Source Code to reflect the seconds delay required in updating the temperature between 1 and 59 seconds.

⁶ This function is defined in the `ad22100.h` header file.

Calculation Optimizations

The floating point calculations require a lot of CPU overhead and take a fairly long time to complete. This overhead can be reduced and performance can be optimized by using a lookup table instead of the equation calculation to determine the current temperature. However, the amount of space taken up by the look up table must be compared to the change in performance achieved by changing the calculation method as the look up table could take up a fairly large amount of space.

LCD Subsystem

Parts Referenced: Hyundai HG25504 LCD

DC-32 (DC-DC inverter)

LM337 Adjustable Negative Voltage Regulator

Potentiometer

24LC256 32Kx8 EEPROM

The Hyundai HG25504 was used as the main display unit due to its size and ease of use. The LCD features a display size of 256x128 pixels of multilayer graphic and text display. The LCD also supports standard ASCII commands for its table of built in 8 bit characters. Bitmaps can be easily created and displayed for larger characters or graphics.

Operation

The LCD operates over the parallel I/O interface using 8 data lines and 5 control lines. The \overline{RD} pin can be tied to 5V VCC as nothing is ever read from the LCD. This saves one extra pin having to be controlled by the microprocessor.

On reset, a series of initialization commands must be sent to the LCD in order to bring it into working state⁷. Once the LCD has been initialized, the following steps need to be taken to write to the LCD:

- 1) The cursor needs to be placed in the correct place either on the text layer (starting at address 0x000) or on the graphics layer (starting at 0x1000).
- 2) The write command needs to be issued.
- 3) The data to be written needs to be sent to the LCD.

An example of writing 2A to the LCD using the internal character ROM⁸:

```
init_LCD();
.
.
LCD_TEXTADDR=LCD_TEXTLAYER_START;
set_cursor_loc(LCD_TEXTADDR_ADDRH,LCD_TEXTADDR_ADDRL);
writeNUM(2);
writeSTR("A");
```

The `writeNUM(int)` and `writeSTR(char[])` functions incorporate sending the write command and the data to be written to the screen.

⁷ LCD Initialization commands can be found in the function `init_LCD()` in the source code. *Figure 8* shows the internal character ROM table.

⁸ This example makes use of the `HG25504.h` header file. Please see the *Software* section for details.

		Character code bits 0 to 3															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Character code bits 4 to 7	2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
	A																
	B																
	C		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	­	®	¯
	D	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
1																	

Figure 8: Internal Character ROM Table

Large Character Generation

The LCD’s graphics layer allows for bitmaps to be displayed. This allows for custom font and graphics generation. To display large numbers and characters, custom bitmaps were created and stored in a 32Kx8 EEPROM. These bitmaps were created by hand but a program that reads a 1 bit BMP file and outputs the HEX values of the image called BMP2HEX⁹ was later discovered. This program was used to create the bitmap of the Brookstone® logo displayed at power-on. *Figure 9* shows the size of the character bitmaps and *Figure 10* shows the EEPROM Address Space Allocation Table.

IMPORTANT: For final production, the EEPROM should be changed to a permanent ROM.

⁹ This program is available from <http://ee.usc.edu/library/ee459/>



Figure 9: Large Character Bitmap Size

Each block is taken to be 8 bits wide, thus:

$$\text{Size of Number Bitmaps} = (3 * 5) * 8 \text{ bits} = 120 \text{ bits}$$

$$\text{Size of Alphabet Bitmaps} = (3 * 3) * 8 \text{ bits} = 72 \text{ bits}$$

An example of writing 2A to the LCD using the large character bitmaps¹⁰:

```
init_LCD();
.
.
LCD_TEXTADDR=LCD_GRAPHICLAYER_START;
writeGraphic(2,LCD_TEXTADDR,3);
LCD_TEXTADDR=LCD_GRAPHICLAYER_START+5;
writeGraphic('A',LCD_TEXTADDR,3);
```

The `writeGraphic(char,location,width)` function takes care of cursor placement, and length determination based on what type of graphic is going to be displayed. The width of the graphic needs to be provided as a parameter, however, this field can also be incorporated into the function.

Contrast Voltage

The HG25504 needs a negative voltage for contrast. Although this varies based on the temperature and the total system load, the operating voltage will normalize between -11V and -10V. A DC-32 DC-DC inverter was used to convert +5v to an amplified and inverted voltage of around -20V. This varies based on the load on the inverter. The -20V was then fed through an LM337 Adjustable Negative Voltage regulator equipped with a potentiometer which allows for a very efficient contrast control system.

IMPORTANT: Care must be taken here not to exceed LCD maximum voltage ratings. The potentiometer must be limited such that the maximum voltage rating of the V0 pin is not exceeded.

¹⁰ This example makes use of the `HG25504.h` and `24LC256.h` header files. Please see the *Software* section for details.

[Embedded Systems Design Laboratory: Spring 2008: Team 3A: Brookstone]

Number Bitmaps		Alphabet Bitmaps	
Address	Description	Address	Description
0x0000 - 0x00078	0	0x04BA - 0x0502	A
0x0079 - 0x00F1	1	0x0503 - 0x054B	B
0x00F2 - 0x016A	2	0x054C - 0x0594	C
0x016B - 0x01E3	3	0x0595 - 0x05DD	D
0x01E4 - 0x025C	4	0x05DE - 0x0626	E
0x025D - 0x02D5	5	0x0627 - 0x066F	F
0x02D6 - 0x034E	6	0x0670 - 0x06B8	G
0x034F - 0x03C7	7	0x06B9 - 0x0701	H
0x03C8 - 0x0440	8	0x0702 - 0x074A	I
0x0441 - 0x04B9	9	0x074B - 0x0793	J
Graphics Bitmaps		0x0794 - 0x07DC	K
Address	Description	0x07DD - 0x0825	L
0x0C24 - 0x0E91	Brookstone® Logo	0x0826 - 0x086E	M
		0x086F - 0x08B7	N
		0x08B8 - 0x0900	O
		0x0901 - 0x0949	P
		0x094A - 0x0992	Q
		0x0993 - 0x09DB	R
		0x09DC - 0x0A24	S
		0x0A25 - 0x0A6D	T
		0x0A6E - 0x0AB6	U
		0x0AB7 - 0x0AFF	V
		0x0B00 - 0x0B48	W
		0x0B49 - 0x0B91	X
		0x0B92 - 0x0BDA	Y
		0x0BDB - 0x0C23	Z

Figure 10: EEPROM Address Space Allocation Table

Input Buttons & Wireless Receiver Subsystems

Parts Referenced: EDE1144 Keypad Encoder

4MHZ Oscillator

Push Buttons (x9)

RLp434 Wireless Receiver from WRL-07813 package

DM74LS125A (Quad Tri-State Buffer with active low enable)

DM74LS04 (HEX Inverter)

DM74LS08 (Quad 2 input AND gates)

DM74LS32 (Quad 2 input OR gates)

DM74LS174 (6 bit edge triggered register)

Nine simple push buttons are used as an input scheme for the system as described in the *User Interface* section. The buttons are connected in a matrix form and are then connected into the EDE1144 Keypad encoder. The encoder requires a 4MHZ Oscillator to operate over a 4 pin parallel or a 1 pin SCI interface programmable to run at either 2400 or 9600 baud rate. The encoder also provides a square wave whenever a button push is detected. This output is connected to a buzzer to provide a beep on a button push.

The WEL-07813 Wireless Receiver/Transmitter package was used to communicate with the Wireless Temperature Sensor Module¹¹. The package is a wireless serial data what-you-see-is-what-you-get pair. Meaning, the serial data being received by the transmitter's data input pin is what will be output on the receiver's data output pin. This particular package supports data baud rates of up to 2400, however packages with higher baud rates are also available. This pair has a range of up to 500 ft in open space and an antenna is required on both the transmitter and receiver. A simple twisted wire was adequate for testing purposes.

One important aspect to note is that the receiver's data output will oscillate if it does not receive any data from the transmitter, or if the transmitter's data input pin is either held high or low for a long period of time¹¹.

