DEVELOPMENT OF A COLOR IDENTIFICATION DEVICE FOR THE BLIND

Project Teams:

Fall 2000	**Spring 2000**	**Fall 2001**
Christine Paulson	Christine Paulson	Nils Kapp
Martin Weiner	Martin Weiner	Dmitriy Kogan
		Christine Paulson
		Aaron Stratton

Advisor: Dr. David Harris

December 21, 2001

ABSTRACT

Currently, the blind depend on various methods for matching clothing. These include memorizing placement in a closet, reading Braille tags, and relying on friends and family for help. For the purpose of alleviating these dependencies, the goal of this project is to engineer a speaking device that identifies the color of a piece of material placed in front of it.

This project involves two components. The first is the development and calibration of a sensor to obtain a well-defined output for each color. The second is the development of a system that takes data from the sensor, determines the color from the calibration, and announces the color to the user.

A functional prototype was developed during the second semester of the project that correctly identifies greater than 90% of the colors used for calibration. The prototype is significantly less accurate on new fabrics and colors. The speech playback capabilities are effective.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Currently, the blind depend heavily on their friends, family, and their memory to match clothing. During the summer of 2000, Sheena Iyengar, a blind professor at Columbia University, suggested that a talking color identification device would be extremely helpful to her and other people who are visually impaired.

The goal of the project is to develop a handheld device to detect the color of a piece of fabric and announce the color to the user. The project is divided into two parts: color detection, and color readout. After two semesters of working on the project, both the color detection and the voice readout portions were functional, though further developments and refinement were necessary. The goal of the third semester was to explore new approaches that would result in higher color recognition accuracy.

The purpose of this report is to document the design, calibration, and effectiveness of the devices made. It is organized to demonstrate the design process, development, calibration and evaluation of the prototypes.

2. BACKGROUND INFORMATION

2.1. HUMAN PERCEPTION OF COLOR

The perception of color depends on the physical characterization of light entering the eye, as well as physiological processes after light enters the eye. Within the human eye, the retina contains cone cells, which are sensitive to a specific range of light wavelength, and rod cells, which are sensitive to the intensity of light. The rod cells allow sight in very low intensity light conditions. Since only the rod cells function during very low light conditions, no differentiation in color can be detected. In bright light conditions, the cone cells play the predominant role.

There are three types of cone cells, each sensitive to a specific range of light wavelength, as shown in Figure 1. Each type of cone cell is sensitive to red (~700nm), green (~500nm), or blue (~400nm) light. For any specific wavelength of visible light, the cone cells in the eye respond, giving information to the brain about what color is visible. For example, yellow light, which has a peak of approximately 600nm, stimulates both red and green cone cells. The brain interprets the response of the cone cells as the color yellow. For this reason, any color can be determined from its red, green, and blue components.

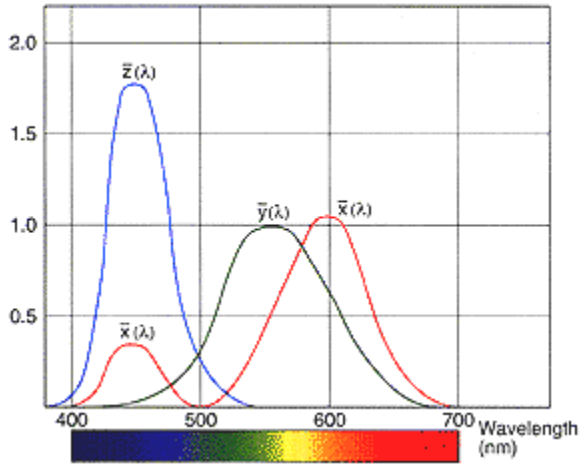

Figure 1: Wavelength Sensitivity of the Three Types of Cone Cells in a Human Eye

Although the cone cells are the "color-seeing" cells, it turns out that the rod cells also respond differently to different wavelengths of light. The rod cells respond much more strongly to blue light than the cone cells. The cone cells on the other hand, respond to other colors of light, such as deep red, that the rod cells are blind to. For this reason, an object may appear to change color when moved from low intensity lighting to high intensity lighting.

The color identification device seeks to mimic the cone cells' ability to detect colors by determining the amounts of different color components of light reflected off of a piece of cloth. It is assumed that the user is only interested in the color of fabric in adequate lighting, and for this reason, the intensity measurement of the rod cells is ignored.

3. FIRST SEMESTER DEVELOPMENT

3.1. FIRST SEMESTER CONCEPTUAL DESIGNS

All colors have unique absorption spectra, which can be used to uniquely identify a color. The basic concept behind first semester prototypes is that by measuring the intensity of color components of light reflected off of a piece of fabric, one can determine the color of the fabric. First semester prototypes employ the RGB (red, green, blue) color definition model and one of two basic methods of measuring the color components of light reflected from a piece of fabric. White light contains equal amounts of each color component of light. The prototype shown in Figure 2 demonstrates reflecting white light off of fabric, and filtering the reflected light into red, green, and blue components at the detection stage. In Figure 3, the prototype emits three flashes of colored light. The intensity of each reflection is measured.

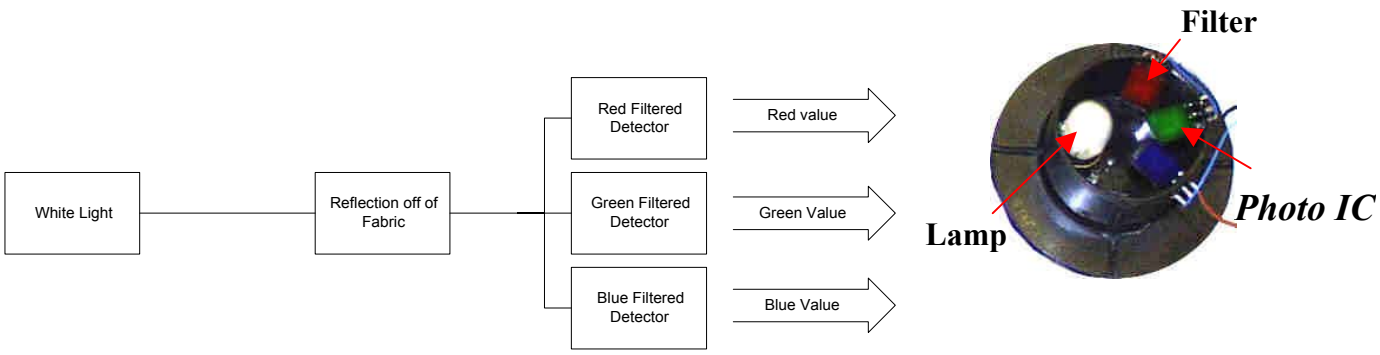

Figure 2 A First Semester Prototype that Filters Reflected Light at Detection

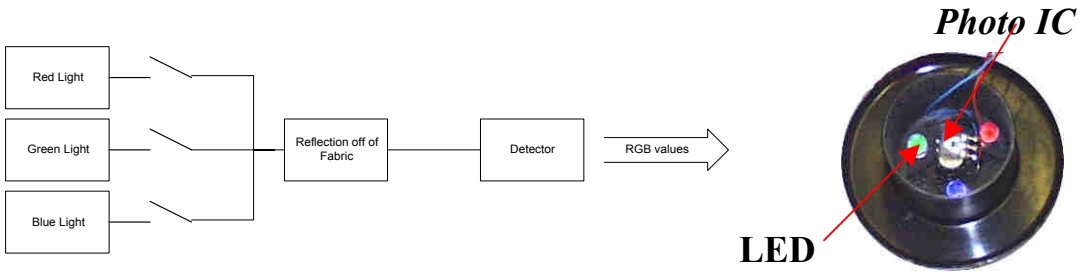

Figure 3 A First Semester Prototype that Filters Light Before Reflection

3.2. FIRST SEMESTER RESULTS

Preliminary prototypes were developed to validate the basic conceptual design, and to evaluate the feasibility of the two approaches. Much effort went into experimenting with different types and configurations of light sources and photo detectors, as well as the developing an effective color identification algorithm. To identify colors, a 'color look-up table' was generated. A data set, associating RGB intensities with fabric colors, was generated from a variety of fabric types and colors.

The final prototype for the first semester is shown in Figure 4. The prototype identifies colors with good repeatability, but fairly low accuracy. For more information about the first semester design, please refer to the first semester documentation report.

