

A decorative background pattern of orange circuit board traces and nodes on a dark red gradient background.

# UNIQUE FEATURES OF A 40/20 METER HOMEBREW RECEIVER

LEE SNOOK W1DN

HIGHWAY 285 TECHCONNECT CLUB

TECHFEST

NOVEMBER 3, 2018

# PROJECT GOALS

- Simple receiver design with readily available parts
- Adequate specifications for general use
- At home fabrication techniques
- Simple reconfiguration for the Arduino Controller and LCD Display Layout
- 50 ohm Interstage coupling for easy testing of each stage

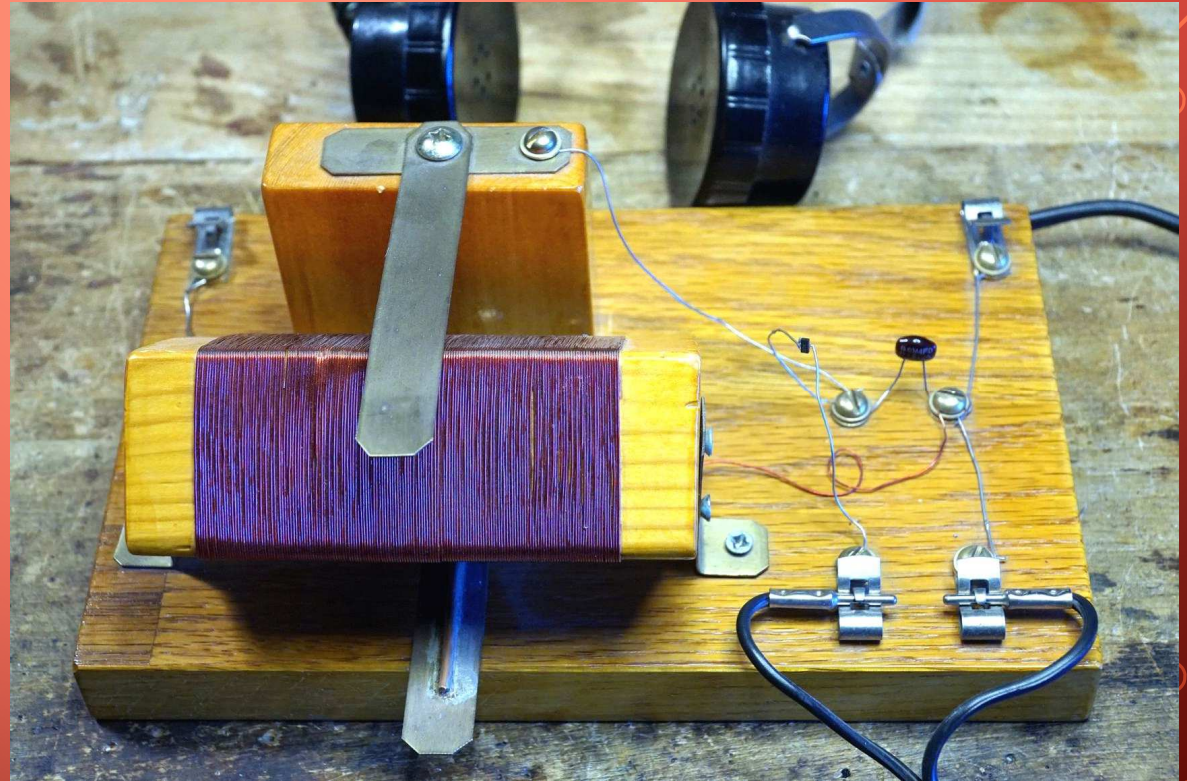
# WILL FOCUS ON THE FOLLOWING TOPICS

- History
- Block Diagram and Overview
- General Specifications
- LO Frequency Selection
- I.F. Crystal Filter Design
- Microprocessor Controlled (ATMEGA 328)
  - Arduino Based Development System
- 7 inch Color LCD Touch Display
  - 4D Systems IDE for display configuration
- At home PCB fabrication
- Single Sided PCB, Maximize Ground Plane
- Rotary Encoder Tuning Knob with selectable increments
- Variable Lowpass Audio Filter
- Two Local Oscillator DDS synthesizers
- Solid state front end switching
- Band Spectrum Display
- 10 User Memories
- Simulated Analog S Meter
- Externally programmable through a serial port
- 50 ohm Interstage coupling impedance
- Passband Offset Tuning



# HISTORY

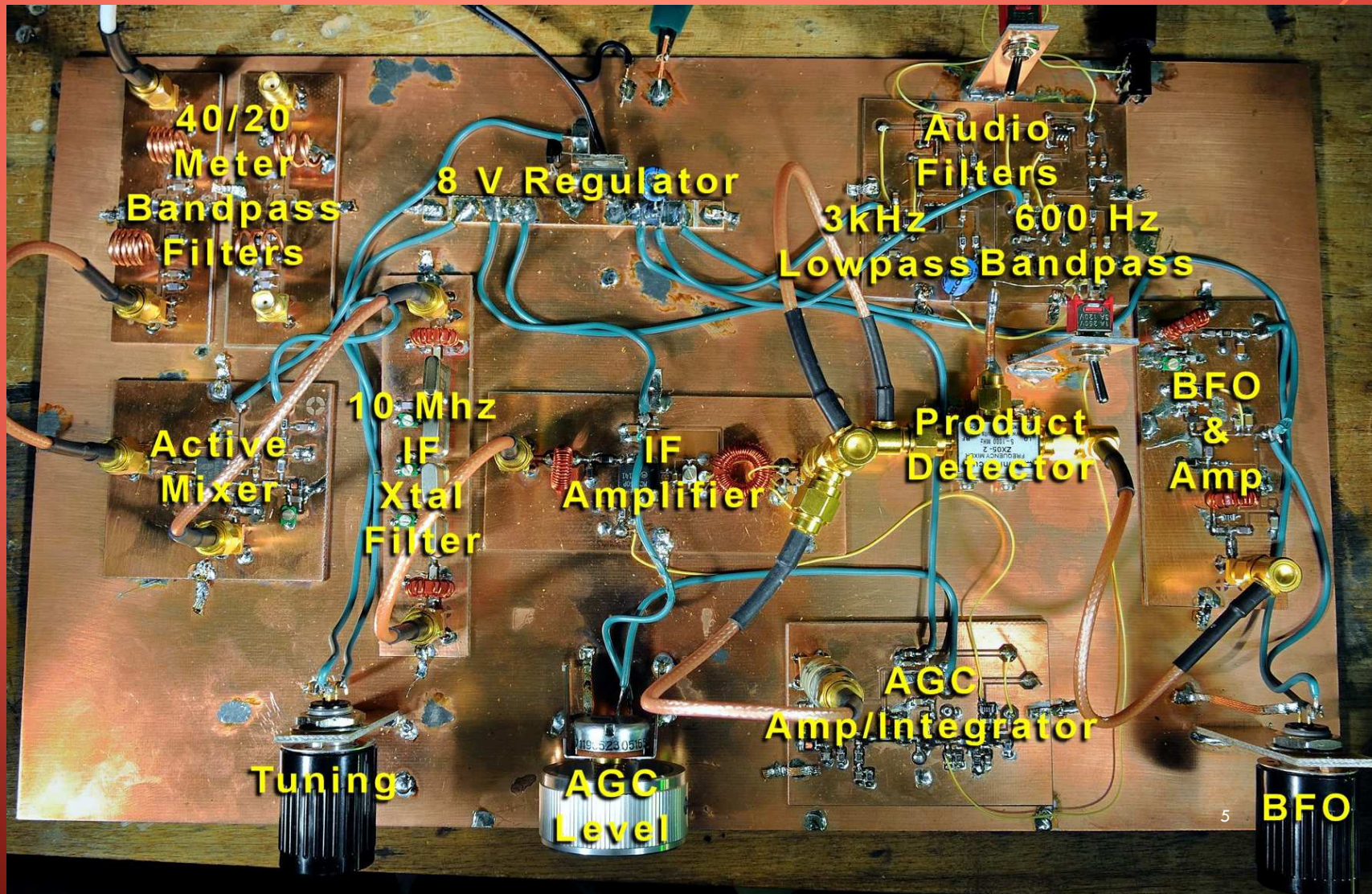
- First Receiver – Foxhole Radio
  - Razor Blade as a crystal detector
  - Of course, 110 turns on a toilet paper roll
- Second Receiver – Dual Slide Tuning
  - 5<sup>th</sup> Grade
  - 11 years old