SCI & Baud Rate Setup

Since both the wireless and keypad encoder support 2400 baud rate, the serial port on the JL16 must be setup to support the same. The baud rate can be calculated using the following equation:

$$\text{baud rate} = \frac{SCI_{clk}}{(64 * PD * BD)} = \frac{\frac{10MHZ}{4}}{(64 * 1 * BD)} = 2400 \therefore BD = 16$$

$$SCI_{clk} = \text{bus clock}$$

$$PD = \text{prescaler divisor}$$

$$BD = \text{baud rate divisor}$$

Thus, the SCBR register must be set to 0x04 (SCP=0 and SCR=4) to achieve a 2400 baud rate.

¹¹ See *Wireless Temperature Module* section for further details.

Furthermore, since data is only being received over the SCI port, only the Rx pin needs to be enabled by setting the **SCC2_RE** bit and the entire SCI module must also be enabled by setting the **SCC1_ENSCI** bit during microprocessor initialization.

Data Acquisition

Data is acquired through the serial port using the SCI Receive Interrupt. Interrupts must be enabled using the *EnableInterrupts*; directive during microprocessor initialization. The SCI Rx Interrupt must also be enabled by setting the **SCC2_SCRIE** bit at this time.

Interrupt Service Routines (ISRs) can be defined using the following compiler directive:

```
void interrupt <vector #> <Function Name>(<function parameters>)
```

By observing the ROM Table on page 32 of the JL16 datasheet, the ISR for SCI Receive can be defined as the following:

```
void interrupt 13 SCISr(void){
.
.
.
}
```

Since there is only one SCI Rx port, tri state buffers are used to separate the keypad encoder and the wireless receiver. Normally, the wireless receiver is polled for data. When a button is pushed, the keypad encoder generates an interrupt signal (KP_SEL) before sending data over the serial port. This signal is used to enable the tri-state buffers, as shown in *Figure 11*, and select the keypad encoder as the device sending the data to the microprocessor.

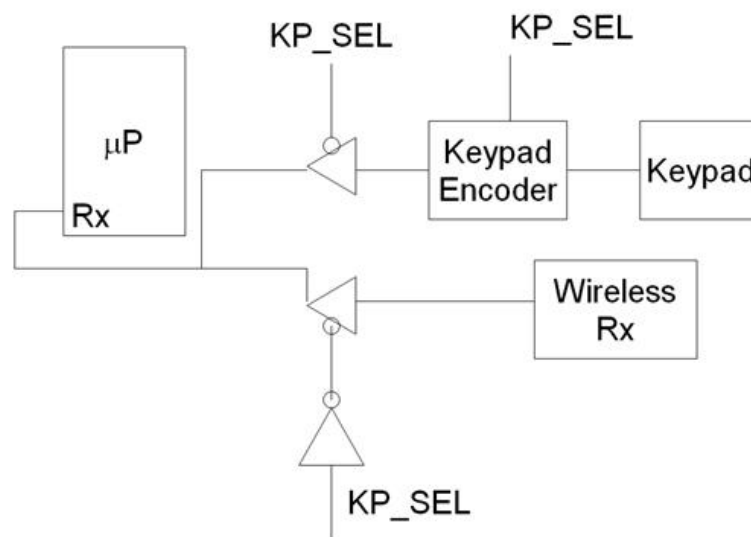


Figure 11: Keypad and Wireless Rx Muxing

Although this tri-stating solution allows accurate data reception from both devices, the KP_SEL signal disables too soon before the microprocessor even enters the ISR as shown in *Figure 12*. This means that even though valid data is being read from both the keypad and the wireless, the ISR is not able to identify which device the data came from. To solve this problem, a flip-flop is used, as shown in *Figure 13*, to hold the KP_SEL signal when the keypad encoder is sending data until the ISR can process it accordingly. The flip-flop is then cleared at the end of the ISR by setting the KP_SEL_CLR pin. *Figure 14* shows this operation on a waveform.

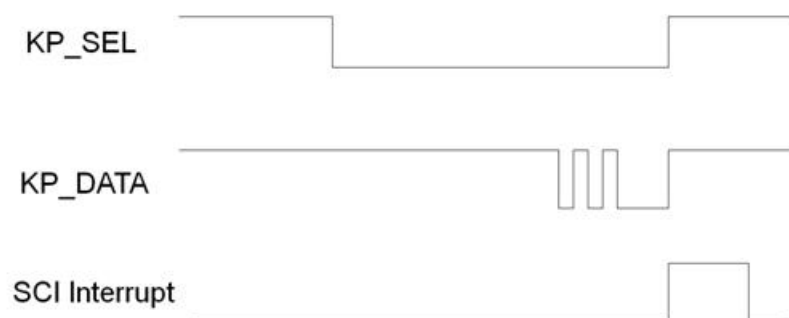


Figure 12: Keypad Interrupt Waveform

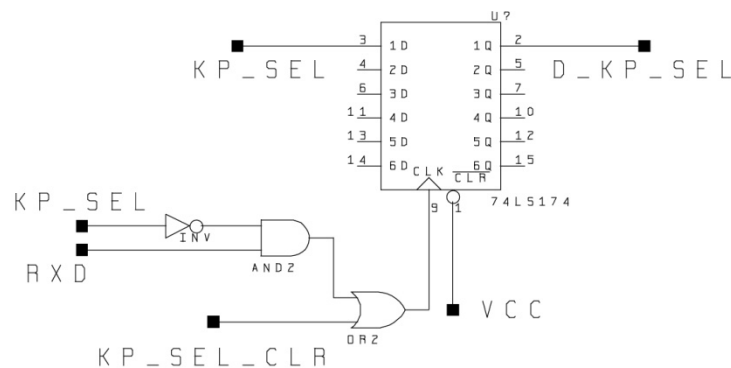


Figure 13: Keypad Interrupt Flip-Flop

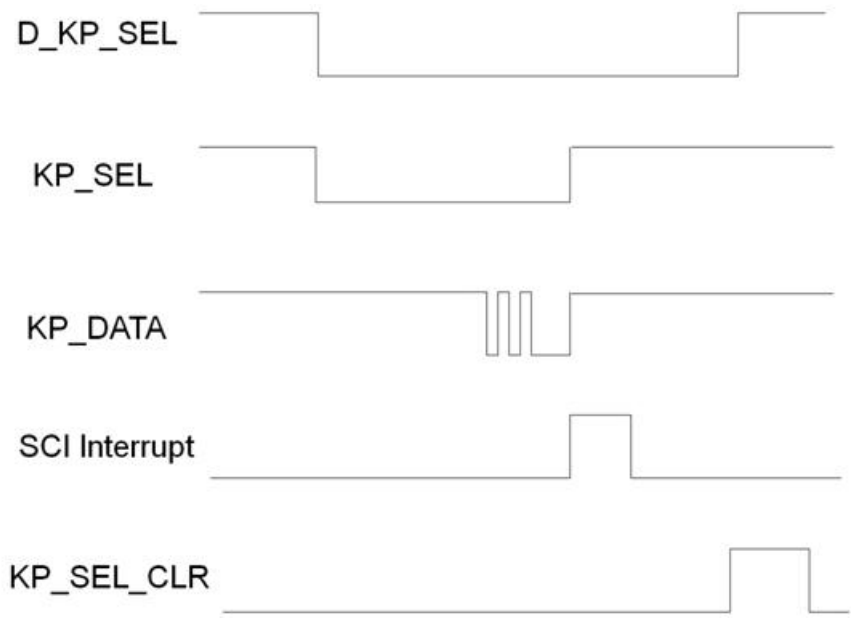


Figure 14: Keypad Interrupt with Flip-Flop Waveform

Night Light and Switch Subsystem

Parts Referenced: LM2674-ADJ (Adjustable Voltage MOS-Switch)

FL-4W (White CCFT Lamp)

INV-4 (CCFT Lamp Inverter)

The FL-4W, a 4 inch white Cold Cathode Florescent Tube (CCFT) Lamp, was chosen as the night light because of its low heat dissipation, low power consumption, and very bright light output. This lamp requires an inverter for operation and optimizes at 12V max with an operating current of 250mA. Due to these load requirements, the LM2674-ADJ Switch was used to control the lamp through the microprocessor. The LM2674-ADJ supports up to 500mA load current with output voltages ranging from 1.21V to 37V, and input voltages ranging from 8V to 40V. Therefore it is an ideal solution for this application.

LM2674-ADJ Required Parts Calculations

Although the LM2674 is called the Simple Switcher®, it is not simple in any shape or form. The MOS-Switch requires very specific parts and very careful tuning to operate accurately. The datasheet provides extensive information, equations, charts, and graphs for selecting external parts that are required for operation. The following outlines the procedure followed on page 18 of the LM2674 datasheet for selecting these parts for this application.