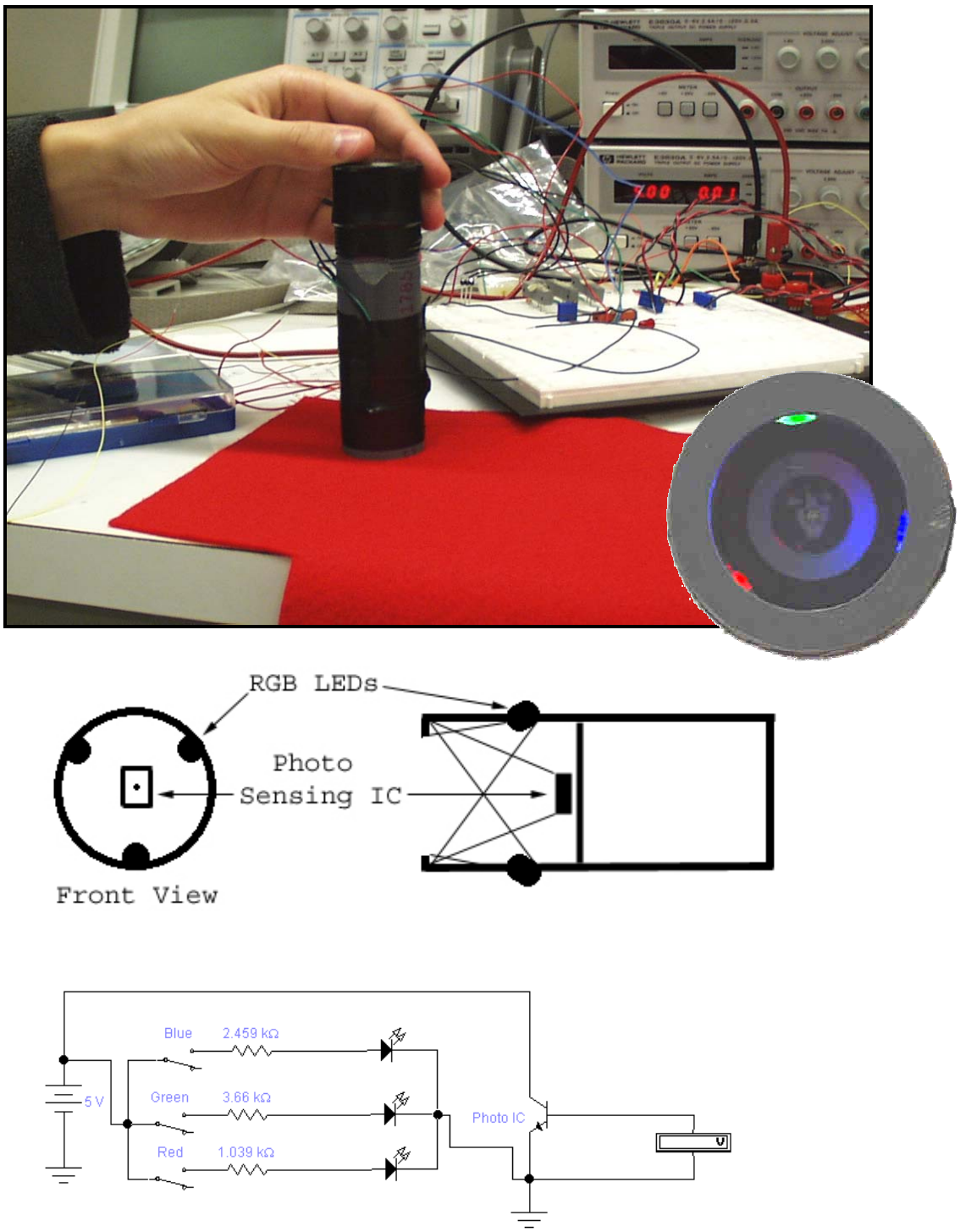

Figure 4 Final First Semester Prototype

4. SECOND SEMESTER DEVELOPMENT

4.1. SECOND SEMESTER GOAL

The second semester goal was to produce a prototype that demonstrated all of the key functions of the device and good color identification capability. This involved the development of: (1) device housing, light emission and detection scheme, and analog electronics, (2) an improved algorithm for color identification, and (3) digital electronics to control the device, store calibration data, and implement the color identification algorithm. This prototype would be sent to Dr. Sheena Iyengar for evaluation.

4.2. SEVEN LED PROTOTYPE – "BLINKY"

At the end of the first semester, the most feasible light emission and detection scheme involved the use of colored LEDs and a photo IC. It was unclear as to whether the emission spectra of the colored LEDs should be narrow and distinct, or broad and overlapping so that all wavelengths in the visible range were expressed. The final experiment in the first semester involved a prototype with three LEDs of very narrow and distinct emission spectra. The color identification capabilities of this prototype turned out to be extremely poor. From these results, it was speculated that the second scheme, the use of LEDs with overlapping and broad emission spectra, would yield higher color identification accuracy.

To test this hypothesis a new prototype ("Blinky") was made that implemented seven LEDs with overlapping emission spectrum. The design involved increasing the number of LEDs because it was not possible to find only three LEDs with broad enough emission spectrums to cover the entire visible spectrum. The LEDs selected are shown in Table 1. The "low" and "high" wavelengths in the table are wavelengths expressed by the LED at half the peak intensity. Figure 5 illustrates an approximate of the LED emission spectra. The packaging of Blinky was similar to that of the first semester prototype, except that seven LEDs formed a circle around the photosensor rather than three.

Manufacturer	Part Number	Low	Peak	High	Intensity	quant. 100	Distributor
Lumex	SSL-LX5093SRC/DV	645	660	670	1300	$0.23	Digikey
Dialight	559-6101-00x	620	635	650	2100	$2.38	Newark
Lumex	SSL-LX5093SOC	595	610	625	2500	$0.83	Digikey
Lumex	SSL-LX5093SYC	575	590	605	1000	$0.59	Digikey
Gilway	E903	510	525	540	5000	$4.26	Gilway
Gilway	E484	450	470	490	2000	$4.50	Gilway
Lumex	SSL-LX5093XUWC	440	550	660	2300	$3.00	Digikey
* orange Lumex burned out, and was replaced with Radio Shack equivalent						**$15.79**	

Table 1: The Seven LEDs selected for "Blinky"

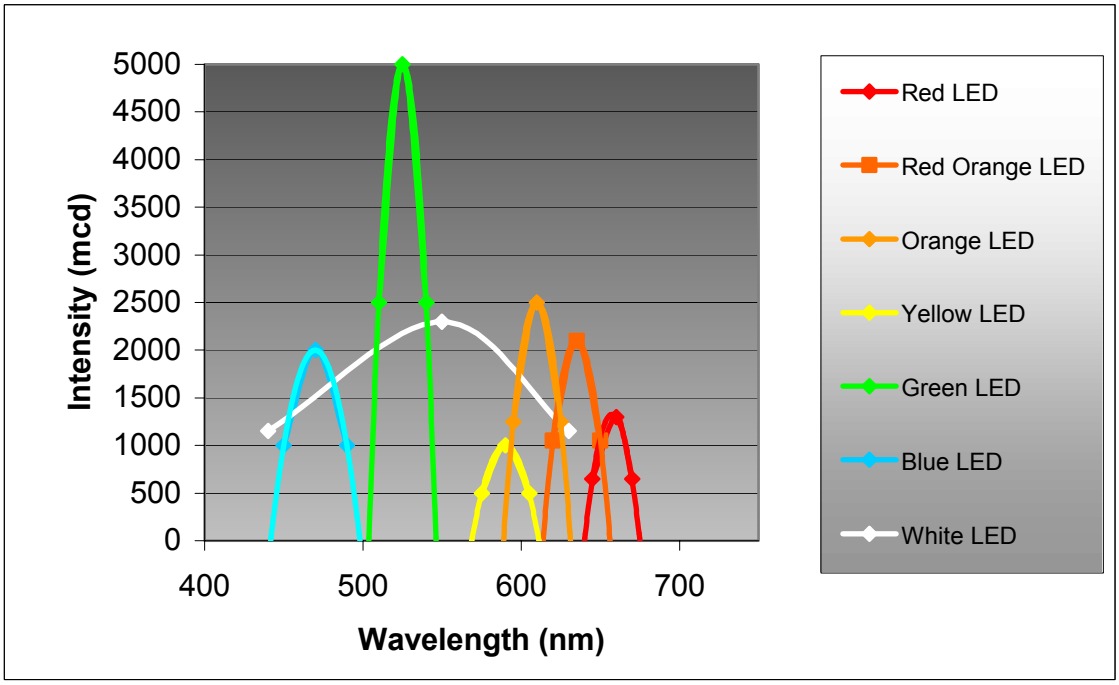

Figure 5: Overlapping LED Emission Spectra (Approximate)

4.3. EFFECTIVENESS OF "BLINKY" PROTOTYPE

The use of seven LEDs dramatically improved the color identification accuracy. The "Blinky" prototype successfully identifies many colors. On testing fabrics used for calibration, it identified 90% correctly. Blinky had difficulty distinguishing dark colors; specifically, dark blue from black and dark purple from dark brown. It identified about 70% of new fabrics correctly. Some of the inaccurate readings were due to inadequate calibration data, and some were due to what is suspected to be overlapping colors in the look up table.

4.4. SPEECH PLAYBACK

A voice record/playback chip is used to record sound clips of color names. The chip is the ISD 2590 ChipCorder, which is available from Winbond Electronics or Digikey. The chip allows one to record up to 90 seconds worth of recordings. Each voice clip is recorded into a specific memory location and can be played back by specifying the 10 bit address. The ChipCorder can be operated either in record or playback mode. By constructing the schematic shown in Figure 6, the user can select either mode with one switch. To record voice clips, set the address pins A0-A9 to the desired address. Check to see that the chip is on, and is in playback mode. Next, hold down the Chip Enable button while speaking clearly into the microphone. To play a certain word, specify the address, set the chip in playback mode, and press the Chip Enable button.

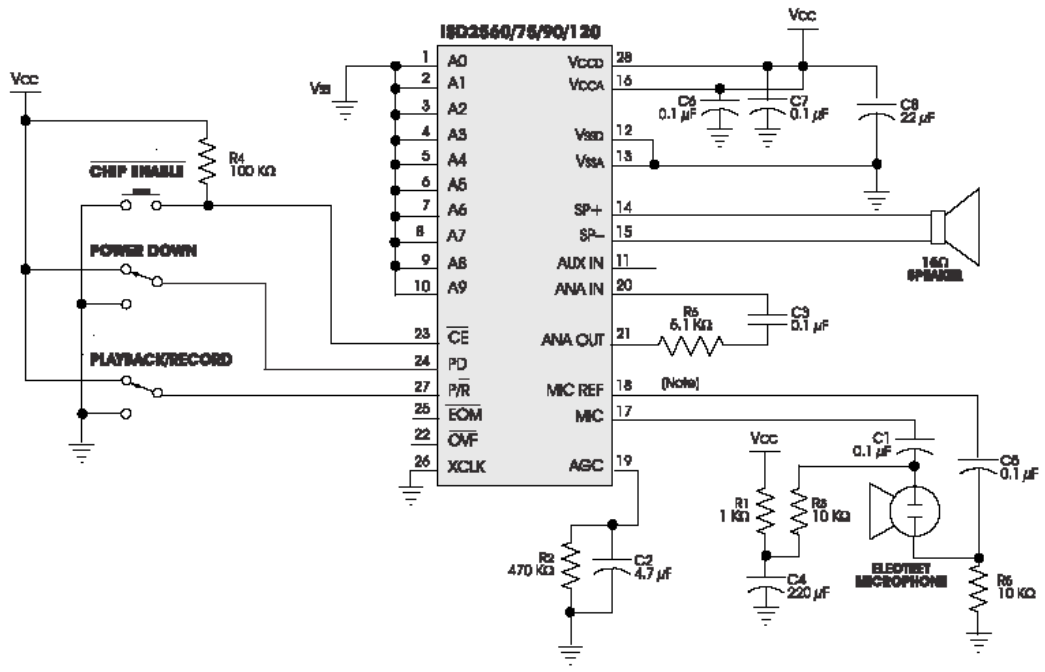

Figure 6: ChipCorder Schematic

When implementing the ChipCorder in the final prototype, the recording circuitry can be removed. Although the ChipCorder has 10 addressing bits, not all of them need to be changed in order to specify a specific voice clip. Some of the bits are used only to specify operation modes. Additionally, the chip is available in either parallel addressing or serial addressing formats. The use of the serial chip can minimize the number of I/O ports needed on a microprocessor. For more information about the operation of the ChipCorder, see the datasheet available at the Winbond website.