# HISTORY

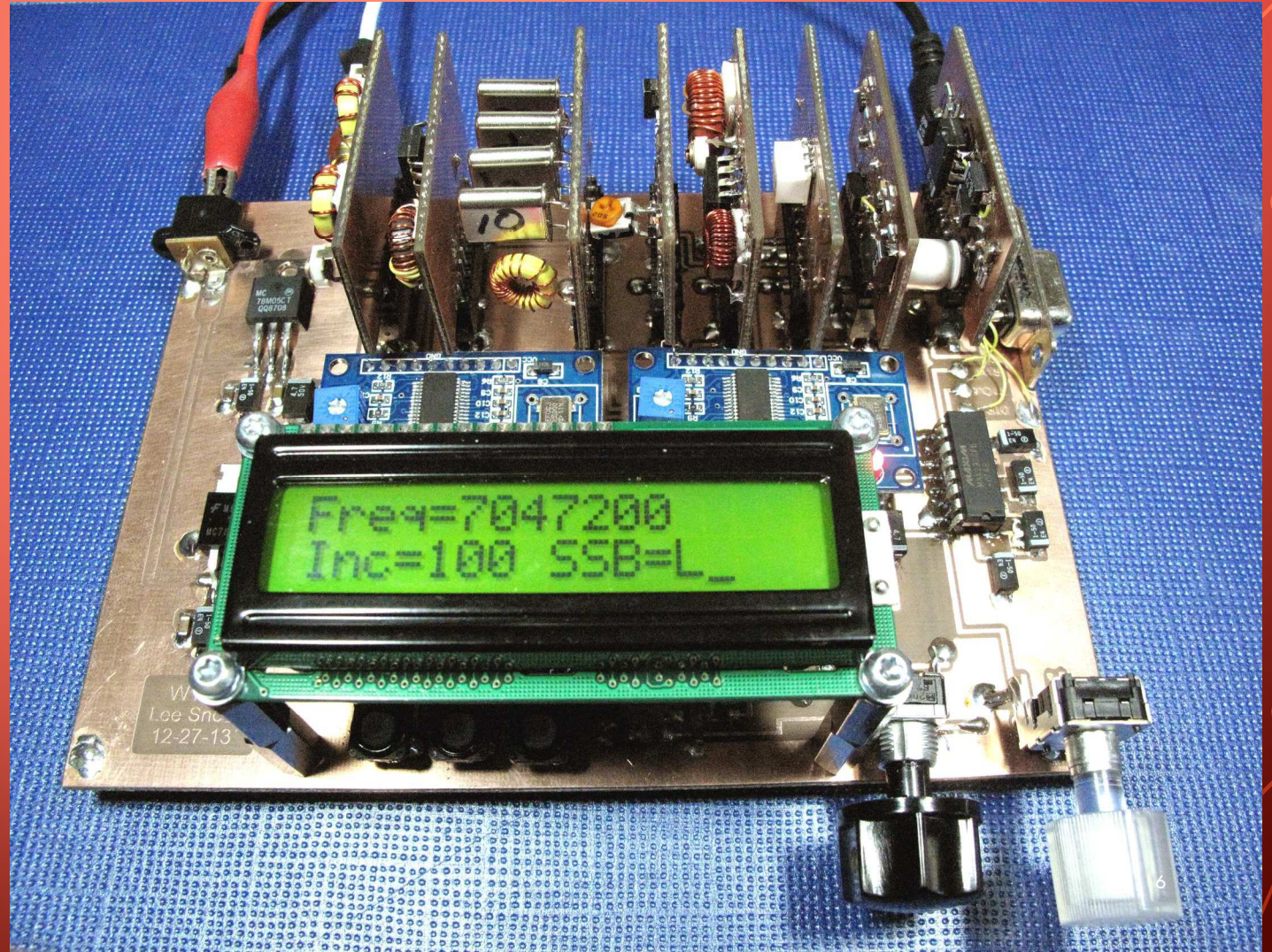
MY FIRST MODULAR  
RECEIVER BREADBOARD.  
EACH MODULE IS ON  
IT'S OWN PCB WITH 50  
OHMS IN AND OUT