Desired input and output voltages:

$$V_{in(max)} = 15V$$

$$V_{out} = 12V$$

1) Programming Output Voltage (R1 & R2)

$$V_{ref} = 1.21V$$

$$V_{out} = V_{ref} * \left(1 + \frac{R2}{R1} \right)$$

$$240\Omega \leq R1 \leq 1.5k\Omega \rightarrow R1 = 270\Omega$$

$$12V = 1.21V * \left(1 + \frac{R2}{270\Omega} \right) \therefore R2 = 2.4k\Omega$$

2) Inductor Selection (L1)

$$V_{sat} = 0.25V$$

$$V_D = 0.5V$$

$$E * T = (V_{in(max)} - V_{out} - V_{sat}) * \frac{V_{out} + V_D}{V_{in(max)} - V_{sat} + V_D} * \frac{1000}{260} = 8.66 \therefore 47\mu H \text{ (L13 based on graphs)}$$

3) Output Capacitor (Cout)

$$\therefore 150\mu F / 35V \text{ (C12 based on graphs)}$$

4) Catch Diode Selection (D1)

$$\therefore 40V = V_R, 3A \text{ (based on graphs)}$$

5) **Input Capacitor (Cin)**

$\therefore 68\mu F/35V$ (300mA) (based on graphs)

6) **Boost Capacitor (CB)**

For all applications:

$CB = 0.01\mu F/50V$ ceramic capacitor

Problems

Although accurate calculations were made, it was not possible to obtain all the parts recommended in the datasheet. Using capacitors available in the lab, it was impossible to certify that all requirements for the RMS current ratings were being met.

If the lamp is turned on at 15V, it will pull current away from the rest of the board instead of pulling more current from the power source. However, by experimentation it was found out that if a 6V power supply was used, the system functioned as expected. Although the lamp is only working at half its potential at 6V, it was still bright enough to be acceptable for this application. It is highly suspected that the culprit for this problem is the input capacitor (Cin) as the datasheet places extremely high emphasis on the RMS current rating requirements of this capacitor. For future implementations of this switch, parts listed in the datasheet MUST be ordered.

Power Subsystem

Parts Referenced: 6V AC-DC power adapter (500mA min current rating)

MC7805CT 5V regulator

CR2032 3V Battery

To accommodate the CCFT Lamp, a 15V power supply could not be used as explained in the *Night Light and Switch Sub System* section. Therefore, a 6V power supply is used. This power adapter should be able to support at least 500mA current load. Since the remainder of the board runs off 5V, the MC7805CT 5V regulator is used to provide the 5V for the rest of the chips.

The CR2032 3V Lithium Battery is used to provide backup power for the DS1307 RTC and the DS32KHZ Oscillator.

Radio Subsystem

Parts Referenced: Si473x

A chip from the Si473x family of radio chips was purchased to incorporate radio functionality into the alarm clock. The chip supports Weather Band tuning and AM/FM smart tuning with Radio Broadcast Information, such as station and song name, and operates over an IIC interface. However, this feature could not be included in the project due to legal and technical issues.

Since the chips are currently under a Non Disclosure Agreement (NDA), an agreement needed to be signed to obtain a datasheet. Furthermore, the size of the chip was a bigger problem as there was no equipment readily available to be able to make connections on the chip.

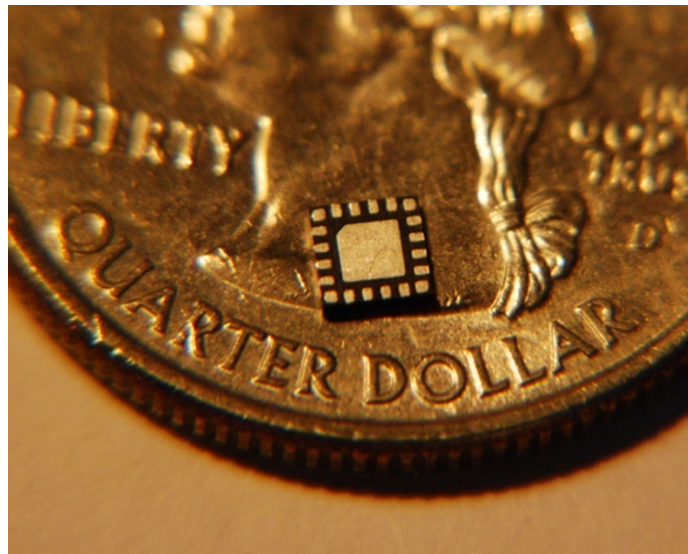


Figure 15: Si473x sitting on top of a \$0.25 coin

Wireless Temperature Sensor Module

Microprocessor Subsystem

Parts Referenced: MC68HC908QT4

The QT4 Microprocessor from Freescale's 8-bit HC08 family line was used as the controller unit for the wireless module due to its extremely low cost and high performance. The QT4 comes in a 8 pin package and provides 6 I/O ports, 6 Keyboard Interrupts, 4 Analog-to-Digital converters, and a 2 channel Timer Interface Module with 128 bytes of RAM and 4096 bytes of Flash memory.

The QT4 also provides an internal oscillator with a frequency of 12.8MHZ. This internally dividing clock results in an internal bus frequency of $\frac{12.8MHZ}{4} = 3.2MHZ$.

Serial Communication

Since the QT4 does not include hardware implemented serial communications, application note AN2502¹² was used to simulate this using the Timer Interface Modules.

The QT4 was programmed using assembly language to guarantee the least amount of CPU resources being used for the serial communication. The aforementioned application note explains this implementation in detail. However, it must be noted that the provided example was created as a byte-bouncing demonstration, i.e. the software waits until a byte is received and transmits the same byte back. For this application, the example was modified to constantly transmit data with each byte being separated by 3 bit times. This constant data transmission also takes care of the wireless receiver's data output pin oscillation problem mentioned in the *Wireless Receiver* section.

Temperature Sensor Subsystem

Parts Referenced: AD22100 analog temperature sensor

The AD22100 is controlled in exactly the same manner here as on the Main Clock with the JL16. Please refer to the *Temperature Sensor Subsystem* section for the Main Clock for details.

The ADC on the QT4 converts the voltage reading from the AD22100 and this value is transmitted directly to the Main Clock where the JL16 calculates its corresponding temperature value.

¹² AN2502: *Using Two Channels of the HC08 TIM to Achieve a Full-Duplex Software SCI* can be downloaded from <http://www.freescale.com>

Wireless Transmitter & Power Subsystem

Parts Referenced: TLP434 Wireless Transmitter from WRL-07813 package

MC7805CT 5V regulator

9V Battery

The Wireless Temperature Module is currently powered by a 9V battery. Although this is adequate for testing purposes, it is highly recommended to replace this with a suitable 12V battery system, which is the highest acceptable supply voltage for the wireless transmitter. A higher voltage supply for the wireless transmitter means a further transmission range for the wireless module.

The remainder of the system is run on 5V using a MC7805CT 5V Regulator.

Software

Software System Diagram

Figure 16 on the following page shows the system diagram which can be considered very similar to a state machine diagram.

Black arrows signify transactions that involve IIC write operations prior to state change.

Blue arrows signify a change in state.

Orange arrows signify a return to the state that originally called the current state.

Although it is not explicitly shown, each button push will trigger the SCI Rx Interrupt ISR. This ISR is also triggered every time data is received from the Wireless Temperature Module.