4.5. COLOR IDENTIFICATION METHOD

The sensor outputs a voltage corresponding to the intensity of light detected. A microcontroller is needed to interpret the intensity readings, and determine the color of the fabric. Also, the microcontroller is used to turn LEDs on and off at the appropriate times, specify the address of the color voice recording, and initiate the audio playback.

4.5.1. MC8HC908KX8 microcontroller

Motorola's MC8HC908KX8 microcontroller was chosen based on the following favorable features:

- 8 Kbytes of on-chip FLASH memory
- Internal oscillator

- 13 I/O ports
- A/D conversion capability on 4 ports
- Availability of evaluation board kit

The FLASH memory made this microcontroller easy and convenient to use. The program would be uploaded to it via the evaluation board and accompanying software. After programming, the microcontroller could run independently. Several features of this microcontroller minimize the amount of space that the electronics occupy. The internal oscillator eliminated the need for additional clock signal circuitry and the use of only 13 I/O ports made it possible to fit it inside the device housing. A/D conversions are needed to read the analog voltage outputs of the sensor.

4.5.2. Assembly Code and Color Identification Algorithm

The microcontroller utilizes "minimum distance" matching against a previously sampled data set to identify the color of the sampled fabric. A look-up table is stored within the microcontroller, which contains calibration data consisting of sensor readings and the corresponding color address on the ChipCorder. After the device is calibrated, data samples are taken from the material to be identified. The matching algorithm then moves through each data point in the reference set calculating the7-space distance from the new sample to the reference points. When a new minimum distance is found, that distance is saved along with the associated color/sound data. When the end of the data table is reached, the program interfaces with the speech playback chip to play the appropriate color information. See Appendix A for complete listing of the code.

5. THIRD SEMESTER DEVELOPMENT

5.1. THIRD SEMESTER GOAL

The goal of the third semester was to improve the accuracy of the color recognition capabilities. Further research into color recognition theory yielded new color recognition methods to be explored. A prototype of a spectrophotometer was made. Additionally, the use of filtered photodiodes was revisited.

5.2. SPECTROPHOTOMETER PROTOTYPE – "PINKY"

There are basically two different types of devices for analyzing what we perceive as color: colorimeters, and spectrophotometers. The purpose of a colorimeter is to measure the color of objects or light. Colorimeters are mostly used in the quality control of consumer products, i.e. products where color matters. Spectrophotometers on the other hand examine the entire spectrum at once. They are generally used for scientific applications, where the actual spectral response is of importance. It is possible to derive a color reading form the spectral

information. Although, theoretically a spectrophotometer overkill for this application, we considered the possibility of building one. If we could produce a spectrophotometer for a price comparable to a colorimeter, we could use the more powerful spectrophotometer to determine the color of our samples. One of our hopes was that it would also help us solve the problem of determining the color of dark samples, since the shape of the spectral curve determines the color more than the actual intensity itself.

Our approach to implementing an inexpensive spectrophotometer was to use a diffraction grating to split up a beam of light onto a CMOS camera array. The camera array had a USB interface and came as a "plug-n-play" device, so we did not have to worry about the particulars of interfacing to the array itself. A sufficient grating cost about $10 per square foot. The cost of the basic components was comparable to the implementation of a colorimeter. However, we soon discovered that this implementation was not feasible. The reason was that the intensity of the light reflected form the sample was not enough to drive our detector assembly directly. To use this approach we would have had to add optical elements to gather more reflected light, however the cost of such devices of a high enough quality is prohibitively high. Therefore we had to abandon this approach. Although the camera was supposedly a color camera, it was impossible to use it directly, since its color resolution was extremely poor.

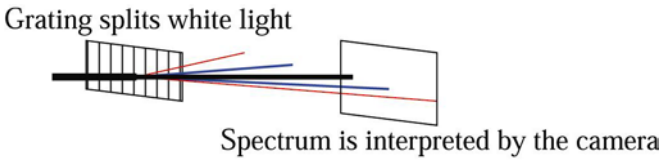

Figure 7: Diagram of the "Pinky" Spectrophotometer Prototype

5.3. FILTERED PHOTODIODE PROTOTYPE – "INKY"

Hamamatsu sells photodiodes that are intended for color recognition use. A set of three photodiodes (S6528, S6529, S6530 or S7505), which are shown in Figure 8, can be used measure red, green, and blue light intensities. After learning about these photodiodes, the method of filtering light into its color components at the detection stage was revisited.

Figure 8: Hamamatsu Photodiodes

The problem with using LEDs, as in "Blinky", the second semester prototype, is that it is difficult to obtain a set whose emission spectra cover the entire visible wavelength range. These photodiodes have wider, overlapping spectra, as shown in Figure 9.

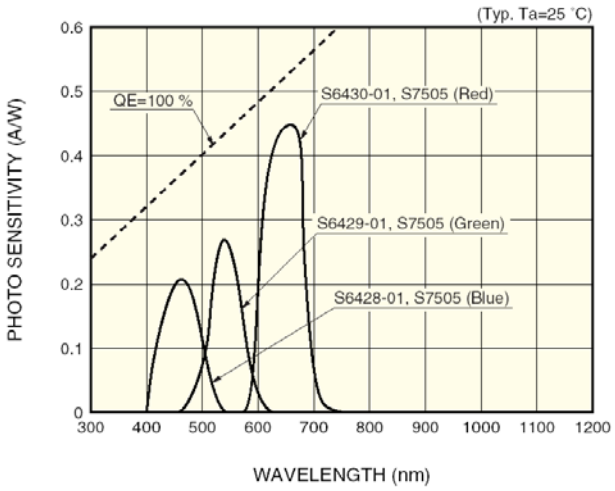

Figure 9: Wavelength Sensitivity of the Hamamatsu Photodiodes

Two ultra-bright white Lumex LEDs were used to provide the light that reflects off of the fabric. However, these "white" LEDs emit a distinctly bluish light. It was assumed that this would not be a problem due to the fact that the photodiodes are less sensitive at the blue end of the spectrum. Since the white LEDs did produce wavelengths spanning the entire visible range, the tendency toward blue was not a problem.

The Inky prototype was constructed as shown in Figure 10. The microcontroller was used to sample voltages from the three sensors. The color identification algorithm was modified from a 7-space minimum distance calculation to accommodate 3-space.

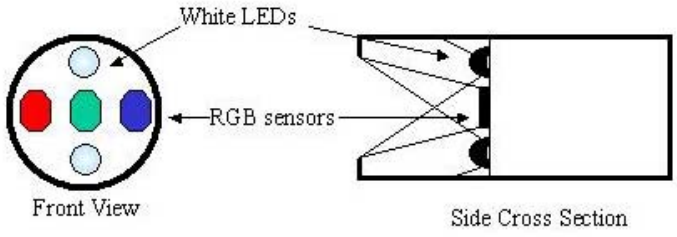

Figure 10: Filtered Photodiode Prototype – "Inky"

5.3.1. Preliminary Amplifier Circuit

The photodiodes output a current that is related to the intensity of light. This current output must be converted to a voltage ranging from approximately 0 to 5 volts, which corresponds to the A/D range of the microcontroller. The amplifier circuit shown in Figure 11 performs this task. The first stage amplifier is an adaptation of an ideal current to voltage converter. Since the output of the first stage ranges from 0.4 V to 1.55 V, a second stage is needed to reduce the bias and increase the range.

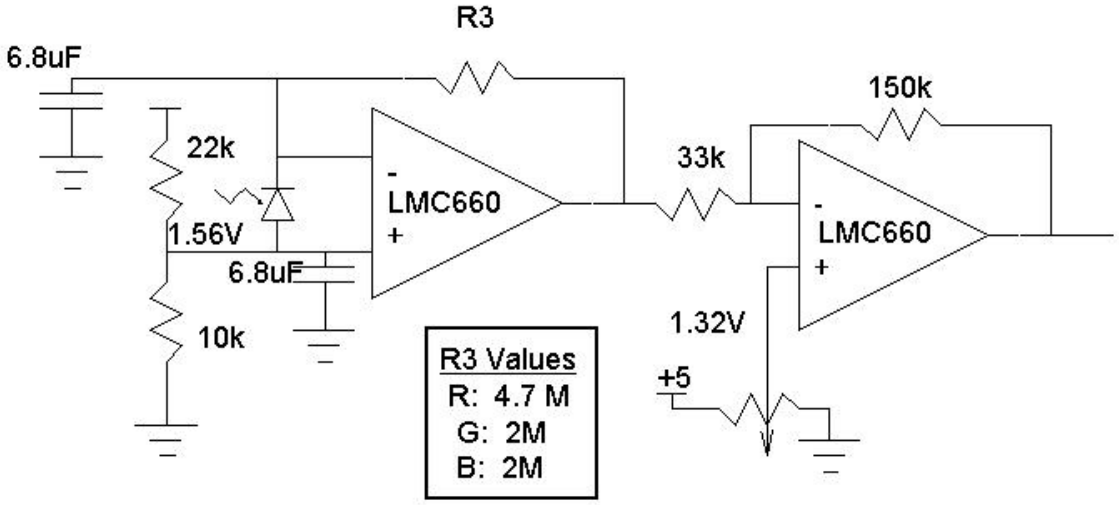

Figure 11: Preliminary Amplifier Circuit

5.3.2. Data Interpretation

The main problem with the previous design, "Blinky", was its inability to reliably differentiate dark colors; specifically dark blues and blacks. To this end, one of the main goals of the new design has been to improve the accuracy on dark fabrics. A large amount of data focusing on dark colors was taken. Data was taken using a multimeter in order to obtain values that were more precise. The goal was to determine if any overlap existed between these two regions. If the output from final amplification stages is graphed as RGB triplets in 3-space, several color regions emerge, as shown in Figure 12.