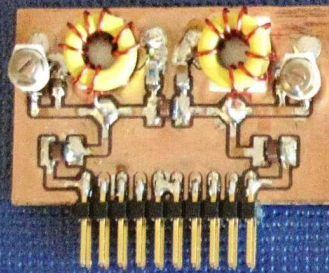
# HISTORY

SECOND RECEIVER  
USING INDIVIDUAL  
PLUG IN MODULES  
AND A LCD DISPLAY

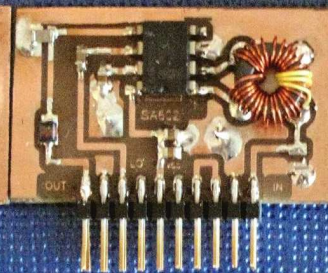




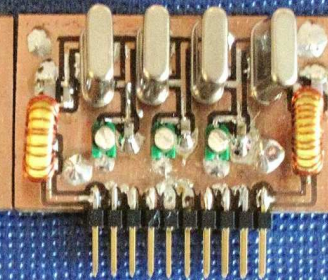
**7.15 Front End  
Bandpass Filter**



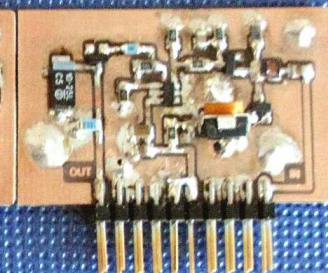
**NE602 Mixer**



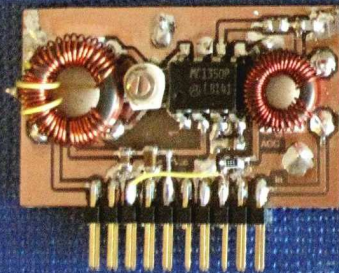
**I.F. Filter  
10 Mhz**



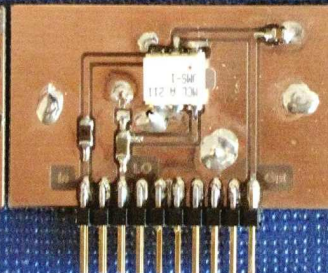
**Automatic  
Gain Control**



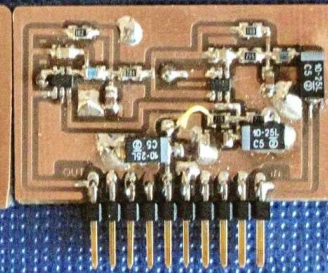
**I. F. Amplifier**



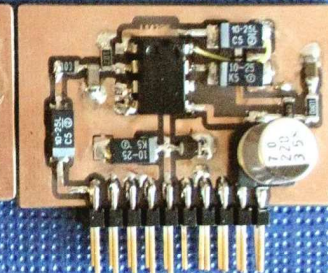
**Product Detector  
Mixer JMS-1**



**2.4 kHz  
Low Pass Filter**



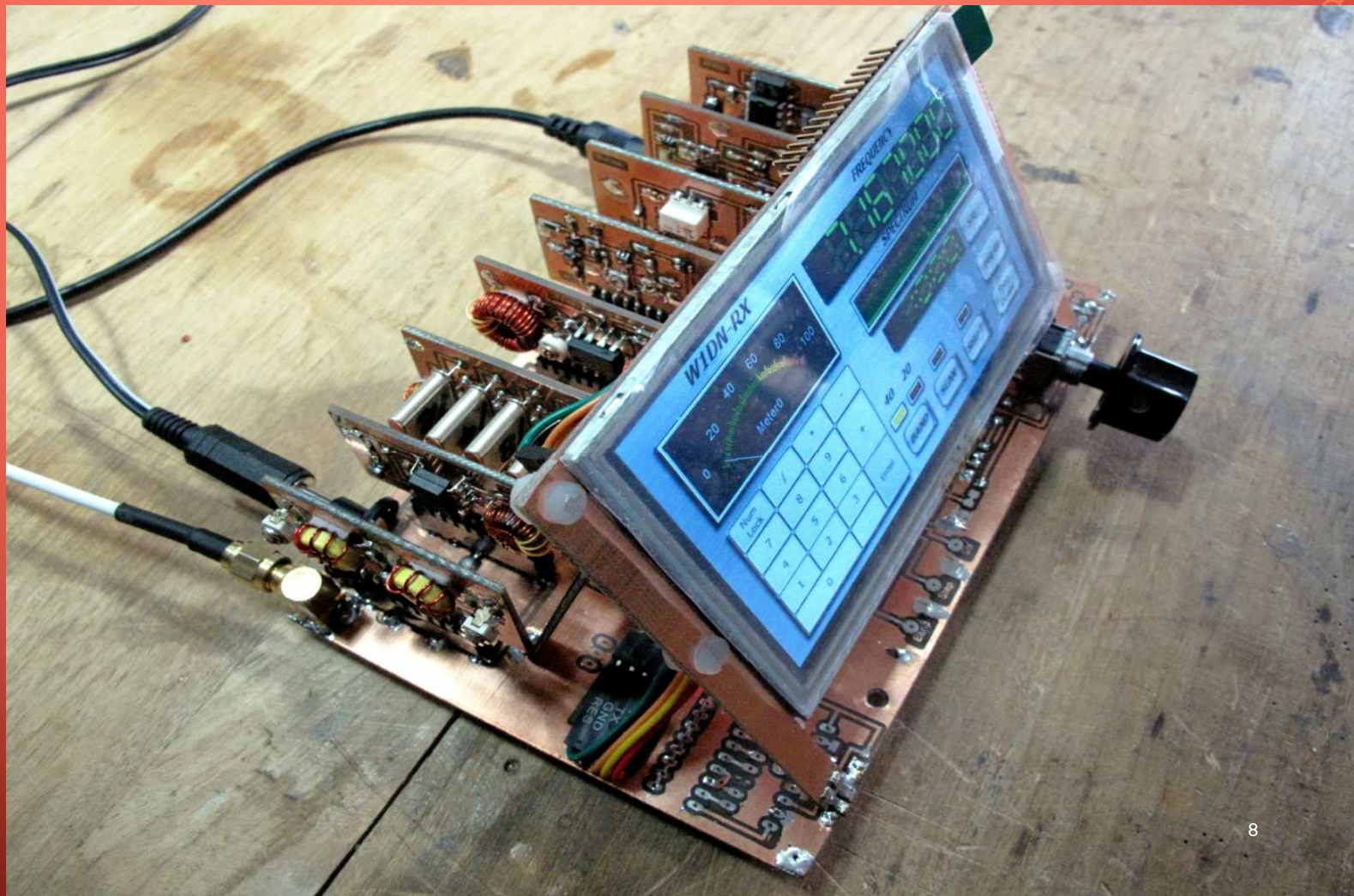
**Audio Amp**





# HISTORY

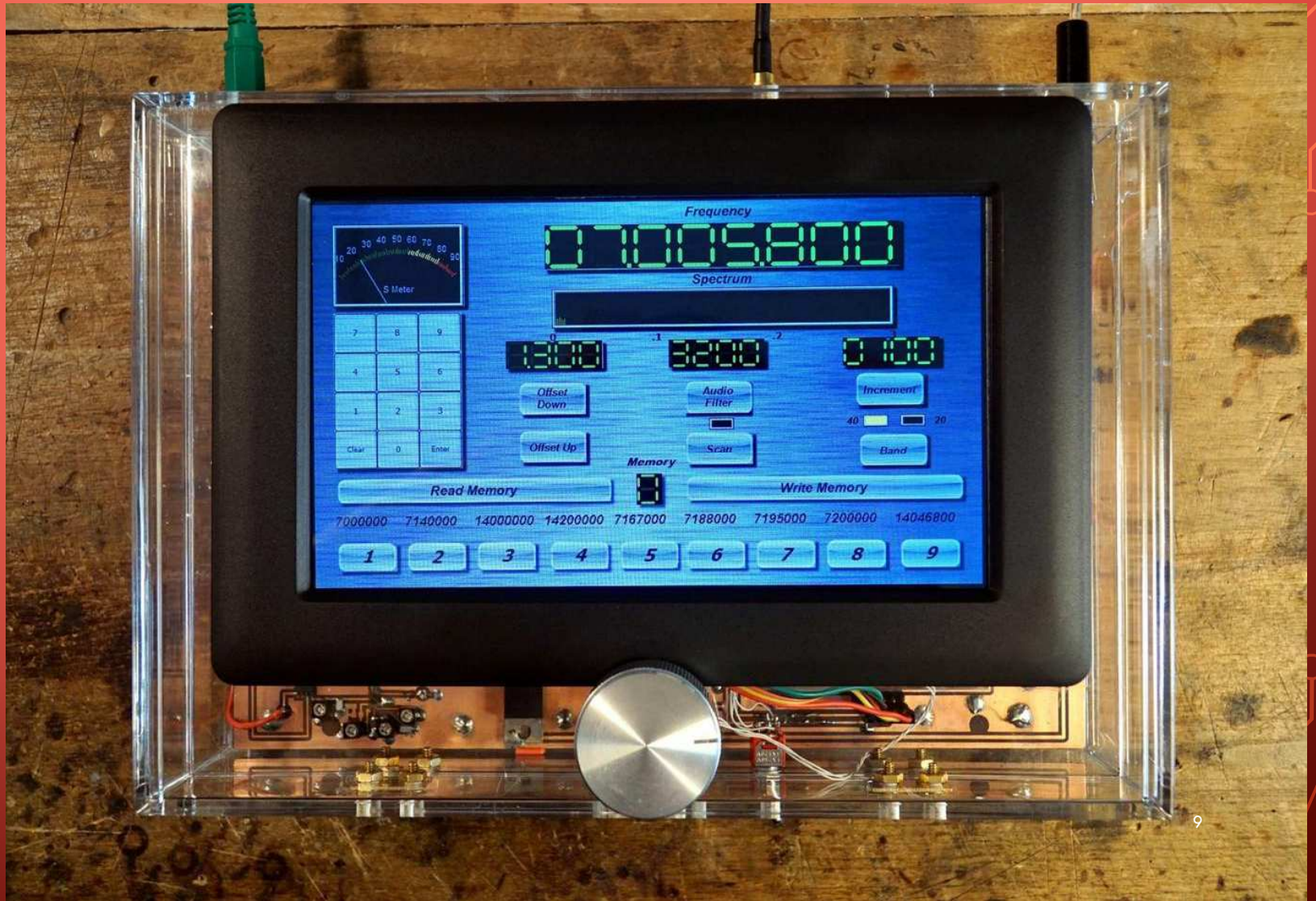
ITERATION WITH  
4D SYSTEMS  
COLOR TOUCH 4  
INCH DISPLAY





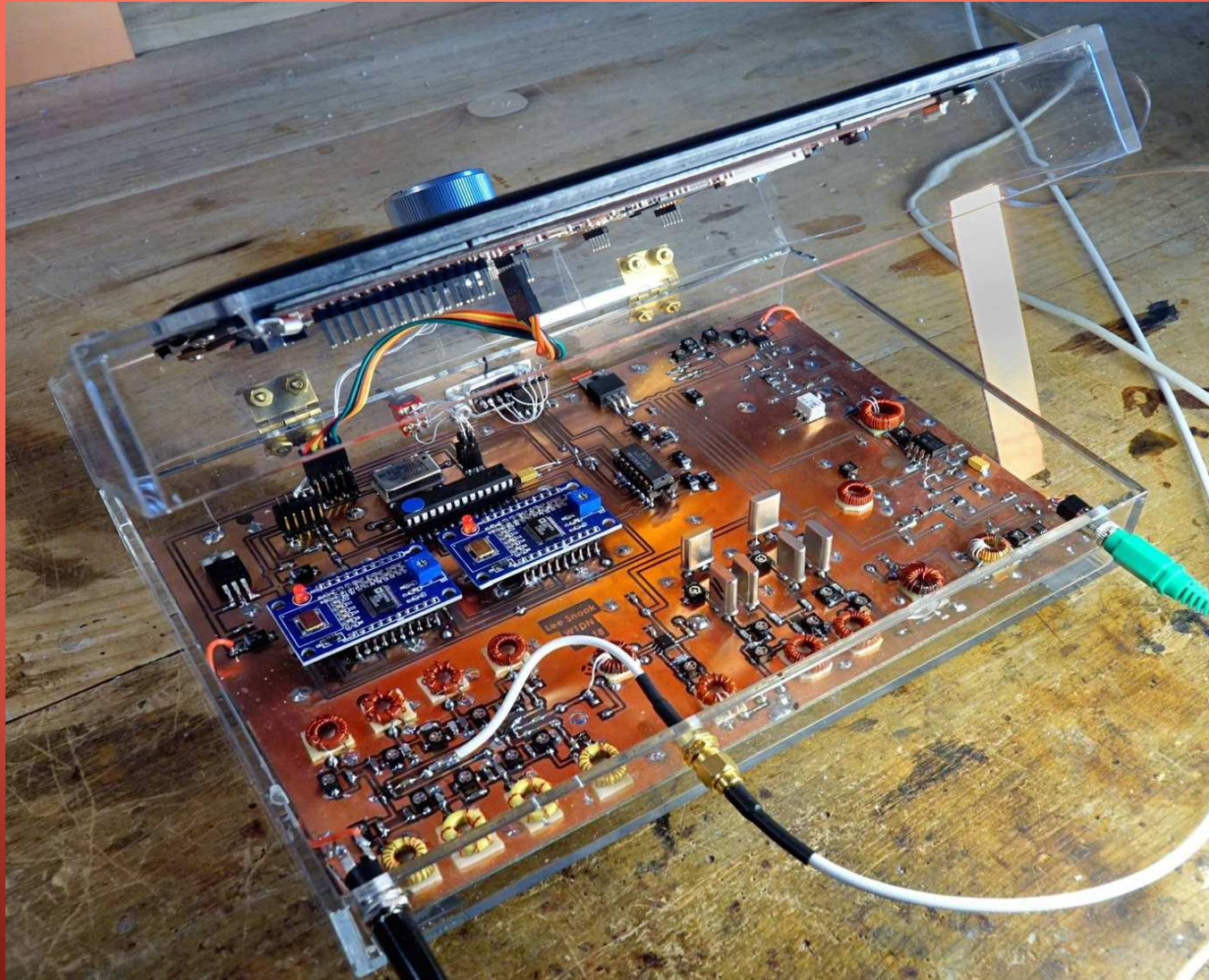
# HISTORY

FOURTH  
RECEIVER  
(TODAY)

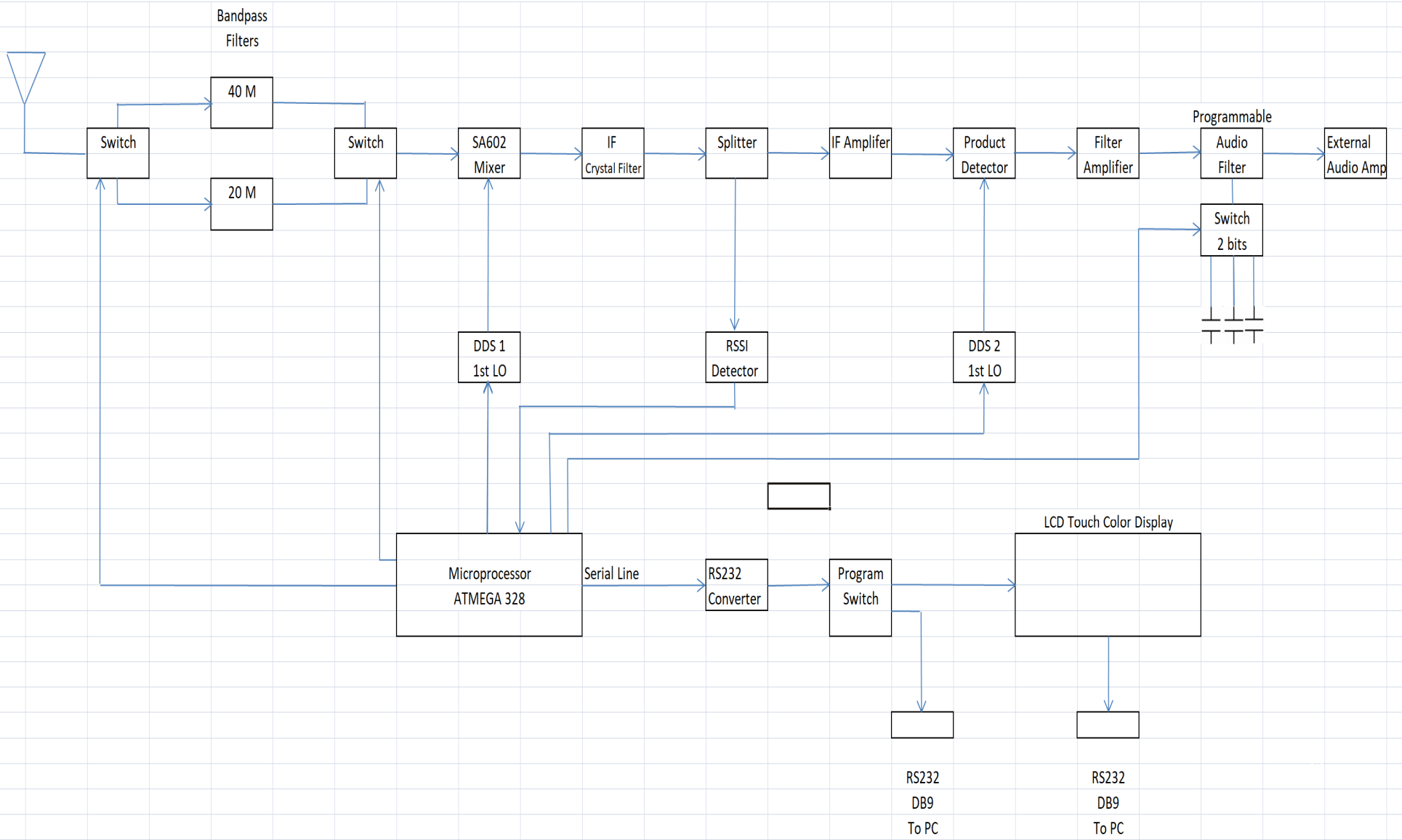




Internal view







Program Switch

DB9 Serial Connector

Programmable

Xtal 10 Mhz

Switch & Caps

Audio Filter

Micro Processor ATMEGA328

Audio Amp

DDS1 & DDS2 AD9850

RS232 Converter

Product Detector Mixer (JMS-1)