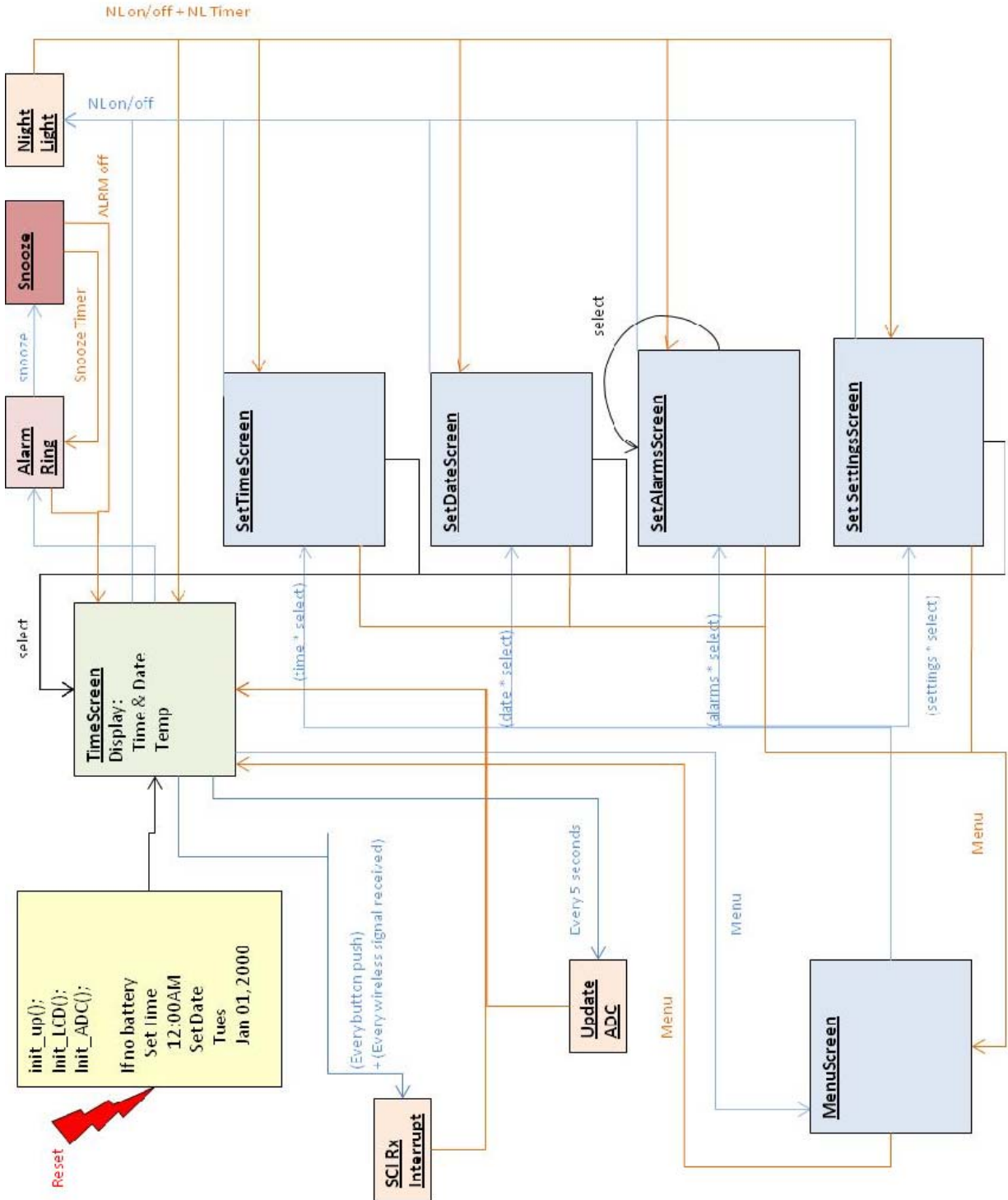


Figure 16: Main Software Diagram

Software Organization and Memory Usage

The software runs in a simple while loop calling different functions based on the current mode. These include: Time, Menu, Set Time, Set Date, Set Alarms and Set Settings. Other functions are called from these main functions for lower level operations.

Header files were created for each major component for modularity and ease of programming. However, this approach can use up a lot of memory due to many control variables and device register simulation variables being declared in each header file.

The current code takes up 12242 bytes of memory. This translates to:

$$\frac{12242 \text{ bytes}}{16384 \text{ bytes}} \approx 75\% \text{ usage of available Flash memory}$$

This should be further optimize-able to use less memory.

On the other hand, the source code for the QT4 uses only 355 bytes of memory. This translates to:

$$\frac{355 \text{ bytes}}{4096 \text{ bytes}} \approx 9\% \text{ usage of available Flash memory}$$

Appendix V contains a copy of all the source code and header files.

24LC256.h

This header implements structures and definitions used with the 32x8 EEPROM. Specific details are provided within the header file itself. Please Refer to *Appendix V* for more details.

DS1307.h

This header implements structures and definitions used with the DS1307 Real Time Clock. Specific details are provided within the header file itself. Please Refer to *Appendix V* for more details.

HG25504.h

This header implements the structures and definitions used with the Hyundai HG25504 LCD. Specific details are provided within the header file itself. Please Refer to *Appendix V* for more details.

AD22100.h

This header implements the structures, definitions, and functions used with the AD22100 Temperature Sensor. Specific details are provided within the header file itself. Please Refer to *Appendix V* for more details.

Cost Analysis

When designing a simple product such as an alarm clock, the consumer would prefer to buy a product that is low cost yet high efficiency. Our design was far from efficient since parts were acquired through retail sellers as well as being purchased at a unit price which is much higher than bulk price. The details for each item are listed in *Figure 17*.

At first glance, the total cost to mass produce the clock would be roughly \$98 per unit. That would be the bulk price if we purchase the parts from the same retailers that we interacted with during our design stage. A large retail store such as Brookstone would be able to obtain the parts at a lower price and possibly cutting the cost to under \$80 per unit.

Should this product be endorsed completely by Brookstone, many modifications can be made to improve the clock's efficiency and lower the cost. Some older and more expensive chips would become obsolete and outdated, thus newer and cheaper chips can certainly replace the originals. Also, through business associations, Brookstone could potentially obtain discounts that we as designers were unable to obtain.

In addition, these figures do not include the cost for the projected features of the radio chip, amplifiers, antennas, and speakers systems that would have been included in the final product.

Figure 17: Cost Analysis Table

Part	Description	Quantity	Seller Catalog #	Seller	Single Unit Cost	Total Cost	Bulk Cost
MC7805CT	5v Regulator	2	876248	Jameco	\$0.32	\$0.64	\$0.51
577202B04000	Heat Sink	3	577202B04000-ND	Digikey	\$0.73	\$2.19	\$0.80
LM2674-ADJ	Switch	1	290861	Jameco	\$2.85	\$2.85	\$2.20
R270/100-R	270 OHM	1	30606	Jameco	\$0.01	\$0.01	\$0.00
CF1/4W242JRC	2.4kOHM	1	690953	Jameco	\$0.01	\$0.01	\$0.00
K103K15X7RF5TL2	0.01 uF/50v	1	BC1078TR-ND	Digikey	\$0.07	\$0.07	\$0.06
UHE1V151MPT	150 uF/35v	1	UHE1V151MPT-ND	Digikey	\$0.07	\$0.07	\$0.07
2209-H-VP	47uH	1	371101	Jameco	\$2.10	\$2.10	\$0.85
MBR340IR-ND	Diode 40v/3A	1	MBR340IR-ND	Digikey	\$0.53	\$0.53	\$0.16
FL-4W	4"CCFT LAMP, WHITE	1	FL-4W	All Electronics	\$6.78	\$6.78	\$6.78

[Embedded Systems Design Laboratory: Spring 2008: Team 3A: Brookstone]

INV-4	INVERTER	1	INV-4	All Electronics	\$6.80	\$6.80	\$6.80
OSC10H	10MHZ Oscillator	1	102779	Jameco	\$1.75	\$1.75	\$0.94
MC68HC908JL16	uP	1	MC908JL16CSPE-ND	Digikey	\$3.90	\$3.90	\$2.25
DBX05-PN-R	Buzzer	2	138740	Jameco	\$1.49	\$2.98	\$1.50
DS32KHZ	32KHZ Oscillator	1	DS32KHZ/DIP-ND	Digikey	\$1.85	\$1.85	\$1.02
DS1307	Real Time Clock Chip	1	1194679	Jameco	\$3.59	\$3.59	\$2.43
CF1/4W103JRC	10kOHM	2	691104	Jameco	\$0.01	\$0.02	\$0.01
24LC256	256KB EEPROM	1	24LC256-E/P-ND	Digikey	\$1.90	\$1.90	\$1.00
DM74LS174	Hex D flip-flop with clear	1	46931	Jameco	\$0.79	\$0.79	\$0.44
DM74LS32	2-input OR gate	1	47466	Jameco	\$0.35	\$0.35	\$0.25
DM74LS08	Quad 2-input AND gate	1	46375	Jameco	\$0.37	\$0.37	\$0.17
DM74LS04	Hex inverter	1	46316	Jameco	\$0.41	\$0.41	\$0.23
DM74LS125A	Quad bus buffer neg. enable tri-state	1	46501	Jameco	\$0.36	\$0.36	\$0.23
WRL-07813	Wireless Tx/Rx Set	1	WRL-07813	Sparkfun	\$13.95	\$13.95	\$11.16
EVQ-PF008K	Push buttons	9	865101	Jameco	\$0.25	\$2.25	\$1.58
R13-66A-B-02-R	Switch	2	316014	Jameco	\$1.25	\$2.50	\$0.75
R330/100	330OHM	3	30867	Jameco	\$0.01	\$0.03	\$0.01
CF1/4W475JRC	4.7kOHM	3	691737	Jameco	\$0.01	\$0.03	\$0.01
EDE-1144	Keypad Encoder	1	171969	Jameco	\$6.99	\$6.99	\$4.46
TQOPD084M0000A3	4MHZ Oscillator	1	325526	Jameco	\$1.65	\$1.65	\$1.20
2.5 MM DC POWER JACK	Power jack, 2.5mm	1	DCJ-6	All Electronics	\$0.25	\$0.25	\$0.10

