Department of Electrical and Computer Engineering

EEL 3923C Engineering Design I

Fall 2008

Final report

(One paper copy due at project demonstration, electronic submission to lab staff)

Max 15 pages

Visual Synthesizer

Submitted by:

Paul Muri and Alexander Randon

Visualphonics
Note 1. All tables and graphs need captions describing what is presented and need to be discussed in the body text.

Note 2. You need to include circuit diagrams with an explanation and discussion in the text.

Table of Contents (include section headings and page numbers)

Project Abstract ........................................................................................................................................ X
Introduction ........................................................................................................................................ X
Technical Objectives ........................................................................................................................... X
Concept Generation and Selection Matrices ........................................................................................ X
System Level Presentation .................................................................................................................. X
Final Product ........................................................................................................................................ X
Business Case ..................................................................................................................................... X
Study Organization ............................................................................................................................. X
References and Bibliography .............................................................................................................. X
APPENDIX ............................................................................................................................................ X

Project Abstract (update to include latest outcomes)

The Visual Synthesizer is an electronic musical instrument that will integrate a real-time visualizing circuit into a voltage-controlled synthesizer to create both audio and visual effects. The audio outputs via a ¼” phono jack and the visual output will be a NTSC composite video signal to be plugged directly into a television. The television will show a visualization of the audio signal created by the user, having various shapes and colors intelligently change both size and state automatically, in real time. Interacting with both sound and visuals, the Visual Synthesizer brings a new twist to audio synthesis, and thus a new element to musical performance. After much hardware and software design the audio generation and video display is successful.

One-hundred words or less abstract or executive summary. Provide the reader with a high-level description of your project, the technical challenge, and state clearly the final outcome.

Introduction

The voltage-controlled musical synthesizer is probably the most influential musical device since the piano. Synthesizers have expanded our conception of sound and musical composition, giving musicians an utmost control over sound. They were developed by Robert Moog, Donald Buchla, and others in the late 1960’s. Electronic music systems have since evolved over the years. With the advent of digital systems, sound synthesis became more accurate, versatile, portable, and most importantly, cheaper. These digital systems grew, began to replace analog designs, and ultimately lead to the analog synthesizer’s decline in popularity and fabrication. However, since the 1990’s, analog synthesizers have reclaimed their place on the stage and in the studio as musicians gradually rediscovered their unique musical qualities.

As an ode to these celebrated machines, the Visual Synthesizer will provide an innovative, yet inexpensive solution to analog synthesis. With a defining visualization feature and a classic look, sound, and feel, the Visual
Synthesizer is a tool for our intangible thoughts and our unprecedented ideas governing sound, making them an audible, tangible, and now visual reality.

Describe the application domain and the possible need and/or value of your proposed invention. Provide a brief overview of the existing art (if any). Examples may come from existing commercial products, web searches, or articles appearing in the literature. Identify what features differentiates your solution from others.

Technical Objectives

The synthesizer will start with a signal generator with variable waveforms. The signals will then be passed through a filtering circuit and amplified. A low frequency oscillator will be added to modulate the original signal, filter, or amplifier. Next, the generated audio signals will be sent to two locations: An outboard speaker and to a real-time visualization circuit. The visualization circuit will generate simple patterns and colors that actively respond to the incoming signal's amplitude (for example: a larger signal will generate a bigger pattern). The visualization circuit's output is a standard NTSC composite video signal [coaxial phono (formally yellow)] to be plugged into almost any television set made since the 1980's. The video circuit outputs a 22 frame/second signal as specified.

Quantify the performance specifications and/or requirements of your design

Concept Generation and Selection Matrices

Customer/Market Needs: Safe, Reliable, Low cost, Adaptable, Sustain Power failure, Sound Quality, Visual Quality