I.F. Amplifier MC1350

Front End Filters

Mixer SA602

I.F. Crystal Filter

Splitter

RSSI Detector AD8310

Lee Snook WIDN 4-9-18



# GENERAL SPECIFICATIONS

- Sensitivity  $-120 \text{ dBm}$   $\text{BW} = 3200 \text{ Hz}$
- IMD  $-42 \text{ dBm}$
- Third Order Intercept  $((120-42)/2+42 = -3 \text{ dBm})$
- SFDR  $120-42 = 92 \text{ dbm}$

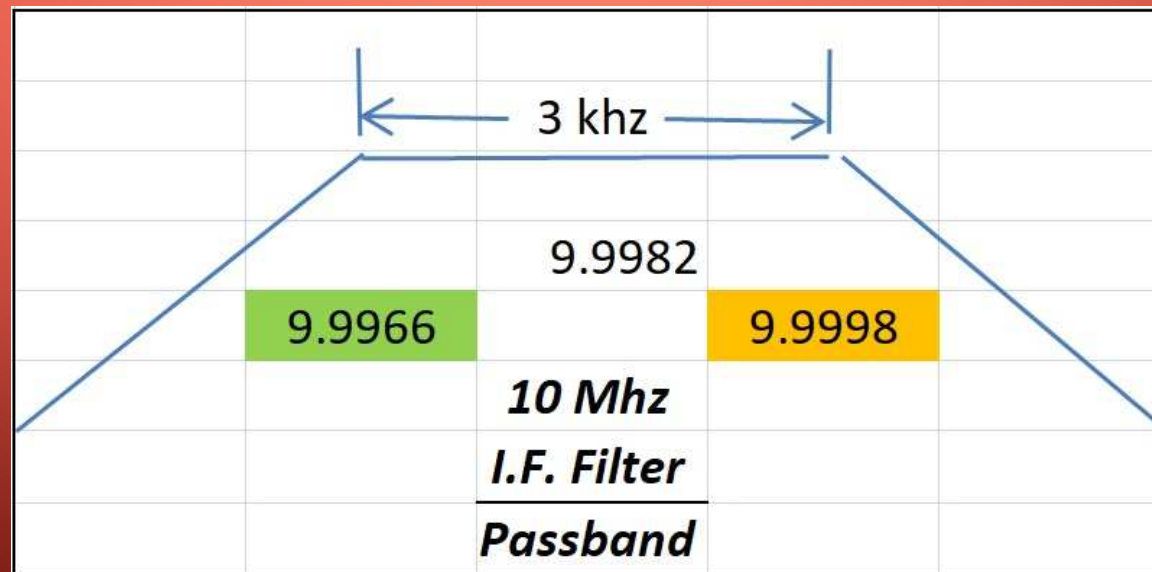


## LOCAL AND PRODUCT DETECTOR FREQUENCY SELECTION TABLE

Select a Local oscillator (LO) such that:

(use the lower cutoff edge since it has the steepest shape factor)

- As the receive frequency increase, the LO increases
- For LSB and USB the sideband falls within the crystal filter passband



# 40 METERS

## RECEIVE 7 MHZ

1<sup>ST</sup> LO=16.9966 Mhz

I.F. =  $16.9966 - 7 = 9.996598$  MHZ

NOTE; THE LOWER EDGE OF THE I.F. CRYSTAL FILTER

### • Case 1

- As the receive frequency increases
- the LO frequency increases
- $LO = IF + RF$

### • Case 2 (For LSB)

- As the Receive Lower Sideband Envelope decreases
- the IF increases
- $IF = LO - RF$

# 20 METERS

## RECEIVE 14 MHZ

1<sup>ST</sup> LO=4.003402 Mhz

$$IF = RF - LO$$

- Case 1

- As the receive frequency increases
- the LO frequency increases
- $LO = IF + RF$

- Case 2 (For LSB)

- the Receive Lower Sideband Envelope decreases
- the IF increases
- $IF = LO - RF$



## EXAMPLES (CONT.)

### RECEIVE 14 MHZ

1<sup>ST</sup> LO=4.003402 MHZ

I.F. = 14-4.003402= 9.996598 MHZ

NOTE; THE LOWER EDGE OF THE I.F. CRYSTAL FILTER

- Case 1; As the receive frequency increase, the LO frequency increases
- Receive 14.1 Mhz
- 1<sup>st</sup> LO=4.003402 Mhz
- I.F. = 14.1 – 4.003402 = 10.096598 Mhz  
Note; The Lower Edge of the I.F. Crystal Filter
- Case 2; For LSB, the Receive Lower Sideband Envelope decreases, the IF increases
- Receive 6.999 Mhz, 1 khz audio tone
- 1<sup>st</sup> LO=16.9966 Mhz
- I.F. = 16.9966 – 6.999 = 9.9976 Mhz  
Note; The LSB envelope is within the passband

# CRYSTAL FILTER DESIGN

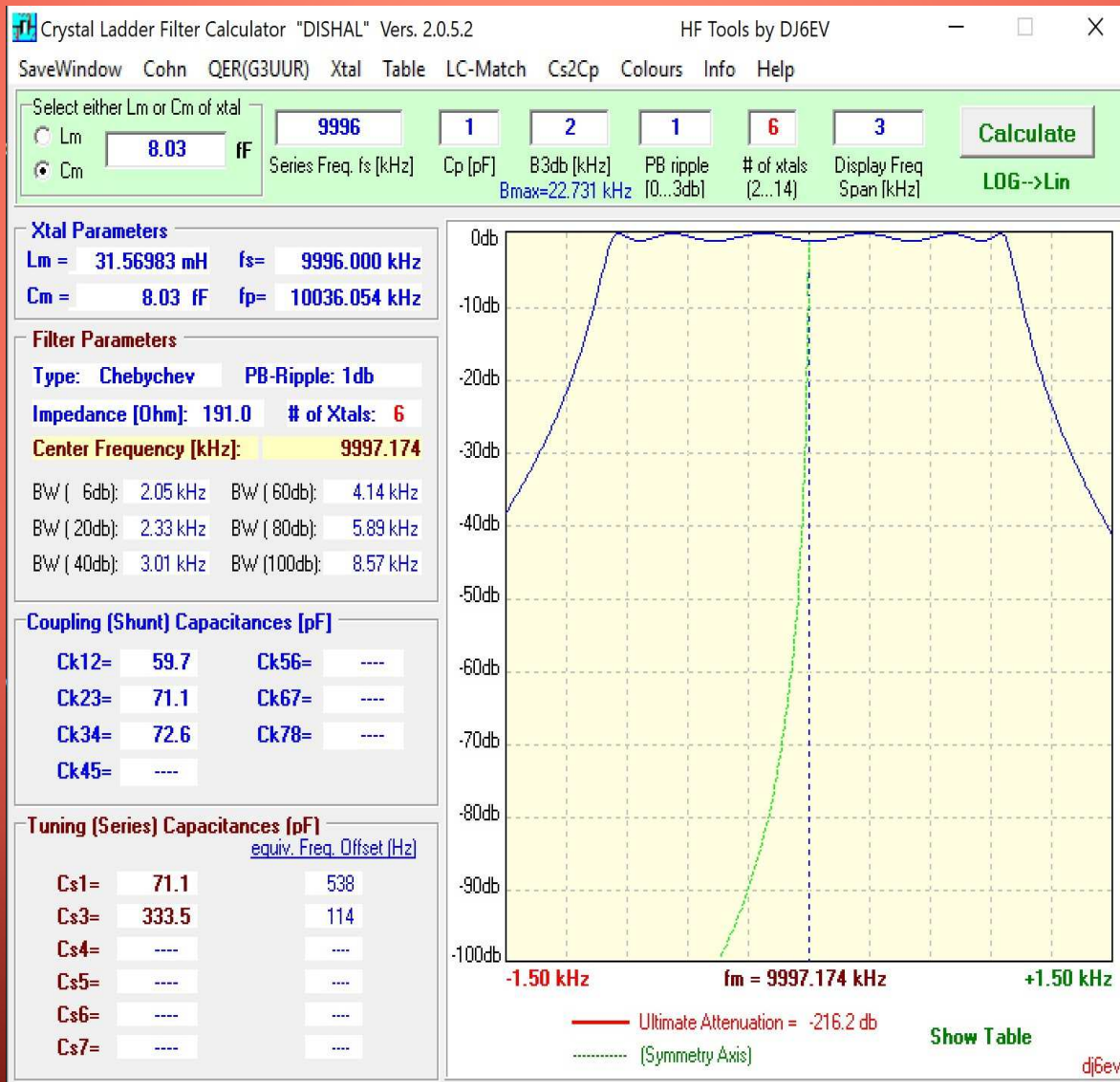
MEASURE A GROUP OF 10 MHZ CRYSTALS AND SELECT 6 THAT HAVE THE CLOSEST PARAMETERS

Xtal	RL	Loss in dB	Rm	Reff	Minus 45 fs	45 fl	Plus 45 fu	deltal f	Cm	Lm
11	12.5	2.62	8.801814	33.80181	9995949	9996160	9996332	172	8.10925E-15	0.03129334
12	12.5	2.37	7.842787	32.84279	9995918	9996130	9996297	167	8.10348E-15	0.031315828
18	12.5	2.49	8.299676	33.29968	9995912	9996126	9996289	163	7.80087E-15	0.032530651
22	12.5	2.97	10.19168	35.19168	9995893	9996113	9996292	179	8.10607E-15	0.031305982
Average:			fl	fc	fh	9996132			8.02992E-15	0.03161145
			9996215	9997815	9999415					