[Embedded Systems Design Laboratory: Spring 2008: Team 3A: Brookstone]

3-12V 1000mA	3V-12V AC-DC power adapter	1	273-029	Radio Shack	\$18.99	\$18.99	\$18.99
CR2032	3v Battery	1	N189-ND	Digikey	\$0.35	\$0.35	\$0.19
103K-ND	Battery holder, 20mm	1	103K-ND	Digikey	\$1.55	\$1.55	\$0.67
AD22100	Temperature Sensors	2	AD22100KT-ND	Digikey	\$3.22	\$6.44	\$6.44
ECE-A1HKG0R1	0.1uF	2	P925-ND	Digikey	\$0.18	\$0.36	\$0.06
R1.0K/100	1KOHM	2	29663	Jameco	\$0.01	\$0.02	\$0.01
HG25504	Hyundai HG25504	1	LCD-101	All Electronics	\$18.50	\$18.50	\$18.50
UHE1V221MPD6	220uF	1	493-1579-ND	Digikey	\$0.52	\$0.52	\$0.25
ECA-1VM100	10uF	3	P5161-ND	Digikey	\$0.35	\$1.05	\$0.51
CE-0381	DC-DC inverter	1	DC-32	All Electronics	\$0.50	\$0.50	\$0.25
R120/100-R	120OHM	1	30083	Jameco	\$0.01	\$0.01	\$0.00
3352W-1-103LF	Potentiometer	2	770355	Jameco	\$1.00	\$2.00	\$1.50
ECE-A1HKG010	1uF	1	P931-ND	Digikey	\$0.18	\$0.18	\$0.03
LM337	Neg Voltage Regulator	1	LM337TNS-ND	Digikey	\$2.06	\$2.06	\$0.80
MC68HC908QT4	uP	1	964361	Jameco	\$2.51	\$2.51	\$1.16
BH9V-W-ND	Battery holder, 9V	1	BH9V-W-ND	Digikey	\$0.86	\$0.86	\$0.62
					Total	\$123.88	\$97.97

Conclusion

Although the design is still incomplete, the results are extremely promising. The design still calls for the radio feature to be included in the final product. Also, it is important to note that given more time, the project would undergo more changes that would improve the efficiency, as well as the overall cost. The design calls for a wider range of wireless communication, thus a better antenna would be installed during optimization.

Many conflicts that occurred came equally from hardware and software situations, thus optimization of both would certainly improve the product's value. Hardware wiring and component placement on the board would allow the designer to compress the physical space the clock would need as well as reducing the chance of creating hardware conflicts such as opening or shorting the circuit. Rewriting the software code would also help the designer in troubleshooting problems easily as well as build new codes for future features that would be implemented.

Future implementation could be the inclusion of music streaming and iPod compatibility. Once the radio is installed, the clock can be selected between the radio mode or music streaming/iPod option. Given more time, this feature would have been possible to implement, but it was not a high priority thus it was not attempted.

Appendix I: Power Consumption Analysis

This section will discuss how much power each major component or subsystem in the clock consumes during operation. Power is barely consumed by the clock itself, roughly 2.449 watts max for the main board, while the wireless board only takes up about 0.406 watts max. These figures are calculated under the absolute maximum electrical characteristics for their respective chips. Under optimal condition, the main board and the wireless board would consume only fractions of the stated values.

When viewing this analysis, it is also important to mention that 1.5 watts from the 2.449 total watts is used by the 4" lamp. When the lamp is not operating the main board would drain at most 1 watt. But once the lamp turns on, the main board would consume more than double its original power. This again is considering the worse case of absolute maximum conditions.

Figure 18: Power Consumption Analysis Table

Board	Part	Description	Voltage(V)	Current(mA)	Power (W)
Main Board					$P = I * V \text{ (A*V)}$
	OSC10H	10MHZ Oscillator	5	45	0.225
	MC68HC908JL16	uP	5	18	0.09
	DBX05-PN-R	Buzzer	5	40	0.2
	DS32KHZ	32KHZ Oscillator	5	0.115	0.000575
	DS1307	Real Time Clock Chip	5	0.5	0.0025
	24LC256	256KB EEPROM	5	3	0.015
	DM74LS174	Hex D flip-flop with clear	5	16	0.08
	DM74LS32	Quad 2-input OR gate	5	4	0.02
	DM74LS08	Quad 2-input AND gate	5	3.4	0.017
	DM74LS04	Hex inverter	5	2.5	0.0125
	DM74LS125A	Quad bus buffer neg. enable tri-state	5	11	0.055

[Embedded Systems Design Laboratory: Spring 2008: Team 3A: Brookstone]

HG25504	Hyundai HG25504			
	*Logic	5	10	0.05
	*LCD (VDD-V0)	15.5	5	0.0775
FL-4W	4"CCFT LAMP, WHITE	6	250	1.5
EDE-1144	Keypad Encoder	5	1.8	0.009
TQOPD084M0000A3	4MHZ Oscillator	5	15	0.075
AD22100	Temperature Sensors	5	0.5	0.0025
WRL-07813	Receiver	5	3.5	0.0175
			Total	2.449075

Wireless Board

AD22100	Temperature Sensors	5	0.5	0.0025
WRL-07813	Transmitter	9	41	0.369
MC68HC908QT4	uP	5	6.8	0.034
			Total	0.4055

Appendix II: Project Screenshots & Pictures

The following pages contain Screenshots and other pictures of the entire project including the Main Clock and the Wireless Temperature Module.

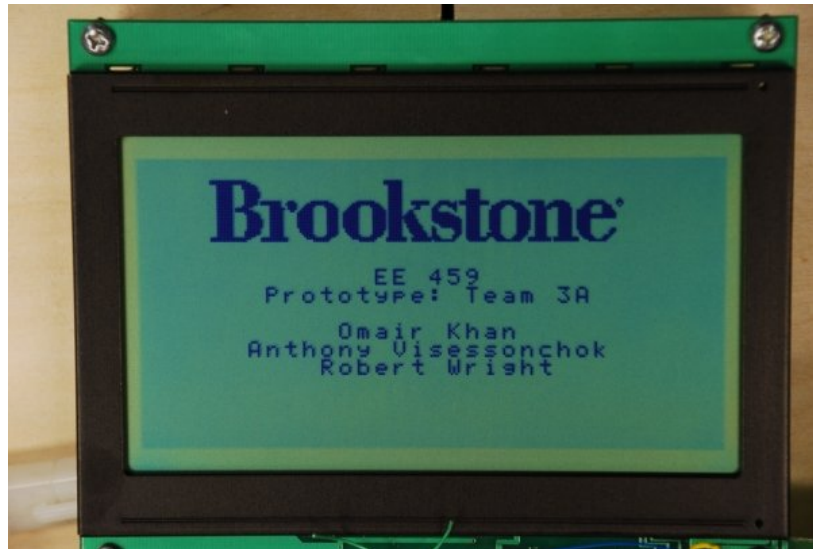


Figure 19: Welcome Screen

The welcome screen when clock is first powered on, showing brand name and engineering team names.