<table>
<thead>
<tr>
<th>Component</th>
<th>User Interface</th>
<th>Sound Generation</th>
<th>Filter</th>
<th>Amplifier</th>
<th>Gain &amp; Buffering</th>
<th>MicroP</th>
<th>Audio Output</th>
<th>Display</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Knobs</td>
<td>BJT/FET Oscillator Circuit</td>
<td>LFP (active)</td>
<td>ASDR via 4-Qaud Multiplier</td>
<td>Transconductance Op-Amps</td>
<td>Propeller</td>
<td>Outboard via 1/4&quot;</td>
<td>TV</td>
<td>Wall Power</td>
</tr>
<tr>
<td>2</td>
<td>Piano Keyboard</td>
<td>XR-2206 Function Generator</td>
<td>HPF (active)</td>
<td>ASDR via 2-Qaud Multiplier</td>
<td>Current Differenting Op-Amps</td>
<td>ATME</td>
<td>Onboard Speaker</td>
<td>LEDs</td>
<td>Batteries</td>
</tr>
<tr>
<td>3</td>
<td>Remote</td>
<td>566 Function Generator</td>
<td>BPF (active)</td>
<td>MOSFET</td>
<td>PIC</td>
<td></td>
<td></td>
<td>LCD</td>
<td>Solar</td>
</tr>
<tr>
<td>4</td>
<td>Relays</td>
<td>Microphone</td>
<td>Notch (active)</td>
<td>BJT</td>
<td>TI</td>
<td>AMPLI</td>
<td></td>
<td>Crank</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Switches</td>
<td>Line In</td>
<td>State Variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reference Table:

<table>
<thead>
<tr>
<th>Component</th>
<th>User Interface</th>
<th>Sound Generation</th>
<th>Filter</th>
<th>Amplifier</th>
<th>Gain &amp; Buffering</th>
<th>MicroP</th>
<th>Audio Output</th>
<th>Display</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>UI2</td>
<td>S1+S2+S3</td>
<td>F1</td>
<td>A1</td>
<td>G2</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>B</td>
<td>UI1+UI2+UI5</td>
<td>S1+S5</td>
<td>F2</td>
<td>A2</td>
<td>G1+G2+G3</td>
<td>MP2</td>
<td>A1</td>
<td>D1</td>
<td>PM1</td>
</tr>
<tr>
<td>C</td>
<td>UI1</td>
<td>S3+S5</td>
<td>F2</td>
<td>A2</td>
<td>G3</td>
<td>MP3</td>
<td>A2</td>
<td>D2</td>
<td>PM1</td>
</tr>
<tr>
<td>D</td>
<td>UI3</td>
<td>S1+S4</td>
<td>F1+F2</td>
<td>A1</td>
<td>G2</td>
<td>MP2</td>
<td>A2</td>
<td>D3</td>
<td>PM1</td>
</tr>
<tr>
<td>E</td>
<td>UI4</td>
<td>S1+S2+S3</td>
<td>F2</td>
<td>A2</td>
<td>G1</td>
<td>MP1</td>
<td>A1</td>
<td>D4</td>
<td>PM1</td>
</tr>
<tr>
<td>F</td>
<td>UI1</td>
<td>S2+S3+S5</td>
<td>F2</td>
<td>A1</td>
<td>G1</td>
<td>MP1</td>
<td>A1</td>
<td>D1</td>
<td>PM1</td>
</tr>
</tbody>
</table>

Figure 2: Concept generation Matrix

There are many different paths to choose from in assembling each block in the system level design. All the options were entered into a Conceptual Generation Matrix, shown above in figure 2. The best concept, concept B uses knobs, a keyboard, and switches for the user interface. This gives the user...
many different options in customizing what sound they would like to generate. Concept B also uses a BJT/FET oscillation circuit and line inputs to generate the sound. The active state variable filter of concept B will give the user the most control over timbre. An assortment of opamps will be applied to concept B to get gain from the synthesizer. The best microprocessor for providing real-time visualization is parallel’s propeller. Since the purpose of the synthesizer is to provide high quality sound and video the audio and video signals will be sent to an external amplifier and television respectively. For reliability, the device will use standard wall power.

<table>
<thead>
<tr>
<th>Concept</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer/Market</td>
<td>Importance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safe</td>
<td>3</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Reliable</td>
<td>3</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Low cost</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Adaptable</td>
<td>1</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Sustain power failure</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sound Quality</td>
<td>3</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Visualize Quality</td>
<td>3</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Sum +’s</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sum 0’s</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Sum -’s</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Net Score</td>
<td>8</td>
<td>13</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Rank</td>
<td>III</td>
<td>I</td>
<td>III</td>
<td>III</td>
<td>II</td>
<td>III</td>
<td>III</td>
</tr>
<tr>
<td>Continue?</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

**Figure 3: Selection Matrix**

Concept B became the leading concept because it met the most customer and market needs. This is shown in figure 3: Selection Matrix. While all conceptual options proved to be safe and reliable, concept B scored high for providing adaptability because it will employ many controls for sound customization. Concept B also provides the best sound quality with its active state variable filtering and the best visualization with the capabilities of the parallax propeller chip. The outcome of concept B uses two wall power supplies because of the current the product draws.