## CRYSTAL FILTER DESIGN (CONT.)

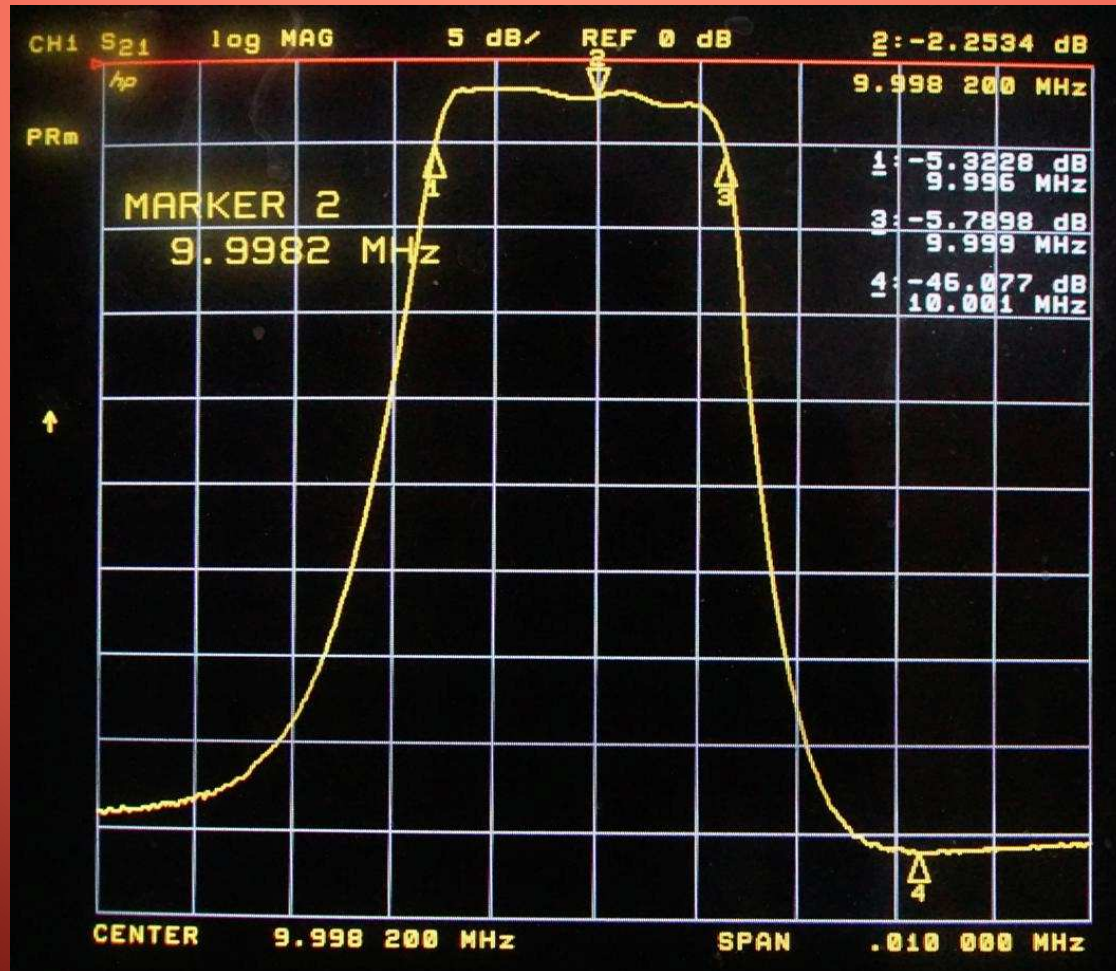
- For a review of Crystal Filter Design use the following reference  
*Crystal Motional Parameters, A Comparison of Measurement Approaches*  
Jack R. Smith K8ZOA, 11 June 2006
- Using the Freeware Program; DISHAL vers 2.0.5.2;  
Enter  $L_m$ ,  $C_m$ ,  $C_p$ , BW, # of Xtals, Ripple





# MEASURED RESULT

BANDWIDTH FOR THIS FILTER = 3 kHz



# MICROPROCESSOR (ATMEGA 328)

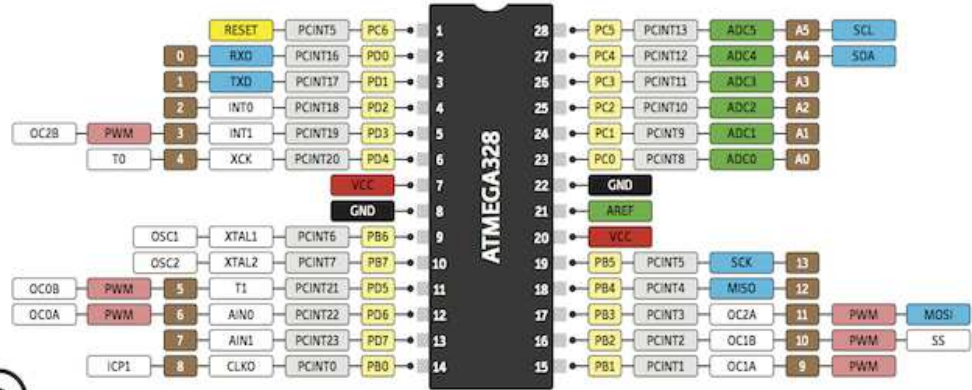
- The Arduino Uno prototyping board
- **Arduino Uno** is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
- Off-Boarding the ATMEGA328 requires changing the pin assignments
- For instance; Programming a command to Pin 13 on the Uno Prototype board must now be changed to Pin19 for the stand alone ATMEGA328. See the diagram below.



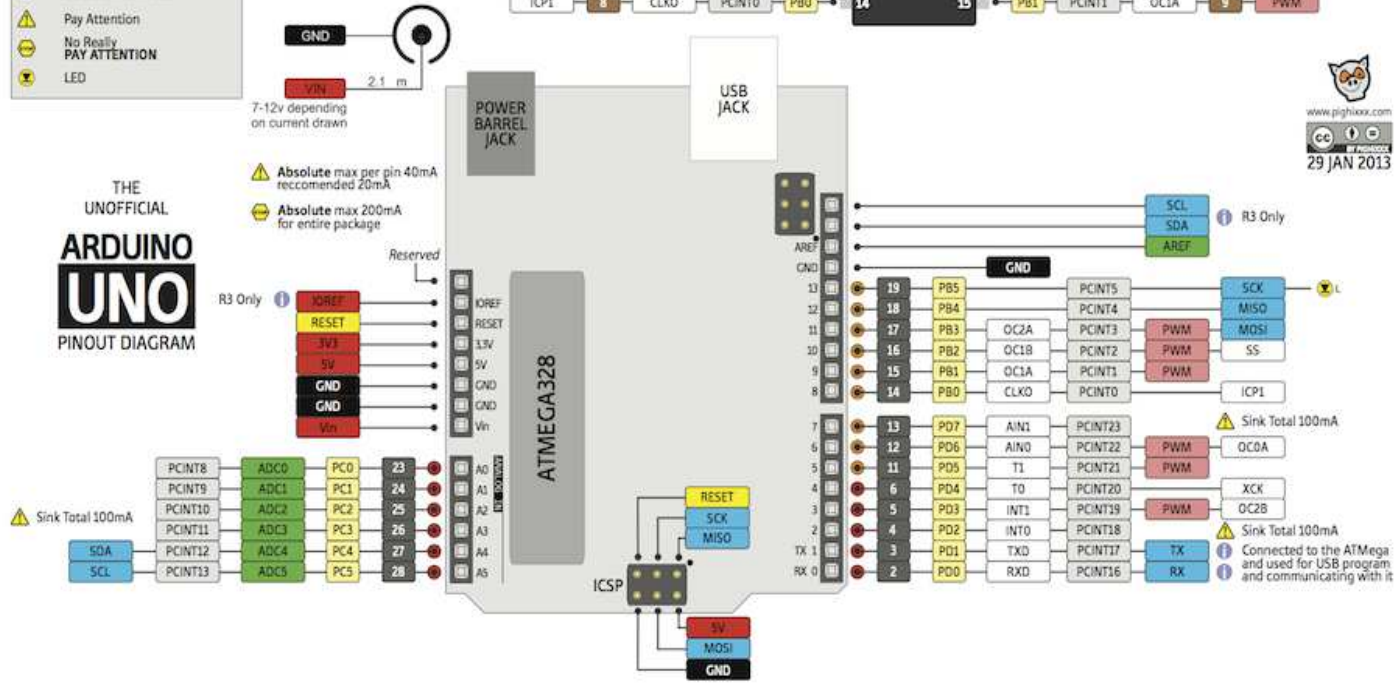
**LEGEND**

	<b>GND</b>
	<b>POWER</b>
	<b>CONTROL</b>
	<b>PHYSICAL PIN</b>
	<b>PORT PIN</b>
	<b>ATMEGA328 PIN FUNC</b>
	<b>DIGITAL PIN</b>
	<b>ANALOG-RELATED PIN</b>
	<b>PWM PIN</b>
	<b>SERIAL PIN</b>
	<b>ARDUINO PIN</b>

Source Total 150mA  
 Source Total 150mA  
 General Information  
 Pay Attention  
 No Really PAY ATTENTION  
 LED



THE UNOFFICIAL  
**ARDUINO UNO**  
PINOUT DIAGRAM



# MICROPROCESSOR PROGRAMMING

- Programming either the Uno board or the stand alone processor requires connecting a serial USB cable to the Uno USB port or, for the stand alone case, using a MAX232 chip to convert the RS232 voltages to 5 volts.

- I use an USB to RS232 converter cable which can be purchased online at;

<https://www.amazon.com/Serial-RS232-Converter-Laptop-Notebook/dp/B00404POL6>



# MICROPROCESSOR PROGRAMMING (CONT.)

- A simple programming example:
- Make an LED blink. Connect an LED to Pin 13 of the Arduino Uno Proto board with the appropriate series dropping resistor of 330 ohms.
- Note: The maximum current sourcing pin current is = 40 ma.
- An **integrated development environment (IDE)** is a software application that provides comprehensive facilities to computer programmers for software development.
- Download the Arduino IDE at <https://www.arduino.cc/en/Main/Software>

Code;  
// The setup function runs when you press reset or power the board  
//Code in the setup(), Intitalizes Pins as Input or Output.  
//Setup only runs Once on startup  
// The void keyword is used only in function declarations. It indicates that the function is expected to return no information to the function from which it was called.  
//The following code is for the UNO proto board, Pin 13 would be changed to pin 19 for the off boarded processor.