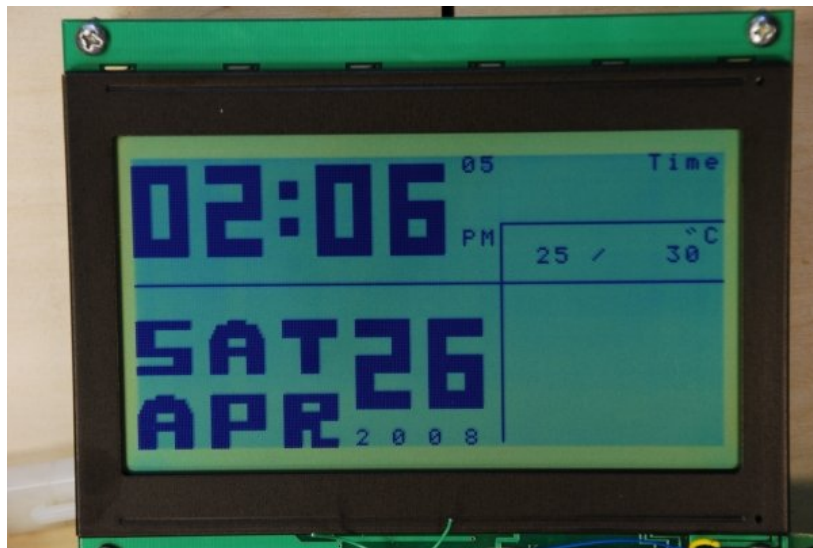


Figure 20: Main Screen with Alarms off and Celsius Temp

The main display of the clock shows the time, date, and the two temperature displays.

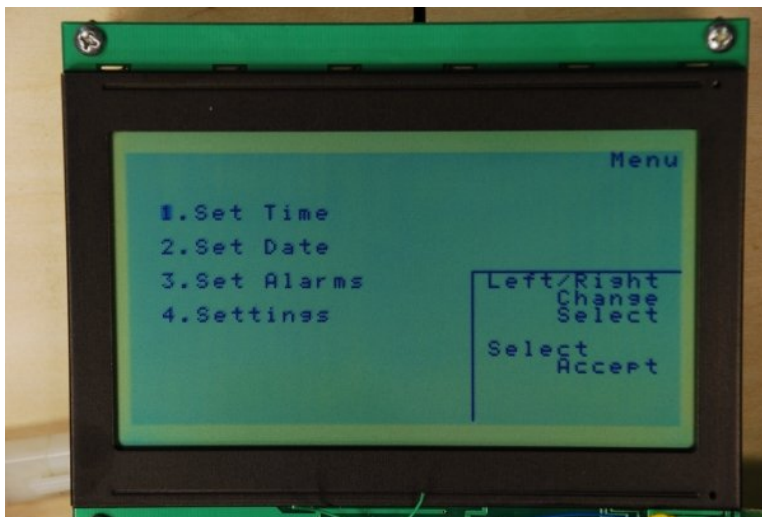


Figure 21: Menu Screen

The menu screen displays the four submenus as well as the directions for scrolling

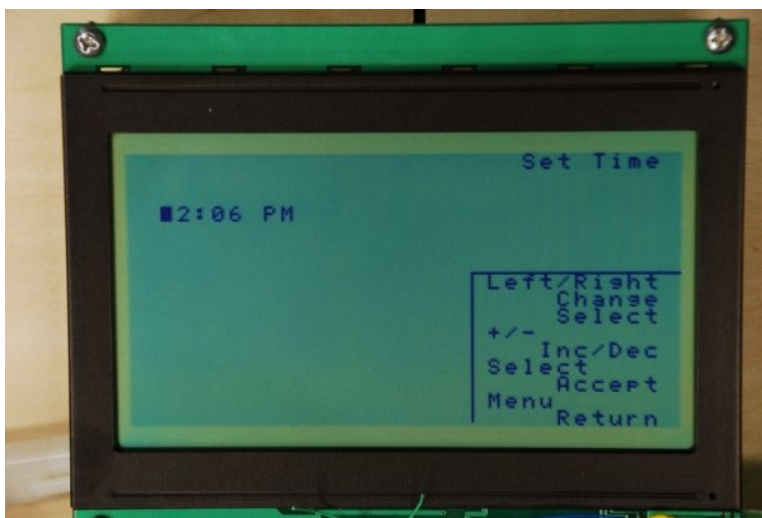


Figure 22: Set Screen Time

The "Set Time" submenu screen allows the user to set each hour and minute digit as well as the AM/PM



Figure 23: Set Date Screen

The "Set Date" submenu screen allows the user to set the month, day, year, and day of week individually.



Figure 24: Set Alarms Screen

The "Set Alarms" submenu screen allows the user to set seven alarms tied to a specific day, much like the "Set Time" submenu as well as a general alarm that can be set to sound each day.

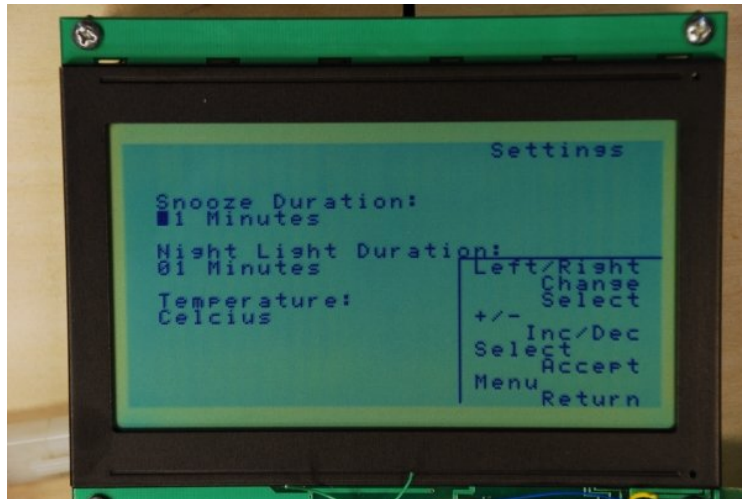


Figure 25: Settings Screen

The “Settings” submenu display allows the user to set the snooze duration, the night light duration and whether the temp should display in Fahrenheit or Celsius.



Figure 26: Saved Screen

The “Saved” screen flashes after the user has made a change in any of the submenus and then presses the select button to save it.



Figure 27: Main Screen with Alarm On and Fahrenheit Temp

Newly saved main display with the updated time and date as well as the alarm on notification and the temperature changed to degrees Fahrenheit.



Figure 28: Main Screen with Wireless Module turned off

Main display when wireless module is turned off displaying an "ERROR" signal where the temperature usually is located.

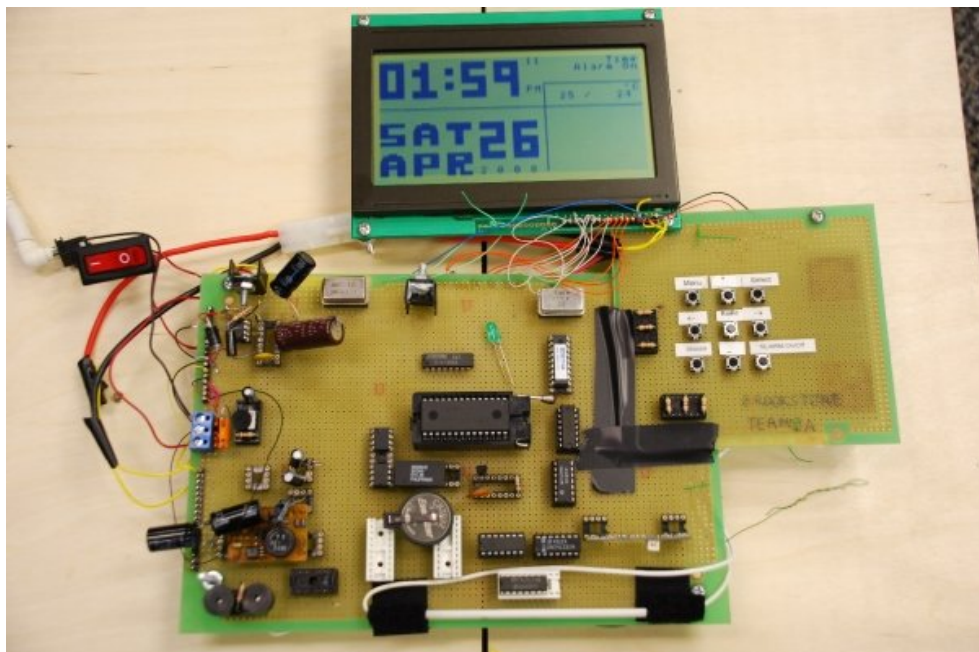


Figure 29: Main Project board

Entire project with attached hardware and keypad.