**Update the example provided in your previous reports if necessary. Comment on outcome**
System Level Presentation

(Modify your earlier version to reflect re-engineering)

- Present the components of your project in terms of a block diagram indicating functional relationships between components using arrows.
- List the specifications for each component

Figure: System Level Diagram

The visual synthesizer consists of two main blocks shown in figure 1. The first block generates the synthesized sound. The user controls the device’s pitch with a piano keyboard and various knobs and switches are used to control the parameters of the sound generator and processors. This block is the analog side. The output signal is then sent to both an outboard amplifier and to the visualization block which creates a video output to the television. Though the voltage control systems that generate audio, the visualize circuitry outputs a high quality video signal to any composite compatible system.

Final Product
The visualize circuit was created to have a 4x4 button pad control each visualization. This required 8 I/O pins of the propeller chip. The other 8 I/O pins were made into headers for future input signals to the visualizer. The rest of the circuit is based on the specification datasheet of the parallax propeller. The chip and EPROM can be programmed from a 4 pin header. Then upon booting the chip loads the program from the EPROM onto the RAM. An audio signal from the D/A converter is processed by the propeller and then a video signal is created and outputted to three output pins: 27, 28, and 29. These make up the video generator's RGB signals which then go through a resistive load to create a 1 volt, 3 bit DAC that is used to create a baseband broadcast analog video signal.

The power regulation circuit was created to generate a +15 and -15 voltage for the components of the synthesizer. An A/C to D/C adapter creates a 30V DC signal which then gets regulated to 15V. All devices then use 15V as ground. So, 30V is +15V to the devices and 0V is -15V.
Visuals:

Upon running the visualizer many custom display routines can be generated. The 16 button panel of the synthesizer allows the user to select which routine to display. The frames are generated at the standard 22 frame/second rate. The processor runs a 4 color scheme for each display which can be modified from the button pad as well.
Business Case

Present prototype and mass manufacturing costs. Calculate NPV and IIR. Provide comments and conclusions.

### Prototype Development Cost

<table>
<thead>
<tr>
<th>Description</th>
<th>Rate/unit</th>
<th>Qty</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic parts miscellaneous</td>
<td></td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Propeller processor</td>
<td>12</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>PCB boards</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Keypad and PCB</td>
<td>20</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Enclosure</td>
<td>25</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>$152.00</strong></td>
</tr>
</tbody>
</table>

### Manufacturing Cost per Unit

<table>
<thead>
<tr>
<th>Description</th>
<th>Rate/unit</th>
<th>Qty</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labor (hourly, unburdened)</td>
<td>10</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Insurance (workman's comp, etc)-20%</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Overhead (equipment,expendables,rent,etc)-100%</td>
<td>10</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Electronic parts miscellaneous (discount?)</td>
<td>50</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>LCD screen (4 lines)</td>
<td>22.45</td>
<td>1</td>
<td>22.45</td>
</tr>
<tr>
<td>Rechargeable battery,charger</td>
<td>20</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Propeller processor</td>
<td>6.37</td>
<td>1</td>
<td>6.37</td>
</tr>
<tr>
<td>PCB boards</td>
<td>5</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Keypad and PCB</td>
<td>10</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Enclosure</td>
<td>25</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>$214.82</strong></td>
</tr>
</tbody>
</table>

### Start-Up Costs

<table>
<thead>
<tr>
<th>Description</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools (soldering -RoHS lead free compliant,clamps,cables)</td>
<td>1,000</td>
</tr>
<tr>
<td>Programmers</td>
<td>48</td>
</tr>
<tr>
<td>NIH/FDA/ISO test and approval (model)</td>
<td>10,000</td>
</tr>
<tr>
<td>Computer/Oscilloscope/Powersupplies</td>
<td>15,000</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$26,048.00</strong></td>
</tr>
</tbody>
</table>

### General Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product life (years)</td>
<td>30</td>
</tr>
<tr>
<td>Production first year</td>
<td>100</td>
</tr>
<tr>
<td>Production subsequent years</td>
<td>100</td>
</tr>
<tr>
<td>Sale price(excl. shipping)</td>
<td>500</td>
</tr>
<tr>
<td>Advertising per year</td>
<td>1,000</td>
</tr>
<tr>
<td>Cost of Capital/discount rate</td>
<td>10.00%</td>
</tr>
</tbody>
</table>
Analysis
Sales
Production cost
Cash flow (sales-production cost)
Net Present Value - NPV=Sum_t(PV/(1+r)^t)-II
Internal Rate of Return - II=Sum_t(Ct/(1+IRR)^t)