#### **void setup()**

```
{  
pinMode(Pin 13, OUTPUT); // initialize digital pin 13 as an output.  
}
```

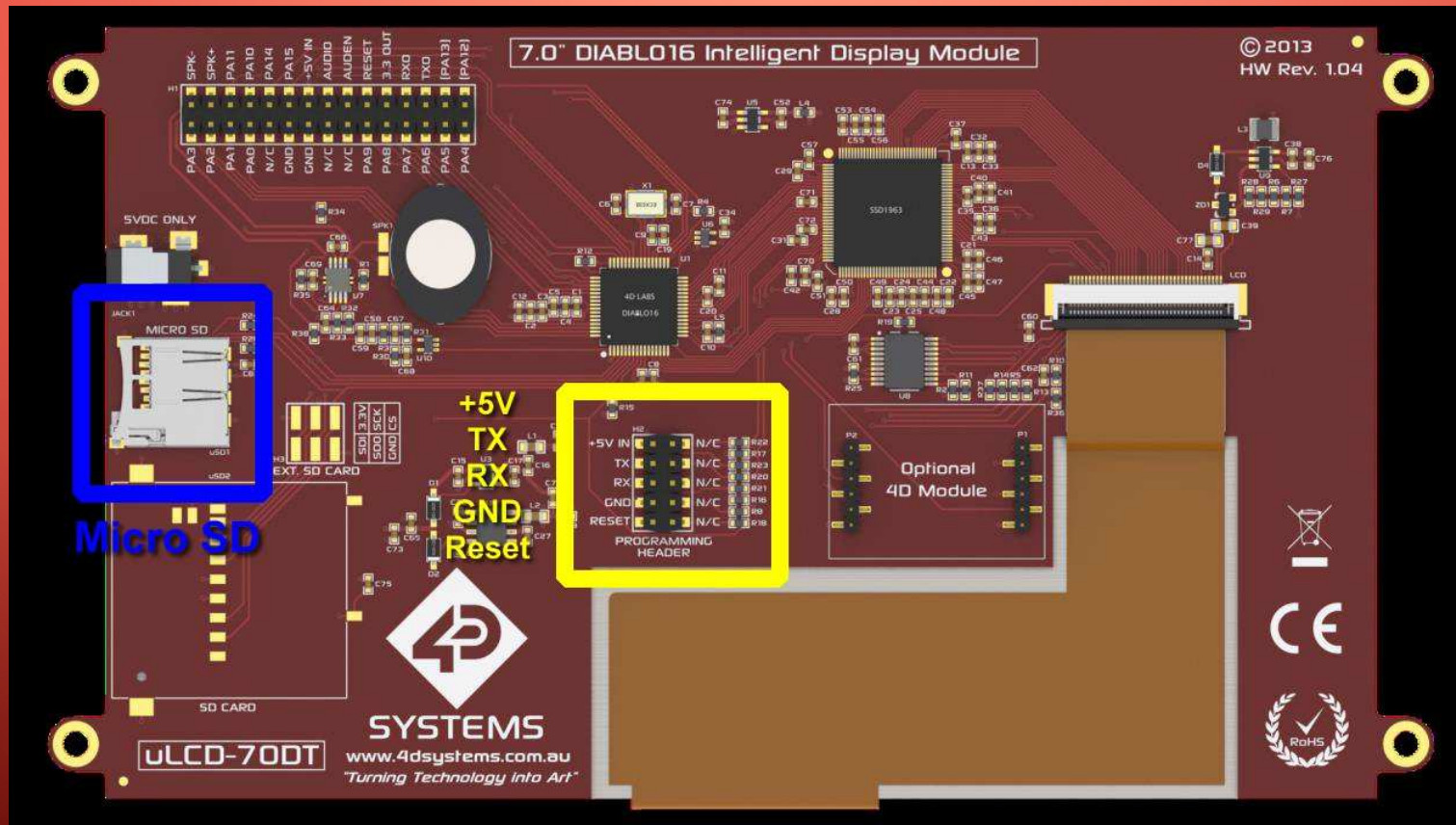
#### **void loop()** // The loop function runs over and over again forever

```
{  
digitalWrite(Pin 13, HIGH); // turn the LED on (HIGH is the voltage level)  
delay(500); // wait for a second  
digitalWrite(Pin 13, LOW); // turn the LED off by making the voltage LOW  
delay(500); // wait for a second  
}
```



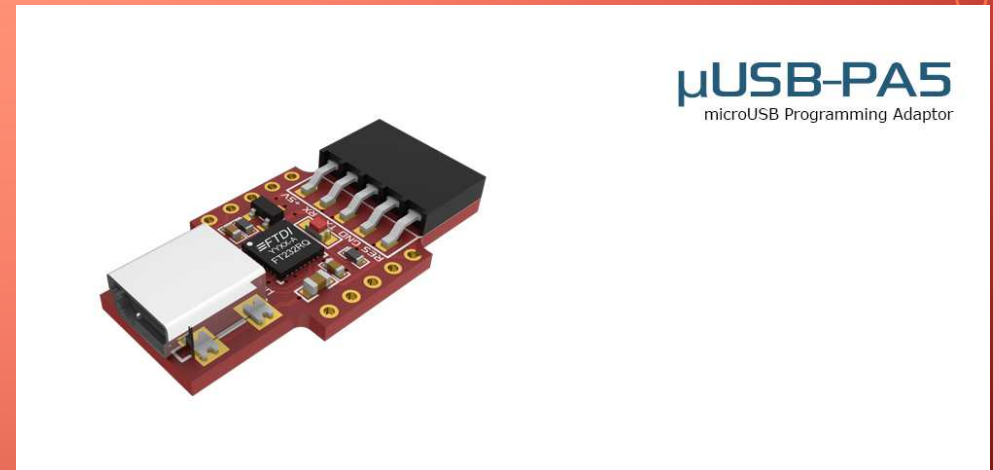
# PROGRAMMING THE 4D SYSTEMS LCD DISPLAY

Backside view



## PROGRAMMING THE LCD DISPLAY (CONT.)

- The microUSB Programming Adaptor (uUSB-PA5 and uUSB-PA5-II) is a USB to Serial-TTL UART bridge converter
- Insert the micro SD card into the PC USB port (use micro to mini USB adapter card if necessary)
- Connect uUSB-PA5 from +5V, TX, RX, GND, Reset connector to PC USB port.





# PROGRAMMING THE LCD DISPLAY

Boot up the IDE program

**WORKSHOP 4 IDE** Version 4.5.0.17

**Create a new 4D Systems Project**  
Start building a new Visi, Genie, Designer or Serial program.










**Create a new 4D Labs Project**  
Start building a new Visi, Genie, Designer or Serial program.  
Coming Soon.

**Create a new Project**  
Start building a new program using the same settings as you last used (Arduino Extended Graphics Gen4-111 CD-700CT-C18-AR)

**4D SYSTEMS** TURNING TECHNOLOGY INTO ART

**4D LABS** SEMICONDUCTORS

Recent (Click here to Clear)

 <b>Last Radio5-9-18.4DGenie</b>	5/9/2018 9:19 AM
C:\Users\Isnoo\Lee's Files\Radios\Radio 6 7-2-16\Genie\RX6 7_32_16.ImgData>Last Radio5-9-18.4DGenie	
 <b>Jeffrey Tic Tac Toe 7-4-18.4DGenie</b>	Notfound
C:\Users\Isnoo\Desktop\Jeffrey\Jeffrey Tic Tac Toe 7-4-18.4DGenie	
 <b>NoName1.4DGenie</b>	Notfound
C:\Users\Isnoo\Desktop\Jeffrey\NoName1.4DGenie	
 <b>Last Radio 4-30-18.4DGenie</b>	5/9/2018 9:17 AM
C:\Users\Isnoo\Lee's Files\Radios\Radio 6 7-2-16\Genie\RX6 7_32_16.ImgData>Last Radio 4-30-18.4DGenie	
 <b>Last Radio 4-28-18.4DGenie</b>	4/30/2018 5:31 PM
C:\Users\Isnoo\Lee's Files\Radios\Radio 6 7-2-16\Genie\RX6 7_32_16.ImgData>Last Radio 4-28-18.4DGenie	
 <b>Last Radio 2-17-18.4DGenie</b>	4/28/2018 9:29 AM
C:\Users\Isnoo\Lee's Files\Radios\Radio 6 7-2-16\Genie\RX6 7_32_16.ImgData>Last Radio 2-17-18.4DGenie	
 <b>Last Radio 2-13-18.4DGenie</b>	3/16/2018 6:51 PM
C:\Users\Isnoo\Lee's Files\Radios\Radio 6 7-2-16\Genie\RX6 7_32_16.ImgData>Last Radio 2-13-18.4DGenie	
 <b>2-12-18.4DGenie</b>	3/12/2018 10:52 PM
C:\Users\Isnoo\Lee's Files\Radios\Last Radio 10-30-17\Genie\2-12-18.4DGenie	
 <b>RX6 7_5_16.4XE</b>	7/8/2016 1:48 PM
C:\Users\Isnoo\Lee's Files\Radios\Radio 6 7-2-16\Genie (New 7 inch LCD)\RX6 7_5_16.4XE	

Scroll using mousewheel, or click and drag using left mousebutton

# PROGRAMMING THE LCD DISPLAY (CONT.)

Select the ViSi  
Genie Display  
Type and  
Orientation

Workshop 4 - Last Radio5-9-18(uLCD-70DT, LANDSCAPE)

File Home View Tools Comms Project

CHOOSE YOUR PRODUCT

All

	panel & Cover Lens Bezel and Adapter shield	
	<b>Gen4-uLCD-70D-AR</b> 7.0" Gen4 Intelligent Display Module with embedded DIABLO16 processor and Adapter shield	800x480
	<b>Gen4-uLCD-70D-CLB-AR</b> 7.0" Gen4 Intelligent Display Module with embedded DIABLO16 processor w/ Cover Lens Bezel and Adapter shield	800x480
	<b>Gen4-uLCD-70DT-AR</b> 7.0" Gen4 Intelligent Display Module with embedded DIABLO16 processor w/ Resistive Touch panel and Adapter shield	800x480
	<b>Gen4-uLCD-70DT-CLB-AR</b> 7.0" Gen4 Intelligent Display Module with embedded DIABLO16 processor w/ Capacitive Touch panel & Cover Lens Bezel and Adapter shield	800x480
	<b>Gen4-uLCD-24PT-AR</b> 2.4" Gen4 Intelligent Display Module with embedded PICASO processor w/ Resistive Touch panel and Adapter shield	240x320
	<b>Gen4-uLCD-28PT-AR</b> 2.8" Gen4 Intelligent Display Module with embedded PICASO processor w/ Resistive Touch panel and Adapter shield	240x320
	<b>Gen4-uLCD-32PT-AR</b> 3.2" Gen4 Intelligent Display Module with embedded PICASO processor w/ Resistive Touch panel and Adapter shield	240x320
	<b>uOLED-96-G2-AR</b> 0.96" Intelligent OLED module and Adapter shield	96x64