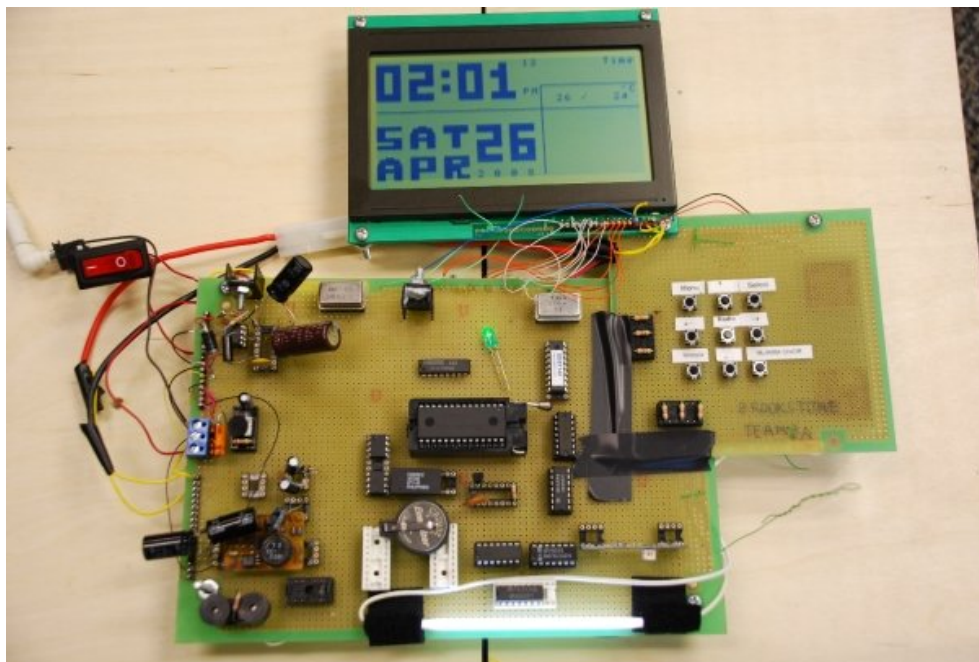


Figure 30: Main Project Board with Night Light On

Entire project with hardware and keypad attached with illuminated night light.

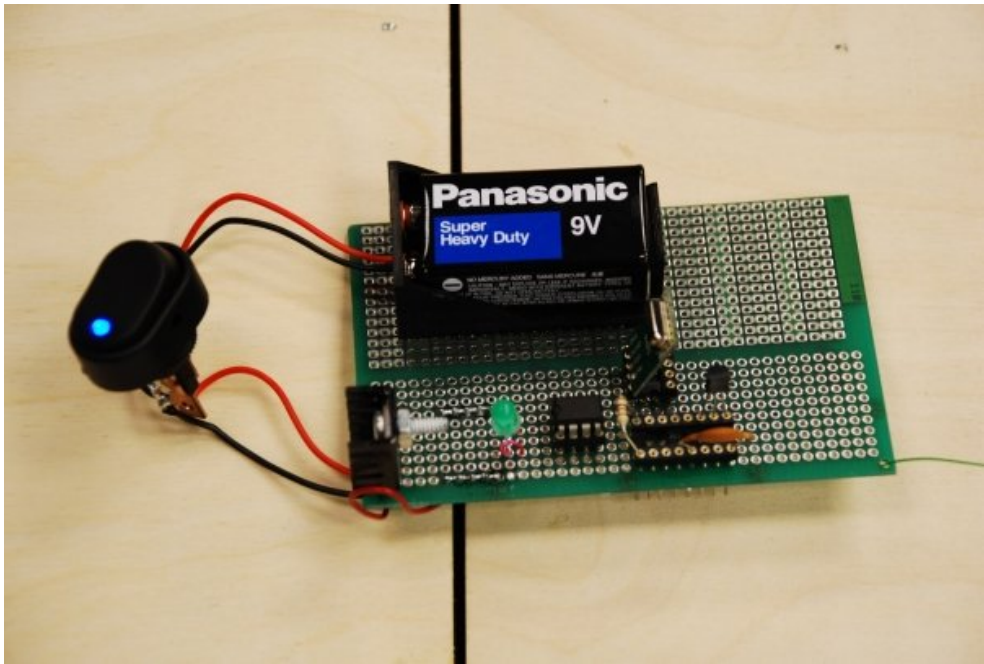
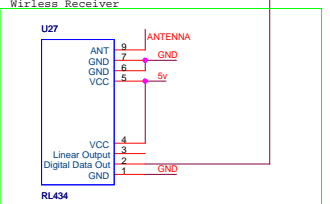
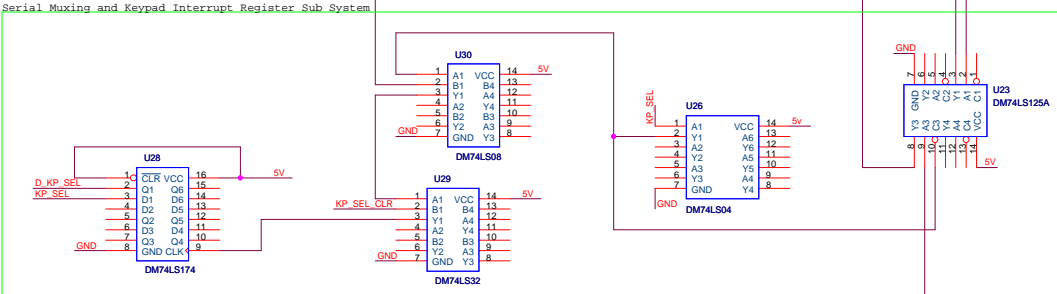
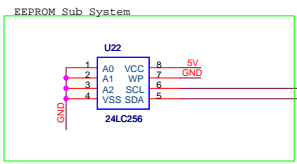
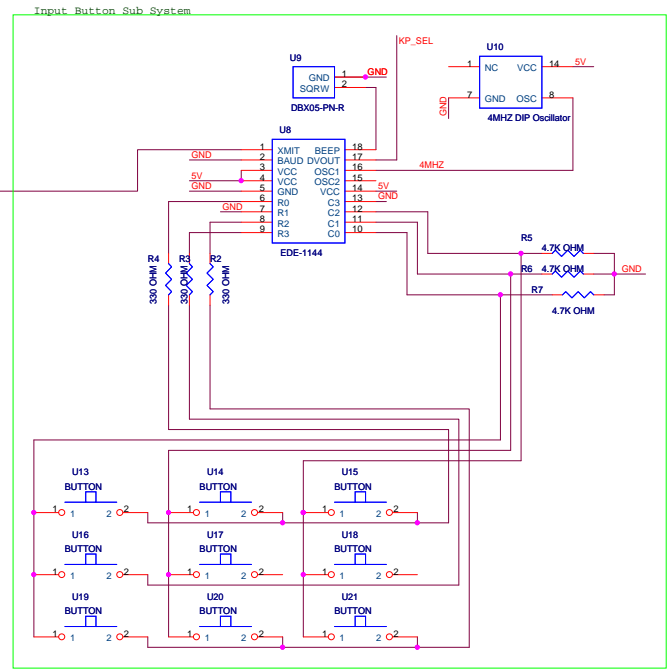
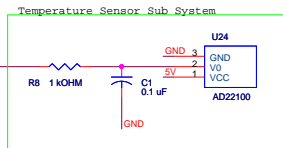
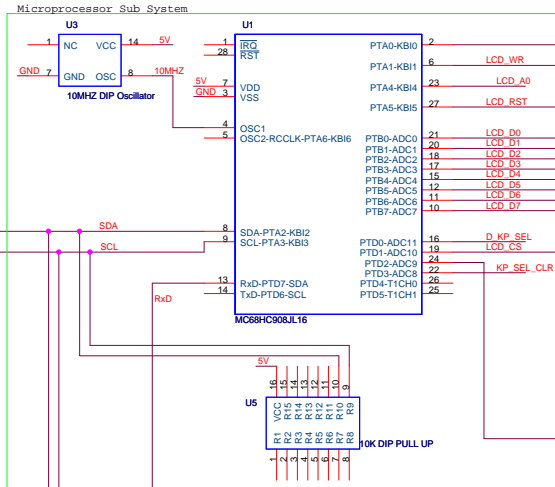
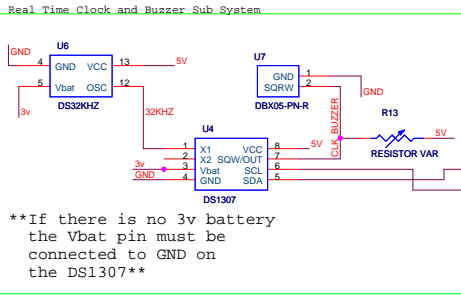
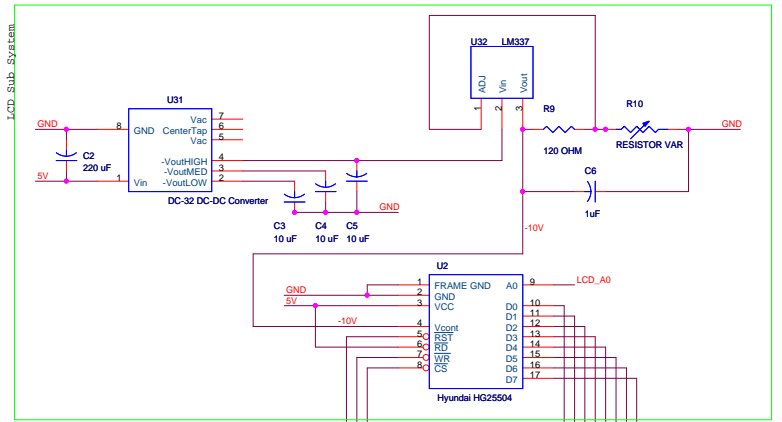
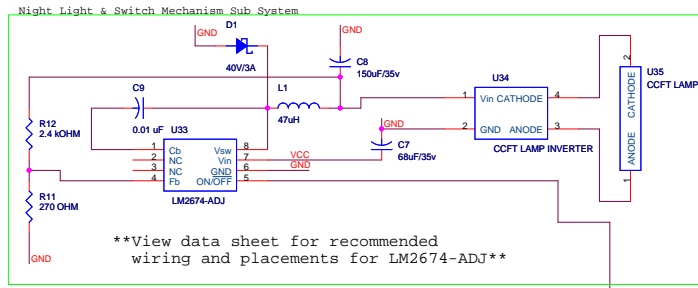
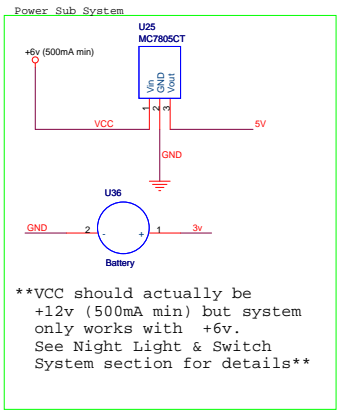