Year 0

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Investment (II)</td>
<td>-$26,048.00</td>
</tr>
<tr>
<td>Start-up</td>
<td>$68,204.59</td>
</tr>
<tr>
<td>95%</td>
<td></td>
</tr>
</tbody>
</table>

Year 1 Year 2 Year 3 Year 4 Year 5

<table>
<thead>
<tr>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Year 4</th>
<th>Year 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>50,000</td>
<td>50,000</td>
<td>50,000</td>
<td>50,000</td>
</tr>
<tr>
<td>26,048</td>
<td>22,482</td>
<td>22,482</td>
<td>22,482</td>
<td>22,482</td>
</tr>
<tr>
<td>23,952</td>
<td>27,518</td>
<td>27,518</td>
<td>27,518</td>
<td>27,518</td>
</tr>
</tbody>
</table>

Study Organization

Present your original and final Gantt chart and discuss any changes.

Original Gantt Chart

Available Weeks in Fall 2008

Figure 4: Gantt Chart
The Gantt chart in Figure 4 shows the tasks delegated week by week. Paul is leading the activities of the visualizer. This includes code design and testing of the visualizer circuitry. Alex is the head of the synthesizer circuit design block. This involves arduous research to first understand musical engineering principles, and then numerous trials to build the most cost efficient, high quality analog designs.

**Final Gantt-Chart:**

Many extentions were needed from the original Gantt chart. This was because many of the processes relied on each other. Components which came in late delay population of PCBs. Code design came along slower than expected. Troubleshooting PCBs delayed the project a few days as well.

**References or Bibliography**

http://makezine.com/14/pixelm3/
http://www.parallax.com/propeller/
**APPENDIX**

Visualizer Source Code:

```plaintext
CON

_clkmode = xtall + pll16x  'set clock speed to 80 MHz (16 x external crystal

_xinfreq = 5_000_000
_stack = ($2000 + $3000 + 100) >> 2   'accomodate display memory and stack

x_tiles = 16
y_tiles = 12

_stack = ($2000 + $3000 + 100) >> 2   'accomodate display memory and stack

paramcount = 14
bitmap_base = $2000
display_base = $5000

'can increase or decrease this number to make more or less time between color or
shape changes
_DELAYUNIT = 1_000_000  '1/8 of a sec was 10_000_000

' set up the Analog to Digital Converter (ADC) Pins on the Propeller
_ADC_CLK       = 0  'Clock to ADC
_ADC_DIN       = 1  'Data to/from ADC
_ADC_CS        = 2  'CS to ADC

' how many adc cycles before taking an avg value, lower values: more
speedy/responsive, higher values: more smooth
_ADC_AVG = 125       'old: 75

VAR

' set the tv object variables...
long  tv_status     '0/1/2 = off/visible/invisible           read-only
long  tv_enable     '0/? = off/on                            write-only
long  tv_pins       '%ppmmm = pins                           write-only
long  tv_mode       '%ccinp = chroma,interlace,ntsc/pal,swap write-only
long  tv_screen     'pointer to screen (words)               write-only
long  tv_colors     'pointer to colors (longs)                 write-only
long  tv_hc         'horizontal cells                           write-only
long  tv_vc         'vertical cells                             write-only
long  tv_hx         'horizontal cell expansion                  write-only
long  tv_vx         'vertical cell expansion                    write-only
long  tv_ho         'horizontal offset16                       write-only
long  tv_vo         'vertical offset16                          write-only
long  tv_broadcast  'broadcast frequency (Hz)                   write-only
long  tv_auralcog   'aural fm cog                              write-only

word  screen[x_tiles * y_tiles]
long  colors[64]
long  stk[16]
long  displayStack1[16]          'a stack of longs used by display routine 1

byte lastrand
byte delayflag

OBJ

'using propeller's included tv, graphics and mcp3208/adc objects

  tv : "tv"
  gr : "graphics"
  adc : "MCP3208"
```
'turn on the external LED
dira[16]~
outa[16]~

'start the adc object
adc.start(ADC_DIN, ADC_CLK, ADC_CS, $FF)

'start tv object
longmove(@tv_status, @tvparams, paramcount)
tv_screen := @screen
tv_colors := @colors
tv.start(@tv_status)