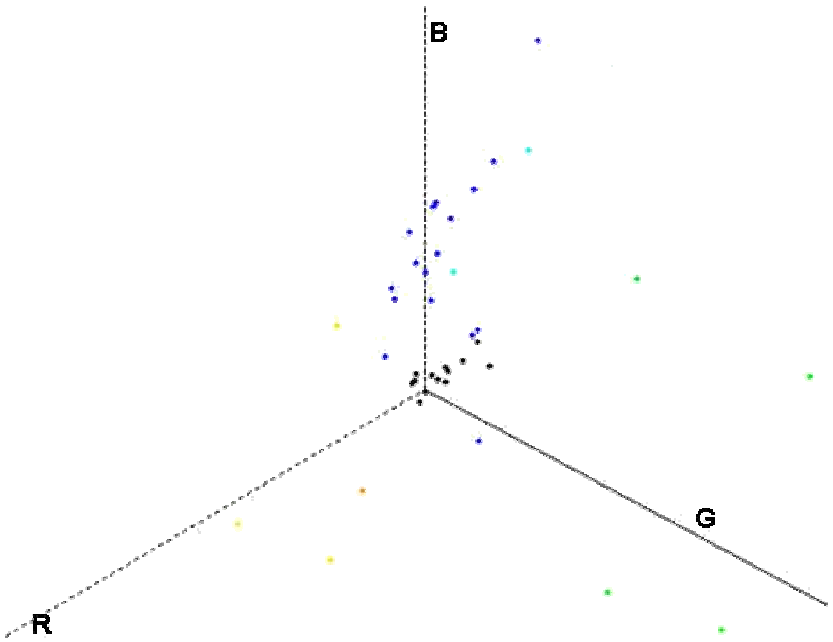

Figure 12: Inky's Dark Blue and Black Data Clusters

Examining the black and blue clusters, it is evident that the two color regions do not intersect. However, the minimum separation between the two is less than 20mV. It is known from the data collection that variations of ±1 in the least significant bit of the output from the A/D convert occur regularly as a result of uncertainty and noise. With an 8-bit A/D converter working from 0-5V, the minimum resolution is (5V) / (2^8) = 0.0195V. It is possible that normal variation and uncertainty will result in confusion of colors near the blue/black boundary.

5.3.3. Reproducibility

While taking samples, it was decided to test the reproducibility of the Inky prototype. After taking a set of data, a smaller set was chosen to resample to see how closely they would coincide with the first readings. The readings were reproducible to within ±1 of the least significant bit. This error is as large as the minimum distance found between dark blue and

black data. This is a potential problem. For this reason, it is suspected that dark blues and blacks may occasionally overlap.

It was noticed that in order to get more consistent readings, the material had to be placed flat on a table. As soon as the geometry of the material changed with respect to the LEDs and the photodiodes, the variance in the readings rose significantly (up to ±6 LSB). This suggests that the physical arrangement of the light sources relative to the light sensors is such that the amount of light reflected by the material onto the sensors is sensitive to the shape of the sample.

There are several different approaches to tackling this problem. Some preliminary experiments were performed, both varying the nature of the light source as well as making the inside surface of the sensor barrel more reflective. One particular material sample was sampled in two different ways. First, a reading was taken with the material flat on the table. A second sample was taken with the material pressed into the opening of the sensor in such a way that the light would reflect off of a convex surface. We took such sample pairs for 4 different configurations:

1. No alterations to the sensor
2. Putting a diffuser over the LED's
3. Putting a reflective surface inside the sensor
4. Using both a reflective surface and a diffuser.

The idea behind the diffuser is that the sensitivity of the readings to the specific shape of the material was due to the fact that the surface is not illuminated evenly. By diffusing the emitted light, a more even illumination of the material is achieved. The theory of the reflective inside surface was that if bright rays of light reflected off the surface of the sample, they would not be absorbed by the walls of the barrel but instead would bounce back and forth and eventually strike the sensor or the material again. The results suggest that the diffuser is an effective solution to the problem. In our preliminary experiments we were able to cut down the variations from 6 LSB, in Configuration 1, to ±3 LSB in Configurations 2 and 4. The diffuser used was not of very high quality; when examining the pattern that the light source made on the material, there were still very distinct bright and dark regions. A more diffuse light source might fix the problem.

It was difficult to tell whether or not the reflective surface on the inside of the barrel helped with the sensitivity to material texture. If it had an effect on the variation between the two measurements, the effect was negligible. However, making the inside of the sensor reflective had another interesting effect in that it generally increased the value of all readings. This was deemed useful because the photodiodes operate with less noise in higher light levels.

5.3.4. Signal Noise

Testing Inky's initial configuration quickly showed that considerable variations existed between individual readings of the same color state (color, texture, and alignment). The captured A/D values indicated a variation in some cases of more than 1V. Examining the

amplified input into the microcontroller on an oscilloscope showed two effects. The first was extensive noise in multiples of 60Hz. The noise was on the order of 0.6V and at low light levels (dark colors) easily masked the actual data signal. The source of this noise was related to the position of the electronics. By moving the sensor, protoboard, and interface board away from computers, monitors, and power supplies the 60Hz coupling was reduced to negligible amounts (< 20mV).

The second effect observed was an unusual coupling to our feet on the lab floor. By lifting one's foot off the floor, a sinusoidal and decaying disturbance could be seen on the output lines. The initial impulse depended on how fast one's foot was removed from the floor and could exceed 1.5V. By moving the electronics and being very careful not to move, much more reliable data was taken with the sensor.

These two effects are unacceptable in a final version. At this point, it is believed that both effects are related to the long wire leading from the sensors to the amplification electronics. The long wires containing low voltage and current levels are very susceptible to external noise events. Changing the connection cable to a shielded variety seemed to minimize these issues and the final design will place the amplification stage much closer to the sensors to further reduce the potential of coupled noise.

5.3.5. Top Compression Logarithmic Amplifier

It was encouraging to find that the blue and black data clusters did not overlap, however, only one LSB difference between a dark blue and a black does is not sufficient to repeatedly distinguish the two. For this reason, it is believed that some type of data manipulation prior to A/D conversion that would expand the tightly packed dark regions would yield higher color recognition accuracy.

In an attempt to reduce the possibility of region ambiguity, a logarithmic term was added to the amplification stage. The goal was to increase the separation between low intensity values at the expense of some resolution with brighter colors. Neither this design nor the previous design demonstrated difficulty identifying bright colors, so this trade-off seems reasonable. Since the log function has large slope for low input values and shallow slope for high input values, differences for dark colors would be stretched out much more than those for light colors. The task was to amplify the current produced by the photodiodes and to convert it into a voltage range about equal to 0-5V employing a logarithmic dependence.

To make an amplifier with a logarithmic dependence, an element with a logarithmic or exponential V-I characteristic is needed. A diode is such an element. A 1N4148 diode was used for the amplifier because it has a very logarithmic V-I characteristic in the 1uA-10uA range. The current produced by the photodiode is in the 100nA range so the 1N4148 is sufficient. Unfortunately, it is non-trivial to build a non-inverting (a requirement for single supply designs) logarithmic amplifier using op-amps. Eventually, a 2-stage design with logarithmic behavior was developed, and is shown in Figure 13

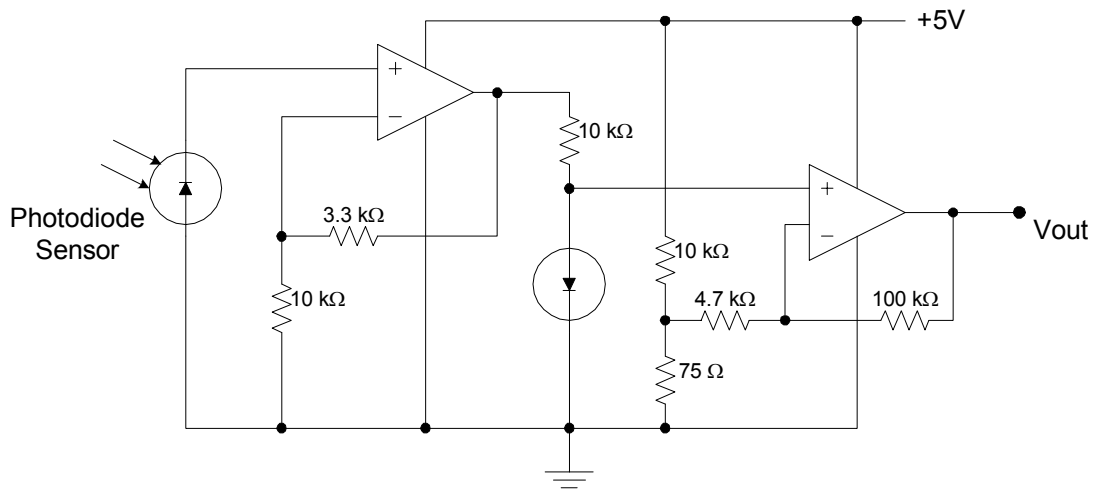

Figure 13: Two Stage "Top-Compressing" Op Amp Schematic

The circuit analysis of this effect is as follows. On the first op amp:

$$V_1 = \left(1 + \frac{3.3k\Omega}{10k\Omega}\right) \cdot V_{phdiode} = 1.33 \cdot V_{phdiode}$$

The governing equation for the output from the Resistor-Diode pairing is:

$$V_1 - V_{diode} = i \cdot R = 10k\Omega \cdot \left[I_0 \cdot \left(e^{\frac{V_{diode}}{\eta \cdot V_T}} - 1 \right) \right]$$

Where I_0, η, and V_T are constants of the diode.
Manipulating:

$$V_1 = 10k\Omega \cdot \left[I_0 \cdot \left(e^{\frac{V_{diode}}{\eta \cdot V_T}} - 1 \right) \right] + V_{diode}$$

Graphical or computer solutions show that V_{diode} increases quickly for low input values of V_1, but increases at a slower rate for higher values of V_1. Thus, the semi-logarithmic behavior was termed "top-compression."