Select Display Type

Landscape (Click image to rotate)

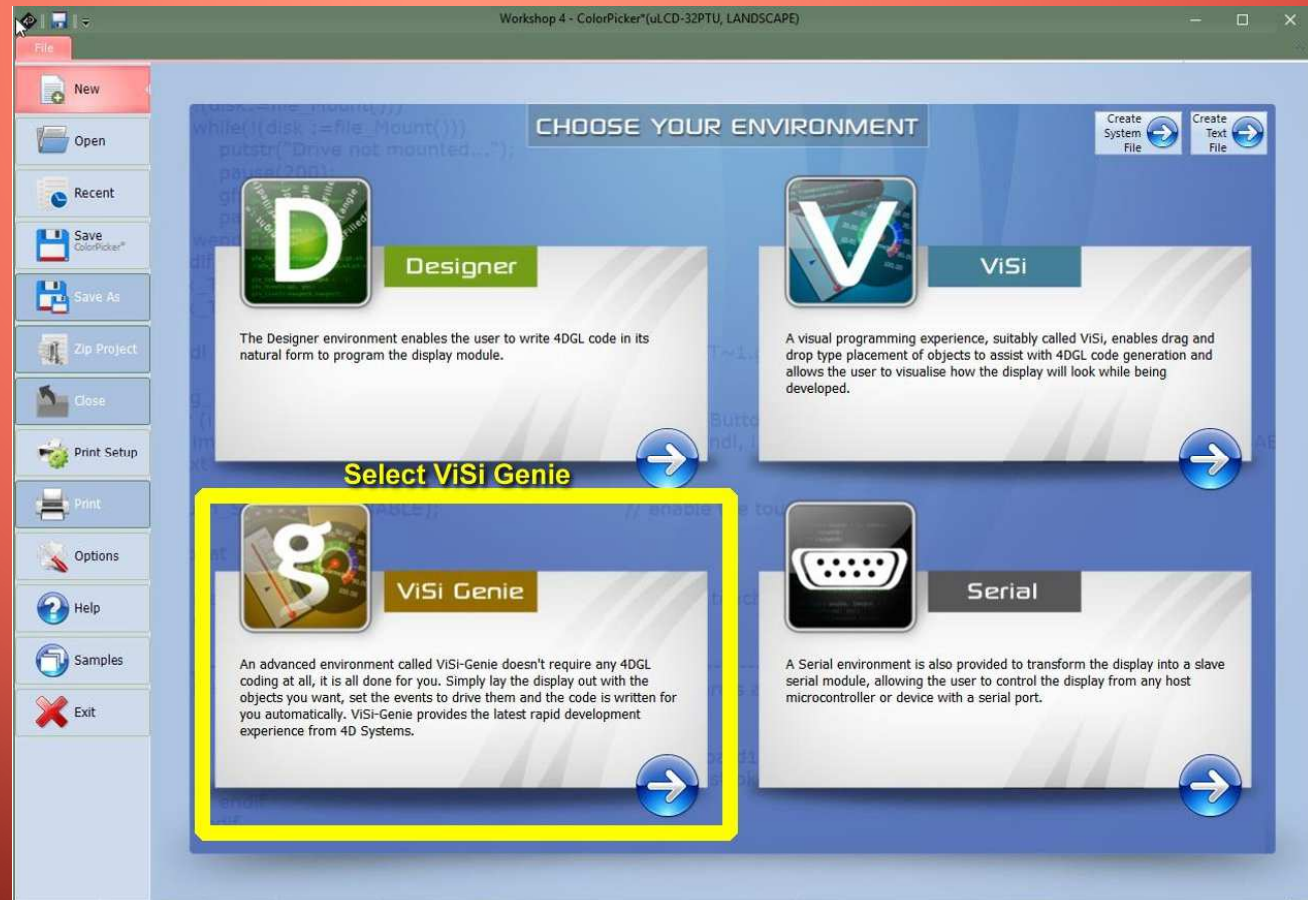
Next

Scroll using mousewheel, or click and drag using left mousebutton



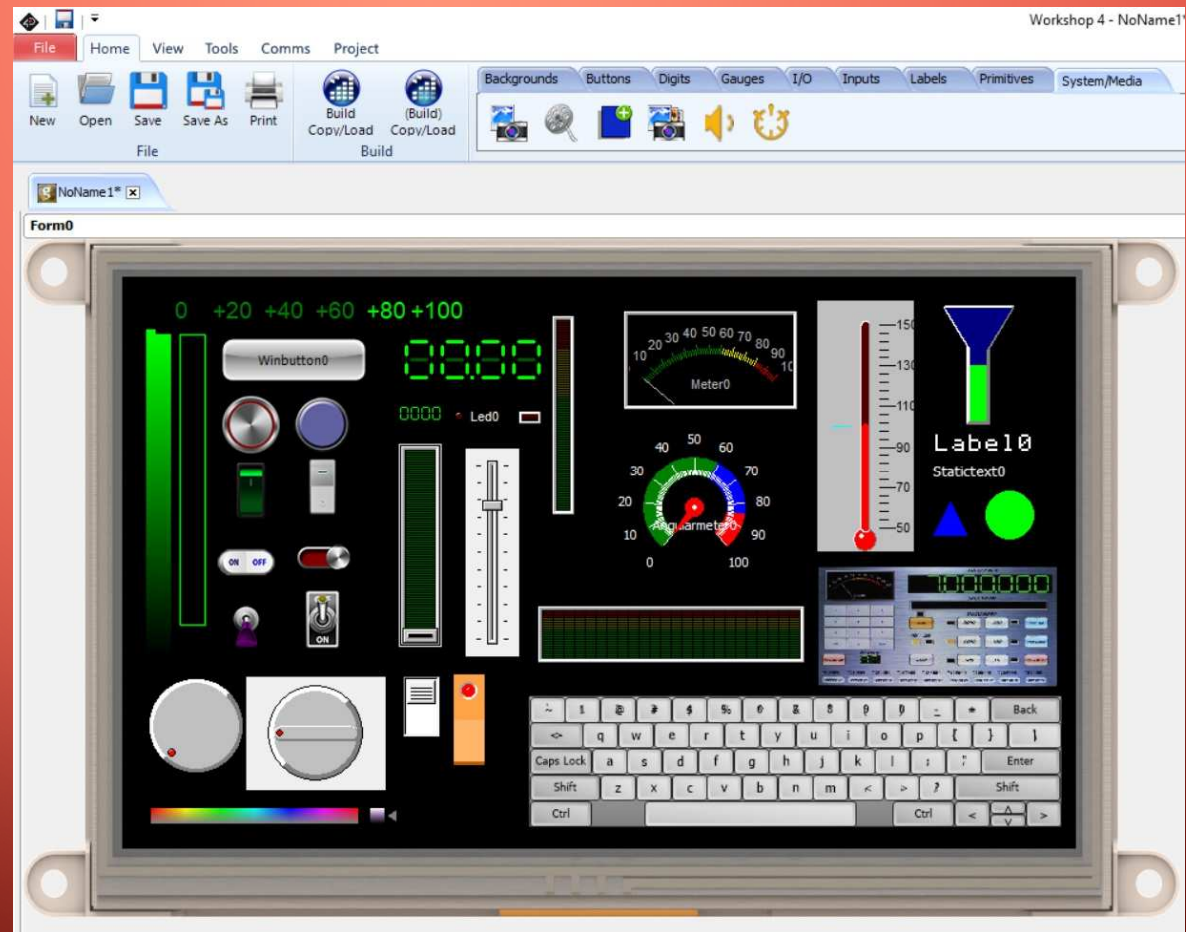
# PROGRAMMING THE LCD DISPLAY (CONT.)

Select Programming Environment;  
By simply laying the display out with the wanted objects and setting the events to drive them, the code is written automatically. ViSi-Genie provides the latest rapid development experience from 4D Labs