'start and setup graphics object
gr.start
gr.setup(16, 12, 128, 96, bitmap_base)

''************************************************
**************************
''  COLOR PALETTES
''  These are the colors that the PM3K will cycle through
''  use 4 colors onscreen at a time and one color is always black
''  To get more color variety, create different palettes and combos of colors
''  Each palette group has a slight variation in the order or hue of the colors
''  This is exploited by the shapes which also arrange the color bits with variation
''  Refer to 'Graphics_Palette.spin' in the Example Library if you would like to pick
different color values
''***************************************************************************

'browns and yellows
SetColorPalette(0,$02,$ac,$18,$ab)
SetColorPalette(1,$02,$18,$9e,$ab)
SetColorPalette(2,$02,$bb,$18,$9b)
SetColorPalette(3,$02,$ac,$18,$9b)
SetColorPalette(4,$02,$ab,$ac,$18)
SetColorPalette(5,$02,$ac,$18,$9b)

'blues and purples
SetColorPalette(6,$02,$3c,$b8,$eb)
SetColorPalette(7,$02,$0c,$5e,$fb)
SetColorPalette(8,$02,$b8,$3c,$eb)
SetColorPalette(9,$02,$b8,$5d,$88)
SetColorPalette(10,$02,$eb,$3c,$5d)
SetColorPalette(11,$02,$b8,$eb,$88)

'purple pink white
SetColorPalette(12,$02,$68,$cc,$4d)
SetColorPalette(13,$02,$58,$88,$3e)
SetColorPalette(14,$02,$sed,$dd,$3e)
SetColorPalette(15,$02,$38,$b8,$5d)

'old 15 SetColorPalette(15,$02,$58,$88,$3e)

'dd light pink
SetColorPalette(16,$3e,$7c,$7e,$5c)

'old 16 SetColorPalette(16,$02,$68,$cc,$4d)

'58 pink
SetColorPalette(17,$02,$3e,$b8,$58)

'light blues and greens
SetColorPalette(18,$02,$5e,$3d,$f8)
SetColorPalette(19,$02,$3d,$5e,$f8)
SetColorPalette(20,$02,$6d,$3e,$6e)
SetColorPalette(21,$02,$5e,$4e,$f8)
SetColorPalette(22,$02,$6e,$3d,$4e)
SetColorPalette(23,$02,$f8,$6e,$3e)

'orange and greens
SetColorPalette(24,$02,$8d,$8e,$38)
SetColorPalette(25,$02,$8e,$8d,$28)
SetColorPalette(26,$02,$5c,$8e,$38)
SetColorPalette(27,$02,$7c,$7e,$5c)
SetColorPalette(28,$02,$8d,$38)
SetColorPalette(29,$02,$28,$8e,$5c)

'start a new delay counter in its own cog
'delayflag:=0
'cognew(delay(@delayflag, 8), @delayStack)

'the screen is made of 16 horizontal x 12 vertical tiles (groups of pixels)
'while each tile can have its own color palette, we set a color palette to apply globally across all screen tiles

displayroutine:=1
SetAreaColor(0,0,TV_HC-1,TV_VC-1,0)

'setup of a button scan routine in a separate cog
cognew(SetDisplayRoutine1(@displayroutine), @displayStack1)
'cognew(SetDisplayRoutine2(@displayroutine), @displayStack2)

repeat  'repeat forever
'outa[4]:=INA[3]

'**********************************************************************************************
'' retrieve ADC L and R values, build pixel scalar values
   rightADCavg := adc.average(0,ADC_AVG) 'right channel volume/voltage
   leftADCavg := adc.average (1,ADC_AVG) 'left channel volume/voltage

   scalar_r := (rightADCavg*16/4096) 'we're massaging the number into something applicable to the shapes
   scalar_l := (leftADCavg*16/4096)
   scalar_bigd :=((scalar_r)*2/5) +16

   totalavg:=(scalar_r+scalar_l)/2 'we will use totalavg to help make sure not to change anything if no input

   'Temp := INA[3]     'get P3 value
   'if Temp== 1
   '  outa[4] :=1      'light LED when P3 is high
   'else
   '  outa[4] :=0

'**********************************************************************************************
'' change the color and shape?