Figure 31: Wireless Temperature Module

Appendix III: Main Clock Schematics

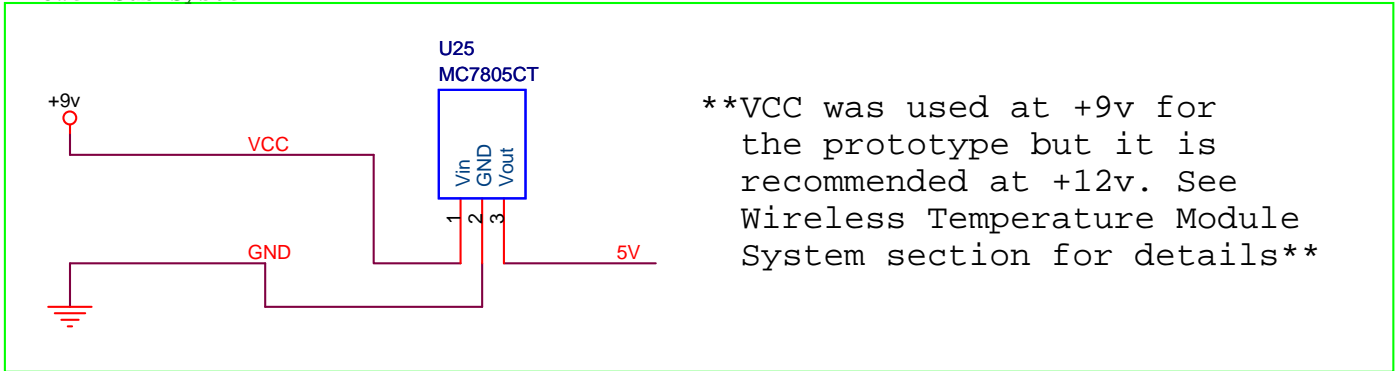
The following pages contain the schematic for the Main Clock.



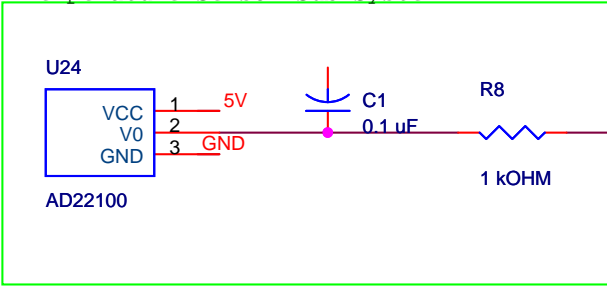
Appendix IV: Wireless Temperature Module Schematics

The following pages contain the schematics for the Wireless Temperature Module.

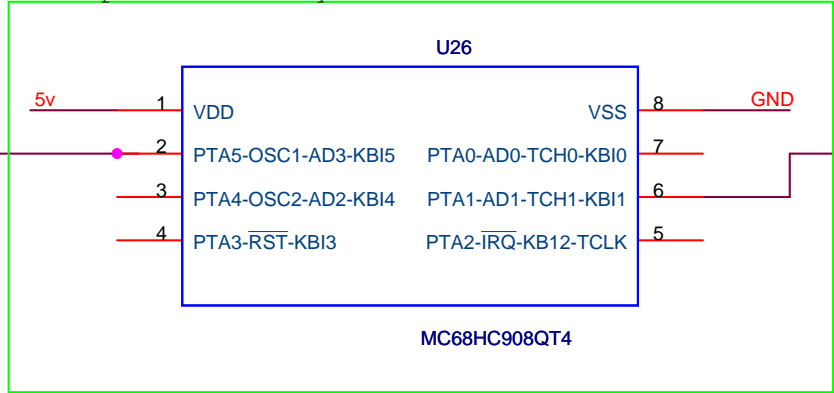
Power Sub System



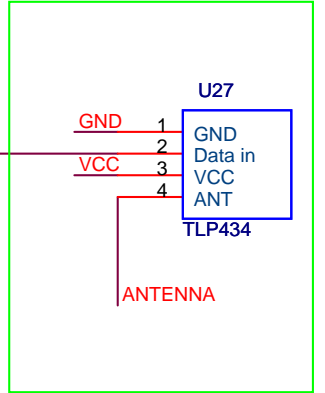
Temperature Sensor Sub System



Microprocessor Sub System



Wireless Transmitter



Title
Temperature Module

Size A Document Number
EE459 - 04

www.omair-khan.com

Rev
<RevCo

Date: Thursday, May 08, 2008

Sheet 1 of 1

Appendix V: CD

The attached CD on the back cover contains a copy of all the source codes used in this project:

- 1) Main Clock Source Code
- 2) Wireless Temperature Module Source Code
- 3) 24LC256.h
- 4) DS1307.h
- 5) HG25504.h
- 6) AD22100.h

The CD also contains a PDF version of this report.

Appendix VI: Marketing Team

Geeta Arora
Miquel Vasquez
Adrienne Ng
Lila Coghen

Appendix VII: Member Contributions

	Omar Khan	Anthony Visessonchok	Robert Wright
System Design	33.33%	33.33%	33.33%
Hardware Design	40.00%	30.00%	30.00%
Hardware Assembly	30.00%	35.00%	35.00%
Hardware Checkout	60.00%	25.00%	15.00%
Software Design	95.00%	2.50%	2.50%
Software Checkout	90.00%	5.00%	5.00%
System Integration	90.00%	5.00%	5.00%
Final Presentation	33.33%	33.33%	33.33%
Documentation / Final Report	40.00%	30.00%	30.00%
Interacting with the Marketing Team	10.00%	35.00%	55.00%
Date			
Signature			