The resulting non-linear response curve allowed a high gain to be applied with the second op amp stage. A bias voltage was used to help center V_{out} in the 0-5V range useable by the A/D converter on the HC908 microcontroller.

$$\frac{V_{out} - V_2}{100k\Omega} = \frac{V_2 - V_B}{4.7k\Omega}$$

$$\frac{5 - V_B}{10k\Omega} = \frac{V_B - 0}{75\Omega} \quad \rightarrow \quad V_B = \frac{5 \cdot 75\Omega}{10k\Omega + 75\Omega} = 37.2mV$$

$$V_{out} = \frac{1}{4.7k\Omega} \cdot \left[V_2 \cdot (100k\Omega + 4.7k\Omega) - V_B \cdot 100k\Omega \right] = 22.28 \cdot V_2 - 0.792V$$

While this design did allow the lower intensity data samples to be spread out more than in a pure linear amplifier design, the circuit parameters were being set and checked experimentally. Without a clear idea of exactly what the range of inputs from the photodiodes were and what sort of output was needed to improve the decoding by the D/A converter, fine-tuning the circuit was abandoned as a non-ideal use of research time.

5.3.6. Dmitriy's Amplifier

A second logarithmic amplifier was developed. A standard method of making the logarithmic amplifier involves an op-amp with a diode feedback. After many attempts to do this, the circuit still did not work. The voltage difference between the two inputs of the op-amp just did not go to zero, which made the output always go to one of the rails. Luckily, another way of connecting the circuit was found. The photodiode can be connected directly to the diode, which will cause the voltage between the two to be in the desired voltage range. Unfortunately the currents involved are very small so a buffer of some sort is needed if we connect anything to the diode-photodiode loop. We need to linearly amplify the voltage in order to scale it up to the 0-5V levels. We need a non-inverting amplifier. The input to this amplifier connects directly to the op-amp without the use of a resistor so a buffer is not needed. The configuration of this amplifier is shown in Figure 14.

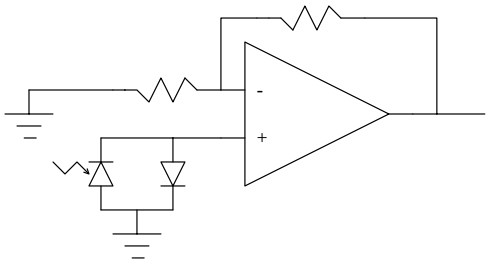

Figure 14: Configuration of Dmitriy's Amplifier

5.4. COLOR RECOGNITION IMPROVEMENT IDEAS

After experimenting with both the 7 LED prototype and the 3 RGB photodiode prototype, the team has assessed the advantages and disadvantages of each. With Blinky, the 7 LED prototype, the 7-coordinate definition of a color reduced the chance that two different samples had the same values. Unfortunately, there was overlap between some of the dark regions, leading to inaccuracy of the device. The three photodiodes were designed for color identification applications. No overlap was noticed in the dark regions. The current output of the photodiodes is extremely low, making the device susceptible to external noise. With the design of a custom PC board that allowed placement of the amplifiers inside the housing and some data manipulation, it is believed that this three photodiode prototype would be successful at distinguishing dark colors. The team has developed several ideas that may improve the accuracy of future prototypes.

5.4.1. Data Interpretation

An alternative method of data interpretation is an artificial neural network. The type of network that would be most appropriate to this device is a back-propagation network. As the network is presented with examples of correct input/output (known colors), it adjusts itself to be able to associate that input with its corresponding output. After enough samples are presented, the network should be able to reasonably extrapolate proper outputs to input that it has not seen before. So by presenting the network with a set of known colors, the network could be trained to associate the raw data coming from the A/D converter with the actual color code, which can be used in the speaking of the color. The algorithm to train a back-propagation neural network is too complicated to be quickly coded on an HC908. Fortunately, the training can be done on a computer and the trained net can be transferred to the micro-controller. Running a trained net is a multiply-accumulate process. Since the HC908 can perform both multiplication and addition, it can evaluate a trained neural net without much trouble.

5.4.2. Tuning the logarithmic amplifier through simulation

As described above, a logarithmic amplifier can be constructed and used to scale the color sensor output to better resolve dark colors. To take full advantage of the logarithmic amplification, the input to the amplifier has to be in the active region of the logarithm. Resistors can be used to linearly adjust the input voltages so that they lie in the active region. It may not be simple to figure out what values of the resistors are needed and even after that, in order to see whether the logarithmic amplifier works, the system will have to be assembled. Alternatively, if linearly amplified data is already available, the logarithm can be numerically simulated to obtain the output values that would be generated if a logarithmic amplifier was used. Given the type of the diode and the resistor values, the logarithmic amplifier can be thought of as a black box with a certain function relating the output to the input. Since this function is known, numerical processing can be done.

5.4.3. Packaging revisions

At this point the main purpose of a package revision would be to make the device less sensitive to the configuration of the material it is measuring. Currently measurements are only reliably repeatable when the sample is in the same configuration for all of them (flat on the work bench for calibration data). We would like to reduce the dependence of the readings on the particular shape of the material as much as possible. There are two fundamentally different approaches to this problem. First, one could attempt to change the physical layout of the sensor so that the device is less sensitive to the configuration of the sample. Second, one could make sure that the material always has the same configuration relative to the sensor.

One way of fixing the position of the sample would be to mount a rigid transparent surface across the opening of the sensor package. Pushing the device firmly onto a sample patch would then press the material completely flat. However, one difficulty one might run into with this approach is that the transparent surface could also be reflective and interfere with the measurements. Assuming that a constant amount is reflected, it is possible to simply have an offset for the amplifier and account for this internal reflection. On the other hand the device has the most trouble with resolving dark color values, i.e. little reflected light. Noise in the reflection will interfere with these measurements and make resolving dark colors even more difficult.

The layout revision for the sensor package should focus on a more uniform illumination of the sample patch. We think that the sharp boundaries of the light cone in the current version are responsible for a large portion of the sensitivity to the shape of the material. When a photodiode is fully illuminated in one configuration of the sample, even a slight change in the position of the surface could cause a large change in the amount of light reaching the sensor. Therefore one of the objectives of the revision should be to make a much more diffuse light source. Solutions that come to mind are to either add a diffuser, or to have many more discrete light sources.

5.4.4. Usage of 2 LUT's and 2 Light Intensities

Industrial colorimeters operate on the same principals that we are trying to exploit, namely the reflectivity or absorption by a given material in a given spectrum. However, many colorimeters are more application specific (i.e.- quality control in manufacturing processes, where the expected readings are very specific). In our application dynamic range is a problem. On a linear scale the dark colors are grouped too close together to reliably differentiate between them. There are a number of different approaches to solve this problem:

Resampling with a different resolution: When the microcontroller samples a given photodiode and the result of the A/D conversion falls blow a critical value, the microcontroller resamples the photodiode with a greater resolution as a trade off for a lower range. In theory this is easily implemented by lowering the upper reference voltage to the A/D converter before resampling. However, in our current microcontroller, the upper reference voltage is internally tied to the Vcc making lowering the reference voltage impossible without switching microcontrollers.

Resampling with a higher illumination: This approach relies on the assumption that the amount of light reflected is proportional to the incident light. When the result of an A/D conversion falls below a critical value the microcontroller turns on additional light sources in the sensor package before resampling. In theory this should amplify the differences proportional to the amount of additional light sources. One problem with this method is that the current sensor package is already tightly packed with LED's and photodiodes. Mounting additional LEDs of other light sources would be difficult.