'if totalavg > 4 'if we're getting some adc values
   'if delayflag==1 'and the random time delay has elapsed (to make sure we aren't changing too fast)
      'i++
      'if i>29
      'i:=0

   'SetAreaColor(0,0,TV_HC-1,TV_VC-1,i) 'pick a new color palette comment this out to keep one color
'delayflag:=0
'displayroutine:=7
'SetAreaColor(0,0,TV_HC-1,TV_VC-1,16)
'SetDisplayRoutine(Temp, displayroutine)

'*******************************************************************
' change the orientation? some of the screen patterns have alternating rotations
if j//9== 0
  rotateflag := !rotateflag

'*******************************************************************
Translate L+R inputs to simple shapes and colors
PM3K has 8 total screen pattern/arrangements of shapes onscreen
We cycle through these patterns randomly when there have been adc level changes
and a certain amount of time has passed
The shapes in the layout are grown or shrunk based in the ADC values
To see what a particular screen pattern looks like just hardcode the
displayroutine var to a number between 0 and 7
gr.clear

case displayroutine
  1:
' stripes
  gr.width(scalar_l+16)
  gr.pix(0, 0, 0, @pixdeftriclear3)
  gr.width(scalar_r+16)
  if rotateflag
    gr.pix(0, 0, 0, @pixdeftriclear2a)
  else
    gr.pix(0, 0, 1, @pixdeftriclear2b)
  2:
' multiples5
' do middle
  gr.width(scalar_r+16)
  gr.pix(0, 0, 0, @pixdeftriclear3)
  gr.width(scalar_l+16)
  gr.pix(0, 0, 0, @pixdefmed2)

' do sides
  gr.width(2)
  if j<8
    gr.pix(90, j/4, 0, @pixdefF)
  else
    gr.pix(90, (-j+8)/4, 0, @pixdefF)
  if j<8
    gr.pix(-90, (-j+8)/4, 0, @pixdefU)
  else
    gr.pix(-90, j/4, 0, @pixdefU)

' update screen bitmap
gr.copy(display_base)

if j ==16
  j:=0
else
    j++

lastscalar := scalar_r

' 3 screen color pubs
Pub SetAreaColor(X1,Y1,X2,Y2,ColorIndex)|DX,DY
    Repeat DX from X1 to X2
    Repeat DY from Y1 to Y2
    SetTileColor(DX,DY,ColorIndex)

Pub SetTileColor(x, y, ColorIndex)
    screen[y * tv_vc + x] := display_base >> 6 + y + x * tv_vc + ((ColorIndex & $3F) << 10)

Pub SetColorPalette(ColorIndex,Color1,Color2,Color3,Color4)
    colors[ColorIndex] := (Color1) + (Color2 << 8) + (Color3 << 16) + (Color4 << 24)

pub SetDisplayRoutine1 (d) | Temp
    dira[10..3] := %11110000 'set the data direction register for the direction of pin to scan the buttons
    color := 0
    repeat
        outa[10..7] := %0001
        Temp := ina[6..3] 'look at input pins 3, 4, 5, 6
        if Temp == %0001
            long[d] := 1
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,1)
        elseif Temp == %0010
            long[d] := 2
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,2)
        elseif Temp == %0100
            long[d] := 1
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,3)
        elseif Temp == %1000
            long[d] := 2
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,4)
        outa[10..7] := %0010
        Temp := ina[6..3] 'look at input pins 3, 4, 5, 6
        if Temp == %0001
            long[d] := 1
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,5)
        elseif Temp == %0010
            long[d] := 2
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,6)
        elseif Temp == %0100
            long[d] := 1
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,7)
        elseif Temp == %1000
            long[d] := 2
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,8)
        outa[10..7] := %0100
        Temp := ina[6..3] 'look at input pins 3, 4, 5, 6
        if Temp == %0001
            long[d] := 1
            SetAreaColor(0,0,TV_HC-1,TV_VC-1,9)


```c
if Temp==%0001
    long[d]:=1
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,13)
elseif Temp==%0010
    long[d]:=2
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,14)
elseif Temp==%0100
    long[d]:=1
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,15)
elseif Temp==%1000
    long[d]:=1
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,16)
```

```
outa[10..7]:=%1000
Temp:=ina[6..3]   'look at input pins 3, 4, 5, 6
```

```
if Temp==%0001
    long[d]:=1
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,13)
elseif Temp==%0010
    long[d]:=2
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,14)
elseif Temp==%0100
    long[d]:=1
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,15)
elseif Temp==%1000
    long[d]:=1
    SetAreaColor(0,0,TV_HC-1,TV_VC-1,16)
```