A. BLINKY'S ASSEMBLY CODE

```
RAMStart     EQU  $0040
RomStart     EQU  $E000
DataStart    EQU  $E300
VectorStart  EQU  $FFDC

$Include 'kxregs.inc'

   org RamStart

ADC_0   ds 1
ADC_1   ds 1
ADC_2   ds 1
ADC_3   ds 1
ADC_4   ds 1
ADC_5   ds 1
ADC_6   ds 1

DC            ds 2
TOT_DATA      ds 1
TCTR          ds 1
DIST          ds 3
MIN_DIST      ds 3
T0            ds 2
T1            ds 1
TADD          ds 1
TMAX          ds 1
SND0          ds 1
SND1          ds 1

   org RomStart

Main_Init:
      rsp
      clra
      clrx
;      cli            ; Enable interrupts
      mov #$3,$1F      ; Disable COP & Enable STOP

      ; Init Port A
      mov #$01,$00     ; Set port A up for PBIC (CE = HIGH, PD = LOW, ADX[1-n]=LOW
      mov #$1F,$04     ; Set all of port A to output
      mov #$01,$00     ; Set port A up for PBIC (CE = HIGH, PD = LOW, ADX[1-n]=LOW

      ; Init Port B
```

```
      mov #$F7,$05     ; Set all of port B (except ADC ch 3) to output
      mov #$00,$01     ; Clear port B

      bset 7,PTB       ; Keep Power On

      ; Configure counter
      mov #$FF,$23     ; Set the timer to trigger ; MSB
      mov #$FF,$24     ; overflow at FFFF count   ; LSB
      mov #$36,TSC     ; Timer - Cleared + Stopped.


Main_Proc:
      jsr iterate
      bra end          ; End the program


; Program end - Freeze or Turn off hardware
end:
;      stop
      bclr 7,PTB       ; Power Down Module
      bra end


; Wait function.. waits for preset amount
wait:
     mov #$30,$20
     mov #$00,$20
     jsr wait_t
     jsr wait_t
     jsr wait_t
     jsr wait_t
     jsr wait_t
     jsr wait_t
     jsr wait_t
wait_t:
     mov #$30,$20
     mov #$00,$20
wait_tt
     brclr 7,$20,wait_tt
     rts

*************************************************
* iterate - Perform all tasks necessary to obtain
* a reading and 'say' the color
*************************************************
iterate:
```

```
;loop:
;     brset 2,PTA,loop

        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete

*** Analog Control ***
        mov #$81,PTB    ; Turn on LED 1
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_0   ; Store obtained value
        mov #$80,PTB    ; Turn off Red LED

        mov #$82,PTB    ; Turn on LED 2
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_1   ; Store obtained value
        mov #$80,PTB    ; Turn off Green LED

        mov #$83,PTB    ; Turn on LED 3
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_2   ; Store obtained value
        mov #$80,PTB    ; Turn off Blue LED

        ; PTB 3 is used for ADC

        mov #$84,PTB    ; Turn on LED 4
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_3   ; Store obtained value
        mov #$80,PTB    ; Turn off Blue LED

        mov #$85,PTB    ; Turn on LED 5
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_4   ; Store obtained value
        mov #$80,PTB    ; Turn off Blue LED

        mov #$86,PTB    ; Turn on LED 6
```

```
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_5   ; Store obtained value
        mov #$80,PTB    ; Turn off Blue LED

        mov #$87,PTB      ; Turn on LED 7
        jsr wait
        mov #$03,$3C    ; Set ADC Ch 3 and start conversion
        jsr adc_wait    ; Wait until ADC conversion complete
        mov $3D,ADC_6   ; Store obtained value
        mov #$80,PTB      ; Turn off Blue LED

**********************

*** Color ID Algorithm ***
; DC = Data Counter (Pointer to data byte) (2-byte)
; TOT_DATA = Number of data samples (1-byte)
; TCTR = Temporary counter (0 to MAX_DATA) (1-byte)
; DIST = Current Distance being tested (3-byte)
; MIN_DIST = Maximum Distance found so far (3-byte)
; T0,T1 = 2-byte temporary storage
        ; Obtain TOT_DATA header information
        ldhx #$E300       ; Get the total number of data chunks
        lda ,x
        sta TOT_DATA

        mov #$FF,SND0
        mov #$FF,SND1

        mov #$FF,MIN_DIST       ; set min dist at max
        mov #$FF,{MIN_DIST+1}
        mov #$FF,{MIN_DIST+2}

        mov #$00,T1
        mov #$00,{T1+1}

        ; loop until TOT_DATA has been reached
        ; for (TCTR=0; TCTR<TOT_DATA; TCTR++)
        mov #$00,TCTR  ; Set TCTR to 0
        ldhx #$E301    ; Set DC to E301
        sthx DC
        bra test
mn_loop:
      mov #$00,DIST       ; Clear Dist
      mov #$00,{DIST+1}
```

```
        mov #$00,{DIST+2}

        ; Init TADD (Temporary variable to store which add
        ; we're performing) and TMAX - the max # of adds
        mov #ADC_0,TADD
        lda TADD
        add #$07
        sta TMAX

add_more:               ; Not finished with this addition
        ; Obtain next ADC value to examine
        clrh
        ldx TADD
        lda ,x
        sta T1

        ; DIST += (tl[i]-l[i])*(tl[i]-l[i])

        ; Obtain next data value to examine
        ldhx DC
        lda ,x

        ; find abs(ADC[i] - l[i])
        cmp T1          ; See if ADC_0 > tl[i]
        bhi t_a1
        sta T0          ; ADC_0 - tl[i]
        lda T1
        sub T0
        bra t_a2
t_a1:
        sub T1          ; tl[i] - ADC_0
t_a2:

        ; (ADC_0 - l[i])^2
        sta T0          ; Square result
        ldx T0
        mul
        stx T0          ; Result in T0
        sta {T0+1}

        ; Add to DIST
        jsr Add_To_Dist

        ; Update DC
        jsr Inc_DC
```

```
        ; Has all addition been performed?
        inc TADD
        lda TMAX
        cmp TADD
        bhi add_more

        ; DIST < MIN_DIST ? MIN_DIST=DIST : <NOTHING>
        lda {DIST+0}
        cmp {MIN_DIST+0}
        bhi DNLTMD
        blo DNGRMD
        lda {DIST+1}
        cmp {MIN_DIST+1}
        bhi DNLTMD
        blo DNGRMD
        lda {DIST+2}
        cmp {MIN_DIST+2}
        bhi DNLTMD
        bra DNGRMD

test:               ; See if finished with all Data
        ; TCTR<TOT_DATA ? goto loop : stop
        lda TCTR
        inc TCTR
        cmp TOT_DATA
        blt mn_loop
        bra Play_Sound

DNGRMD:         ; Set MIN_DIST=DIST because DIST <= MIN_DIST
        mov {DIST+2},{MIN_DIST+2}
        mov {DIST+1},{MIN_DIST+1}
        mov {DIST+0},{MIN_DIST+0}
        ; Place sound addresses in SND0 & SND1
        ldhx DC
        lda ,x
        sta SND0
        jsr Inc_DC
        ldhx DC
        lda ,x
        sta SND1
        jsr Inc_DC
        bra test
DNLTMD:             ; Failure, do not set MIN_DIST=DIST
        jsr Inc_DC      ; Increment past 2 sound bytes
        jsr Inc_DC
        bra test
```

```
; Play chosen clip
Play_Sound:
; Set address, PD low, CE high
      lda SND0
      and #$1C
      ora #$01
      sta PTA
;      and #$FE
      lda SND0
      lsla
      lsla
      lsla
      lsla
      and #$B0
      ora #$80
      sta PTB
; PD LOW, CE LOW - start playback
      bclr 0,PTA
; PD Low, CE high
      bset 0,PTA

; Wait until INT goes low (done playing)
snd_loop0:
      bih snd_loop0
; Wait until INT goes high again
snd_loop0a:
      bil snd_loop0a


; Play Second sound (if applicable (play if SND1 != FF))
; Set address, PD low, CE high
      lda SND1
      cmp #$FF  ; applicable?
      beq CLRIDEND
      and #$1C
      ora #$01
      sta PTA
      lda SND1
      lsla
      lsla
      lsla
      lsla
      and #$B0
      ora #$80
      sta PTB
```

```
; PD LOW, CE LOW - start playback
    bclr 0,PTA
; PD Low, CE high
    bset 0,PTA
snd_loop1:
    bih snd_loop1
snd_loop1a:
    bil snd_loop1a


; End of Color ID and playback algorithms
CLRIDEND:
    rts



*** SUB-ROUTINES *

Inc_DC:
    lda {DC+1}
    add #$01        ; Not doing inc because need carry
    sta {DC+1}
    clra
    adc {DC+0}
    sta {DC+0}
    rts

Add_To_Dist:
    clc
    lda {T0+1}      ; Add value to DIST+2
    adc {DIST+2}
    sta {DIST+2}
    lda {T0+0}      ; Add value to DIST+1
    adc {DIST+1}
    sta {DIST+1}
    clra
    adc {DIST+0}
    sta {DIST+0}
    rts

*************************


adc_wait:
      brclr 7,$3C,adc_wait   ; Loop until ADC COCO bit is set
      rts
```

```
dummy_isr:
      rti         ; return

**** COLOR DATA SECTION ****
* Format of color data information:
* Header:
* Byte 1 - Number of data samples (9-byte data chunks)
*
* Data (repeat):
* Byte 0 to 6 stores IC 8-bit color readings for respective LEDs
* BYTE 7,8 correspond to color byte information (as is sent to the
*         address lines of the pcic)
  org  DataStart

  ; Header (Number of colors)
  DB $48

  ; Data (Color - 7 bytes, Playback Snd 0 - 1 byte, snd1 - 1 byte)
  DB $0B,$03,$09,$03,$0D,$02,$09,$10,$02
  DB $5,$1,$4,$1,$3,$1,$3,$11,$C
  DB $18,$6,$11,$3,$4,$1,$A,$11,$B
  DB $F,$3,$8,$2,$4,$1,$8,$5,$FF
  DB $11,$4,$F,$4,$21,$2,$10,$12,$1
  DB $18,$3,$8,$2,$4,$1,$9,$6,$FF
  DB $1D,$8,$1B,$6,$14,$1,$11,$3,$FF
  DB $1F,$8,$1A,$4,$7,$1,$F,$B,$FF
  DB $9,$2,$4,$2,$3,$1,$4,$11,$6
  DB $9,$1,$4,$1,$4,$1,$4,$11,$1
  DB $C,$3,$9,$2,$5,$1,$7,$D,$FF
  DB $13,$5,$12,$4,$19,$3,$12,$10,$A
  DB $D,$2,$6,$2,$6,$1,$6,$11,$A
  DB $A,$3,$9,$3,$C,$2,$9,$13,$1
  DB $26,$C,$27,$8,$21,$2,$1C,$3,$FF
  DB $24,$8,$13,$3,$4,$1,$D,$0,$FF
  DB $27,$B,$21,$6,$8,$1,$10,$12,$B
  DB $10,$3,$B,$3,$11,$3,$D,$10,$2
  DB $13,$5,$10,$4,$25,$4,$18,$10,$F
  DB $F,$3,$A,$3,$20,$4,$13,$11,$F
  DB $A,$2,$5,$2,$5,$1,$5,$11,$A
  DB $14,$4,$E,$3,$15,$3,$10,$10,$2
  DB $D,$3,$8,$2,$7,$2,$9,$6,$FF
  DB $20,$8,$16,$4,$5,$1,$C,$12,$0
  DB $F,$3,$7,$2,$3,$1,$6,$5,$FF
  DB $24,$B,$22,$7,$17,$1,$17,$3,$FF
  DB $5,$1,$4,$1,$3,$1,$4,$11,$2
```

```
DB $12,$4,$10,$4,$17,$3,$F,$10,$2
DB $6,$1,$3,$1,$3,$1,$3,$8,$FF
DB $23,$A,$24,$8,$2F,$4,$1E,$9,$FF
DB $21,$E,$24,$A,$2E,$8,$25,$9,$FF
DB $24,$B,$25,$7,$30,$5,$1F,$9,$FF
DB $9,$2,$5,$2,$A,$1,$5,$1,$FF
DB $1F,$C,$1D,$6,$17,$1,$13,$12,$B
DB $8,$1,$4,$1,$3,$1,$3,$8,$FF
DB $C,$3,$9,$3,$6,$1,$6,$C,$FF
DB $D,$3,$9,$2,$A,$2,$9,$10,$6
DB $A,$2,$5,$2,$7,$1,$5,$1,$FF
DB $9,$2,$5,$2,$D,$3,$9,$2,$FF
DB $1F,$7,$11,$4,$12,$3,$13,$0,$FF
DB $15,$5,$12,$4,$18,$2,$F,$10,$1
DB $B,$2,$6,$2,$5,$1,$6,$6,$FF
DB $A,$1,$3,$2,$3,$1,$4,$8,$FF
DB $14,$5,$10,$4,$1C,$3,$10,$10,$2
DB $10,$3,$9,$2,$3,$1,$6,$11,$0
DB $27,$B,$25,$8,$35,$5,$23,$9,$FF
DB $21,$9,$1F,$7,$26,$3,$1B,$13,$9
DB $20,$7,$12,$3,$4,$1,$B,$0,$FF
DB $9,$2,$5,$2,$4,$1,$7,$6,$FF
DB $10,$4,$D,$3,$13,$2,$D,$10,$2
DB $B,$1,$4,$2,$3,$1,$4,$11,$2
DB $A,$1,$4,$2,$4,$1,$4,$11,$2
DB $D,$2,$5,$2,$4,$1,$6,$C,$FF
DB $B,$2,$5,$2,$5,$1,$5,$A,$FF
DB $C,$2,$4,$2,$4,$1,$6,$6,$FF
DB $12,$4,$C,$3,$B,$1,$9,$11,$D
DB $21,$9,$1F,$7,$1E,$1,$15,$3,$FF
DB $A,$2,$4,$2,$3,$1,$5,$11,$2
DB $24,$A,$1F,$5,$6,$1,$10,$B,$FF
DB $D,$2,$6,$2,$A,$2,$9,$2,$FF
DB $1F,$A,$20,$7,$2A,$4,$1E,$7,$FF
DB $23,$A,$22,$7,$22,$3,$1B,$10,$D
DB $10,$4,$D,$3,$12,$3,$F,$F,$FF
DB $17,$6,$13,$4,$11,$1,$E,$D,$FF
DB $B,$2,$6,$2,$7,$1,$6,$11,$1
DB $23,$A,$22,$7,$29,$4,$1E,$13,$9
DB $2A,$D,$2A,$8,$16,$1,$1A,$3,$B
DB $D,$3,$7,$2,$6,$1,$7,$5,$FF
DB $15,$5,$10,$4,$16,$2,$10,$10,$A
DB $15,$6,$12,$4,$7,$1,$C,$C,$FF
DB $26,$C,$28,$8,$38,$6,$27,$9,$FF
DB $A,$2,$7,$2,$6,$1,$6,$C,$FF
DB $E,$3,$D,$3,$F,$2,$E,$A,$FF
```

```
DB $19,$6,$E,$2,$5,$1,$B,$0,$FF


org  VectorStart

    dw  dummy_isr    ; Time Base Vector
    dw  dummy_isr    ; ADC Conversion Complete
    dw  dummy_isr    ; Keyboard Vector
    dw  dummy_isr    ; SCI Transmit Vector
    dw  dummy_isr    ; SCI Receive Vector
    dw  dummy_isr    ; SCI Error Vector
    dw  dummy_isr    ; Reserved
    dw  dummy_isr    ; Reserved
    dw  dummy_isr    ; Reserved
    dw  dummy_isr    ; Reserved
    dw  dummy_isr    ; Reserved
    dw  dummy_isr    ; TIM Overflow Vector
    dw  dummy_isr    ; TIM Channel 1 Vector
    dw  dummy_isr       ; TIM Channel 0 Vector
    dw  dummy_isr    ; CMIREQ Vector
    dw  dummy_isr    ; ~IRQ1 Vector
    dw  dummy_isr    ; SWI Vector
    dw  Main_Init    ; Reset Vector
```

B. INKY'S ASSEMBLY CODE

```
*********************************
*  PINOUT
* 1 - Gnd
* 2 - A[1]     ???
* 3 - A[0]     ???
* 4 - IRQ1_Bar ???
* 5 - ADC_0    r
* 6 - ADC_1    g
* 7 - ADC_3    b
* 8 - B[3]     LED output
* 9 - B[7]     N/C
*10 - B[6]     N/C
*11 - B[5]     N/C
*12 - B[4]     N/C
*13 - A[2]     ???
*14 - A[3]     ???
*15 - A[4]     ???
*16 - Vdd/Vrefh
*********************************



RAMStart        EQU  $0040 ; Starting location of variables
RomStart        EQU  $E000 ; Starting location of program code
DataStart       EQU  $E300 ; Starting location of look up table data
VectorStart     EQU  $FFDC ; Location of ISR routine jumps


$Include 'kxregs.inc' ; File containing register aliases



      org RamStart

ADC_0  ds 1 ; A/D
ADC_1  ds 1 ; Conversion
ADC_2  ds 1 ; Results


DC              ds 2 ; Data Counter (Pointer to data byte) (2-byte)

TOT_DATA        db 1 ; Number of data samples (1-byte)
TCTR            ds 1 ; Temporary counter (0 to MAX_DATA) (1-byte)
DIST            ds 3 ; Current Distance being tested (3-byte)
MIN_DIST        ds 3 ; Maximum Distance found so far (3-byte)
T0              ds 2 ; 2-byte temporary storage
T1              ds 1 ; 2-byte temporary storage
TADD            ds 1 ; Temporary variable to store which add we're performing
TMAX            ds 1 ; Maximum # of additions to take place
SND0            ds 1 ; Address of 1st sound to play back
```

```
SND1            ds 1 ; Address of 2nd sound to play back



        org RomStart

Main_Init:
        rsp
        clra
        clrx
;       cli             ; Enable interrupts, NOT done, but this is where it would be done

        mov #$3,$1F     ; Disable COP & Enable STOP

        ; Init Port A
        mov #$01,$00    ; Set port A up for PBIC (CE = HIGH, PD = LOW, ADX[1-n]=LOW
        mov #$1F,$04    ; Set all of port A to output
        mov #$01,$00    ; Set port A up for PBIC (CE = HIGH, PD = LOW, ADX[1-n]=LOW



        ; Init Port B
        mov #$00,PTB    ; Write to Port B as safety procedure
        mov #$F8,DDRB   ; Configure Port B's pins as I or O:  B[7:3] to output, B[2:0] = ADC[2:0]
        mov #$00,PTB  ; Clear Port B


        bset 7,PTB    ; Keep Power On

        ; Configure counter
        mov #$FF,$23    ; Set the timer to trigger;  MSB
        mov #$FF,$24    ; overflow at FFFF count   ; LSB
        mov #$36,TSC    ; Timer - Cleared + Stopped.




Main_Proc:
   jsr iterate ; go to the main portion of the program
   bra end        ; End the program



; Program end - Freeze or Turn off hardware

end:

; stop
   bclr 7,PTB       ; Power Down Module
   bra end
```

```
; Wait function.. waits for preset amount doing useless operations

wait:
   mov #$30,$20
   mov #$00,$20
   jsr wait_t
   jsr wait_t
   jsr wait_t
   jsr wait_t
   jsr wait_t
   jsr wait_t
   jsr wait_t

wait_t:
   mov #$30,$20
   mov #$00,$20

wait_tt:
   brclr 7,$20,wait_tt
   rts



**************************************************
*iterate - Perform all tasks necessary to obtain
* a reading and 'say' the color
**************************************************
iterate:

;loop:
;   brset 2,PTA,loop

; Do a warm up conversion with the A/D converters:
   mov #$00,ADSCR
   jsr adc_wait
   mov #$01,ADSCR
   jsr adc_wait
   mov #$02,ADSCR
   jsr adc_wait


*** Analog Control ***


   mov #$08,PTB      ; Turn on both LEDs
   jsr wait          ; Wait for the hell of it

   mov #$00,ADSCR    ; Set ADC Ch 0 and start conversion
   jsr adc_wait      ; Wait until ADC conversion complete
   mov $3D,ADC_0     ; Store obtained value

   mov #$01,ADSCR    ; Set ADC Ch 1 and start conversion
   jsr adc_wait      ; Wait until ADC conversion complete
   mov $3D,ADC_1     ; Store obtained value
```

```
   mov #$02,ADSCR    ; Set ADC Ch 2 and start conversion
   jsr adc_wait      ; Wait until ADC conversion complete
   mov $3D,ADC_2     ; Store obtained value

   mov #$00,PTB      ; Turn off both LEDs



**************************
**** Color ID Algorithm ****
**************************

; DC = Data Counter (Pointer to data byte) (2-byte)

; TOT_DATA = Number of data samples (1-byte)

; TCTR = Temporary counter (0 to MAX_DATA) (1-byte)

; DIST = Current Distance being tested (3-byte)

; MIN_DIST = Maximum Distance found so far (3-byte)

; T0, T1 = 2-byte temporary storage



   ; Obtain TOT_DATA header information

   ldhx #DataStart    ; Get the total number of data chunks
   lda ,x ; from the first
   sta TOT_DATA ; entry in the data table

   mov #$FF,SND0 ; Initialize the sound playback
   mov #$FF,SND1 ; registers with empty sound

   mov #$FF,MIN_DIST    ; set min dist at maximum possible
   mov #$FF,{MIN_DIST+1}
   mov #$FF,{MIN_DIST+2}

   mov #$00,T1 ; Clear the temporary
   mov #$00,{T1+1} ; variables


   ; loop until TOT_DATA has been reached
   ; for (TCTR=0; TCTR<TOT_DATA; TCTR++)

   mov #$00,TCTR    ; Set TCTR to 0
   ldhx #$E301      ; Set DC to E301
   sthx DC
   bra test

mn_loop:
   mov #$00,DIST    ; Clear Dist
```

```
    mov #$00,{DIST+1}
    mov #$00,{DIST+2}

    ; Init TADD (Temporary variable to store which add
    ; we're performing) and TMAX - the max # of adds

    mov #ADC_0,TADD
    lda TADD
    add #$03 ; number of data samples per data point (i.e.- 3 for RGB)

    sta TMAX

add_more:
    ; Not finished with this addition
    ; Obtain next ADC value to examine

     clrh
     ldx TADD
     lda ,x
     sta T1

     ; DIST += (tl[i]-l[i])*(tl[i]-l[i])


     ; Obtain next data value to examine:

     ldhx DC
     lda ,x


     ; find abs(ADC[i] - l[i])

     cmp T1    ; See if ADC_0 > tl[i]
     bhi t_a1
     sta T0    ; ADC_0 - tl[i]
     lda T1
     sub T0
     bra t_a2

t_a1:
    sub T1    ; tl[i] - ADC_0

t_a2:
    ; (ADC_0 - l[i])^2

    sta T0    ; Square result
    ldx T0
    mul
    stx T0    ; Store to T0
    sta {T0+1}


    ; Add to DIST

    jsr Add_To_Dist ; add squared value to total distance
```

```
    ; Update DC

    jsr Inc_DC


    ; Has all addition been performed?

    inc TADD
    lda TMAX
    cmp TADD
    bhi add_more


    ; DIST < MIN_DIST ? MIN_DIST=DIST : <NOTHING>

    lda {DIST+0} ; Check each distance calculation
    cmp {MIN_DIST+0} ; Compared to the current minimum
    bhi DNLTMD ; if greater, do not update
    blo DNGRMD ; if smaller, update the current minimum

    lda {DIST+1}
    cmp {MIN_DIST+1}
    bhi DNLTMD
    blo DNGRMD

    lda {DIST+2}
    cmp {MIN_DIST+2}
    bhi DNLTMD
    bra DNGRMD


test:    ; See if finished with all Data

    ; TCTR<TOT_DATA ? goto loop : proceed to sound playback

    lda TCTR
    inc TCTR
    cmp TOT_DATA
    blt mn_loop
    bra Play_Sound


DNGRMD:

    ; Set MIN_DIST=DIST because DIST <= MIN_DIST

    mov {DIST+2},{MIN_DIST+2}
    mov {DIST+1},{MIN_DIST+1}
    mov {DIST+0},{MIN_DIST+0}


    ; Place sound addresses in SND0 & SND1
```

```
    ldhx DC
    lda ,x
    sta SND0
    jsr Inc_DC
    ldhx DC
    lda ,x
    sta SND1
    jsr Inc_DC
    bra test


DNLTMD:    ; Failure, do not set MIN_DIST=DIST

    jsr Inc_DC
    jsr Inc_DC
    bra test


; Play chosen clip

Play_Sound:

; Set address, PD low, CE high
; Interfaces with the sound playback chip observing its required timings

    lda SND0
    and #$1C
    ora #$01
    sta PTA
    and #$FE
    lsla
    lsla
    lsla
    lsla
    and #$B0
    sta PTB
    stop


    ; PD LOW, CE LOW - start playback

    bclr 0,PTA;
    jsr wait    ; Wait for one second


    ; PD Low, CE high - end playback (?)

    bset 0,PTA
    jsr wait    ; Wait for one second
    jsr wait    ; Wait for one second
    jsr wait    ; Wait for one second


    ; Play Second sound (if applicable (play if SND1 != FF))
    ; Set address, PD low, CE high
```

```
    lda SND1
    cmp #$FF    ; test if empty sound
    beq CLRIDEND ; if so, don't play anything
    and #$1C
    ora #$01
    sta PTA
    lsla
    lsla
    lsla
    lsla
    and #$B0
    sta PTB


    ; PD LOW, CE LOW - start playback

    bclr 0,PTA


    ; PD Low, CE high - end playback (?)

    bset 0,PTA
    jsr wait    ; Wait for one second

; End of Color ID and playback algorithms


CLRIDEND:

    rts



*** SUB-ROUTINES ***

; Moves Data Counter to next data block:
Inc_DC:
     lda {DC+1}
     add #$01    ; Not doing inc because need carry
     sta {DC+1}
     clra
     adc {DC+0}
     sta {DC+0}
     rts


; adds a value in T0 to Dist
Add_To_Dist:

    clc
    lda {T0+1}    ; Add value to DIST+2
    adc {DIST+2}
    sta {DIST+2}
    lda {T0+0}    ; Add value to DIST+1
```

```
   adc {DIST+1}
   sta {DIST+1}
   clra
   adc {DIST+0}
   sta {DIST+0}
   rts


**************************

; Wait routine for A/D conversion:
adc_wait:

   brclr 7,$3C,adc_wait    ; Loop until ADC COCO bit is set
   rts


; Not needed, from template (?)
dummy_isr:

   rti    ; return




**** COLOR DATA SECTION *****
* Format of color data information:
* Header:
* Byte 1 - Number of data samples (9-byte data chunks)
*
* Data (repeat):
* Byte 0 to 2 stores IC 3-bit color readings for respective LEDs
* BYTE 3,4 correspond to color byte information (as is sent to the
*         address lines of the pcic)

*******
* The colors have the following assignments
; Light = 10
; Dark = 11
; Pink = 12
; Burgandy = 13
; Yellow = 14
; Orange = 15
; Green = 16
; Blue = 17
; White = 18
; Red = 19
; Off = 20
; Beige = 21
; Tan = 22
; Brown = 23
; Turquoise = 24
; Black = 25
; Magenta = 26
```

```
; Khaki = 27
; Grey = 28



   org  DataStart

      DB $60


   DB $F4,$70,$78,$12,$FF
   DB $28,$0D,$0C,$13,$FF
   DB $D7,$55,$11,$14,$15
   DB $13,$16,$0D,$11,$16
   DB $D5,$84,$18,$14,$FF
   DB $16,$23,$12,$16,$FF
   DB $3D,$52,$84,$10,$17
   DB $1C,$23,$36,$17,$FF
   DB $E6,$E1,$D9,$18,$FF
   DB $87,$0D,$0C,$19,$FF
   DB $C1,$1D,$0E,$15,$FF
   DB $C5,$A9,$8D,$20,$18
   DB $D2,$96,$6B,$21,$FF
   DB $68,$43,$27,$22,$FF
   DB $2A,$15,$11,$23,$FF
   DB $12,$0E,$0C,$11,$23
   DB $23,$2C,$43,$24,$FF
   DB $16,$18,$1F,$17,$FF
   DB $4B,$46,$4B,$28,$FF
   DB $0F,$0D,$11,$11,$17
   DB $0F,$0D,$0D,$25,$FF
   DB $1C,$24,$47,$17,$FF
   DB $12,$0E,$11,$11,$17
   DB $1B,$12,$11,$23,$FF
   DB $15,$0F,$12,$11,$26
   DB $0F,$0D,$10,$11,$17
   DB $10,$0F,$10,$11,$17
   DB $10,$0E,$11,$11,$17
   DB $12,$0F,$13,$11,$17
   DB $0D,$0C,$0B,$25,$FF
   DB $0E,$0E,$0E,$11,$17
   DB $10,$0E,$0D,$25,$FF
   DB $0E,$0C,$0C,$25,$FF
   DB $0E,$0C,$0C,$25,$FF
   DB $11,$0D,$10,$11,$17
   DB $0F,$0D,$0D,$25,$FF
   DB $10,$0D,$0D,$25,$FF
   DB $0D,$0D,$0C,$25,$FF
   DB $11,$10,$0E,$11,$17
   DB $10,$0F,$13,$11,$17
   DB $13,$17,$16,$11,$16
   DB $13,$13,$14,$11,$25
   DB $12,$0F,$10,$11,$17
   DB $28,$20,$1C,$23,$FF
```

```
DB $29,$29,$2B,$28,$FF
DB $0F,$0D,$0D,$25,$FF
DB $16,$16,$1A,$17,$16
DB $0D,$0C,$0C,$25,$FF
DB $15,$14,$13,$11,$28
DB $0D,$0C,$0C,$25,$FF
DB $0D,$0C,$0C,$25,$FF
DB $12,$0F,$12,$11,$17
DB $13,$10,$14,$11,$17
DB $47,$4E,$5A,$10,$17
DB $1A,$19,$24,$17,$FF
DB $12,$10,$13,$11,$17
DB $13,$14,$1A,$17,$FF
DB $10,$0F,$14,$11,$17
DB $12,$10,$12,$17,$16
DB $0F,$0D,$0F,$11,$17
DB $0F,$0D,$0C,$25,$FF
DB $85,$8D,$80,$20,$18
DB $44,$A9,$61,$10,$17
DB $7D,$92,$A1,$10,$14
DB $2D,$29,$3D,$10,$17
DB $31,$10,$15,$26,$FF
DB $4D,$0D,$0C,$26,$FF
DB $35,$0E,$0D,$19,$FF
DB $13,$1B,$16,$19,$FF
DB $27,$22,$37,$16,$FF
DB $29,$0E,$0E,$26,$FF
DB $92,$92,$92,$19,$FF
DB $8C,$82,$6C,$10,$28
DB $7F,$70,$50,$20,$18
DB $88,$85,$57,$21,$FF
DB $55,$4D,$3A,$20,$18
DB $79,$70,$64,$27,$FF
DB $27,$2C,$35,$20,$18
DB $E5,$D9,$CC,$17,$FF
DB $5D,$52,$3B,$18,$FF
DB $63,$5C,$47,$27,$FF
DB $8D,$8B,$7B,$27,$FF
DB $9E,$90,$4F,$25,$FF
DB $59,$68,$60,$20,$28
DB $42,$3D,$2F,$14,$FF
DB $1B,$1C,$21,$25,$FF
DB $35,$41,$43,$27,$FF
DB $35,$40,$63,$17,$28
DB $49,$41,$37,$10,$17
DB $22,$29,$21,$10,$17
DB $25,$2B,$43,$10,$16
DB $47,$46,$57,$16,$FF
DB $DC,$D6,$D1,$17,$FF
DB $2C,$2C,$23,$28,$FF


; I think the following is from a code template and not actually used:
```

```
org  VectorStart

   dw  dummy_isr    ; Time Base Vector
   dw  dummy_isr    ; ADC Conversion Complete
   dw  dummy_isr    ; Keyboard Vector
   dw  dummy_isr    ; SCI Transmit Vector
   dw  dummy_isr    ; SCI Receive Vector
   dw  dummy_isr    ; SCI Error Vector
   dw  dummy_isr    ; Reserved
   dw  dummy_isr    ; Reserved
   dw  dummy_isr    ; Reserved
   dw  dummy_isr    ; Reserved
   dw  dummy_isr    ; Reserved
   dw  dummy_isr    ; TIM Overflow Vector
   dw  dummy_isr    ; TIM Channel 1 Vector
   dw  dummy_isr    ; TIM Channel 0 Vector
   dw  dummy_isr    ; CMIREQ Vector
   dw  dummy_isr    ; ~IRQ1 Vector
   dw  dummy_isr    ; SWI Vector
   dw  Main_Init    ; Reset Vector
```

C. COMPONENT DATA SHEETS

MC68HC908KX8 Microcontroller:

HTTP://E-WWW.MOTOROLA.COM/WEBAPP/SPS/SITE/PROD_SUMMARY.JSP?CODE=68HC908KX8&NODEID=03M0YM4T3ZGM98634

LEDs:
http://www.lumex.com/

ChipCorder:
http://www.winbond.com.tw/PDF/sheet/2560_1.pdf

Hamamatsu Photodiodes:
http://sfa.hamamatsu.com/device.nsf/0/5ddc51855d209892852566a4004bb403/$FILE/S6428-01,%20S6429-01,%20S6430-01,%20S7505%20e%20KSPD1013E02.pdf