# PROGRAMMING THE LCD DISPLAY (CONT.)

Control Objects that can be used

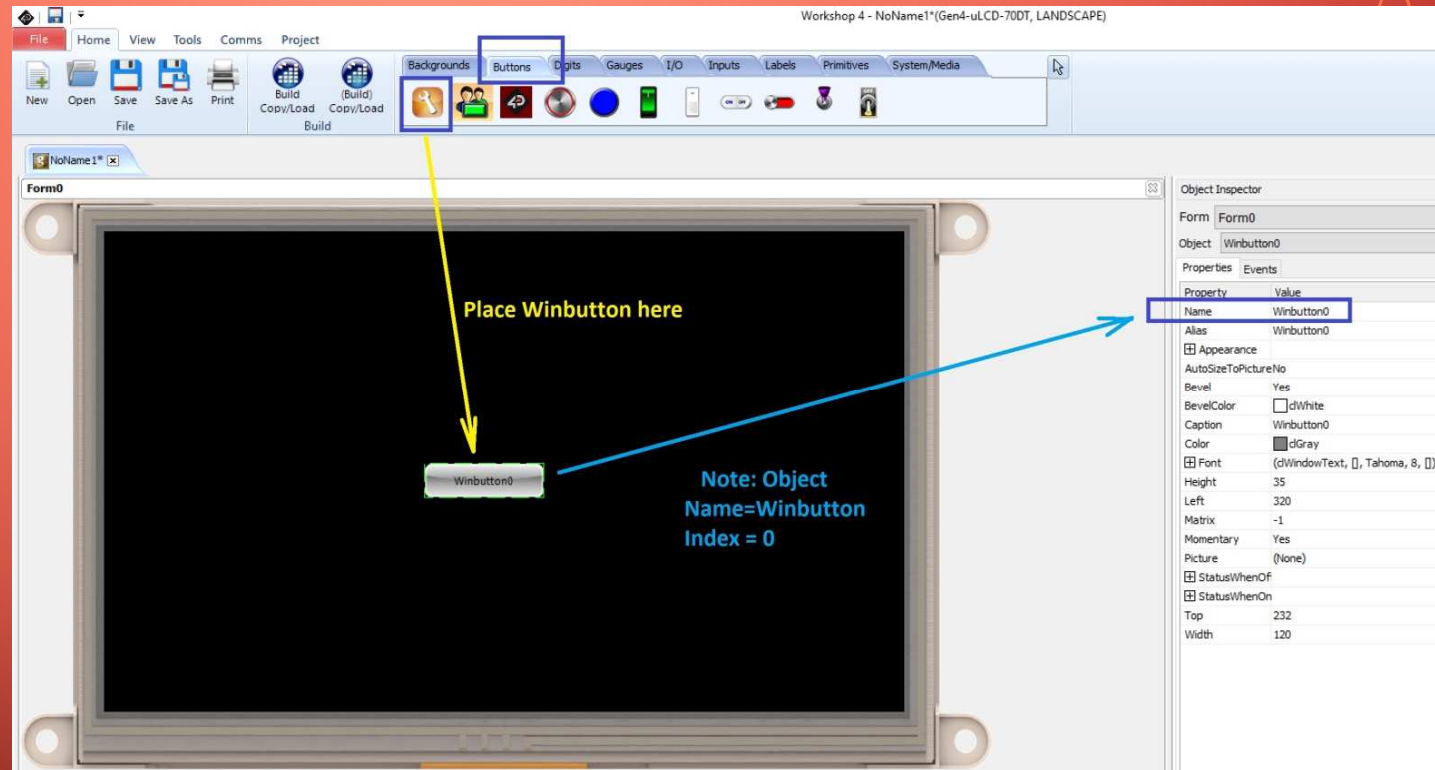




# PROGRAMMING THE LCD DISPLAY (CONT.)

Start the LCD layout by selecting and placing the “Objects” onto the workspace.

Example: In this example we will place a Button on the screen and note the Object Inspector Remember to Select the Comms Tab and Comm Port, it will turn Blue when connected. Name and Index number.



# PROGRAMMING THE LCD DISPLAY (CONT.)

Under Object Inspector on the right side of the screen, select the Events Tab and select “Report Message”.





# EVENT HANDLER EXAMPLE

The Arduino Code (in the Event Handler) will look for a Winbutton Event with an Index of 0 and execute the user code when this button is pressed.

The programming language used for Arduino is based on C language. The syntax is almost identical to that of C or C++.

```
Void Setup()
```

```
{
```

```
  Serial.begin(115200);
```

```
  genie.Begin(Serial);
```

```
  genie.AttachEventHandler(myGenieEventHandler);
```

```
  delay (2000);
```

```
  //let the display start up
```

```
  genie.WriteContrast(brightness);}
```

## EVENT HANDLER EXAMPLE (CONT.)

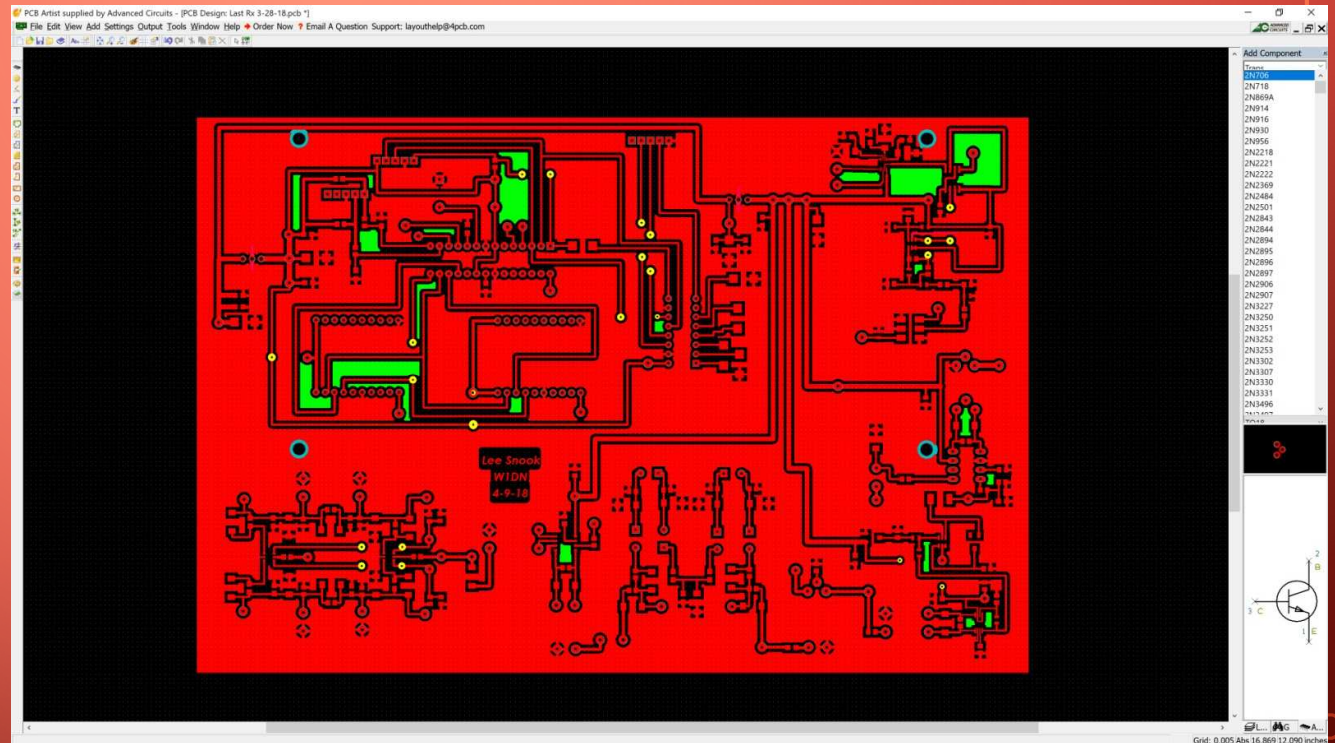
```
Void Loop()
{
  genie.DoEvents();
}
//-----Event Handler-----
void myGenieEventHandler(void)
{
  genieFrame Event;
  genie.DequeueEvent(&Event);

  if(Event.reportObject.cmd == GENIE_REPORT_EVENT)
  {
    if (Event.reportObject.object == GENIE_OBJ_WINBUTTON)
    {
      //-----If Winbutton Pressed, Do Something-----
      if (Event.reportObject.index == 0)           // If Button(0)
      {
        //Do Something
      }
    }
  }
}
```



# PCB FABRICATION PROCESS

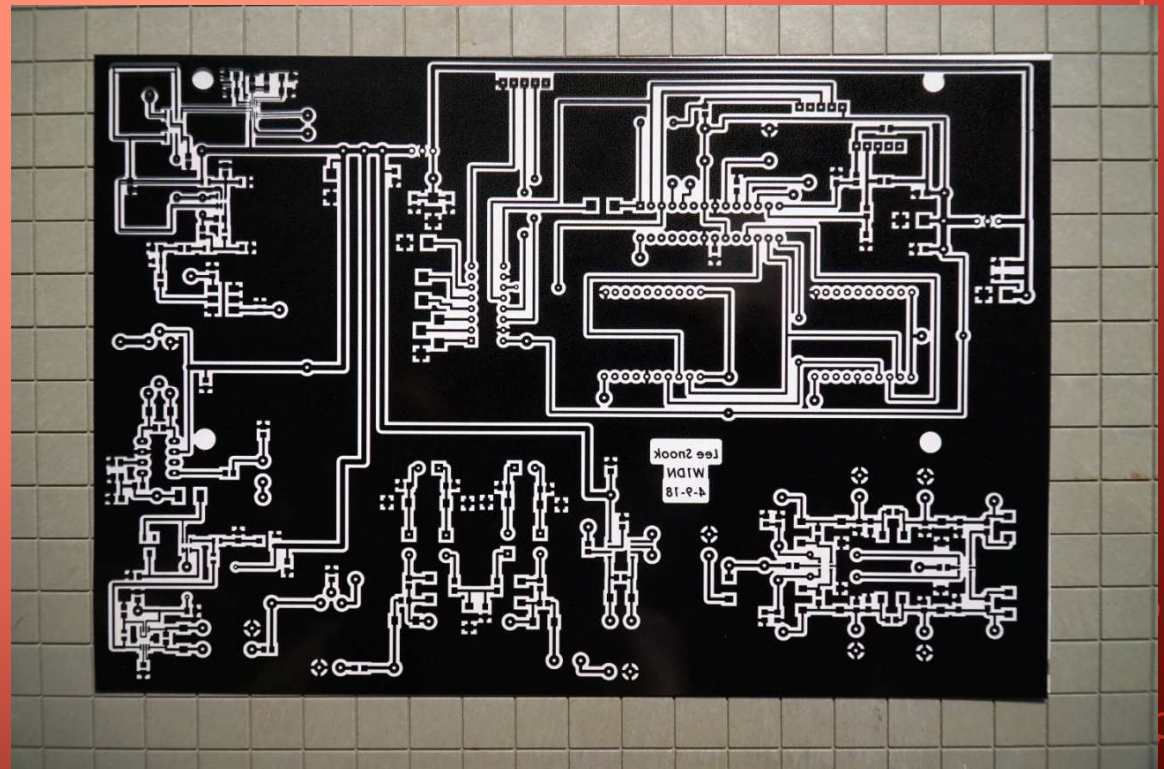
I use PCB Artist for Schematic Capture and Board Layout



The Receiver Layout in PCB Artist:

## PCB FABRICATION PROCESS (CONT.)

- Using an Ink Jet printer,
  - print the Top Copper side of the Layout
  - High Contrast Black
  - onto rough side of transparent film
- Remember, under printer settings, Flip the print under the printer setup so that the rough side is facing the photo sensitive side of the PCB board.





## PCB FABRICATION PROCESS (CONT.)

- **Remove the light protective film from the photo sensitive PCB board**
- **Insert the photo sensitive board under the transparent film and expose in a lightbox**



## PCB FABRICATION PROCESS (CONT.)

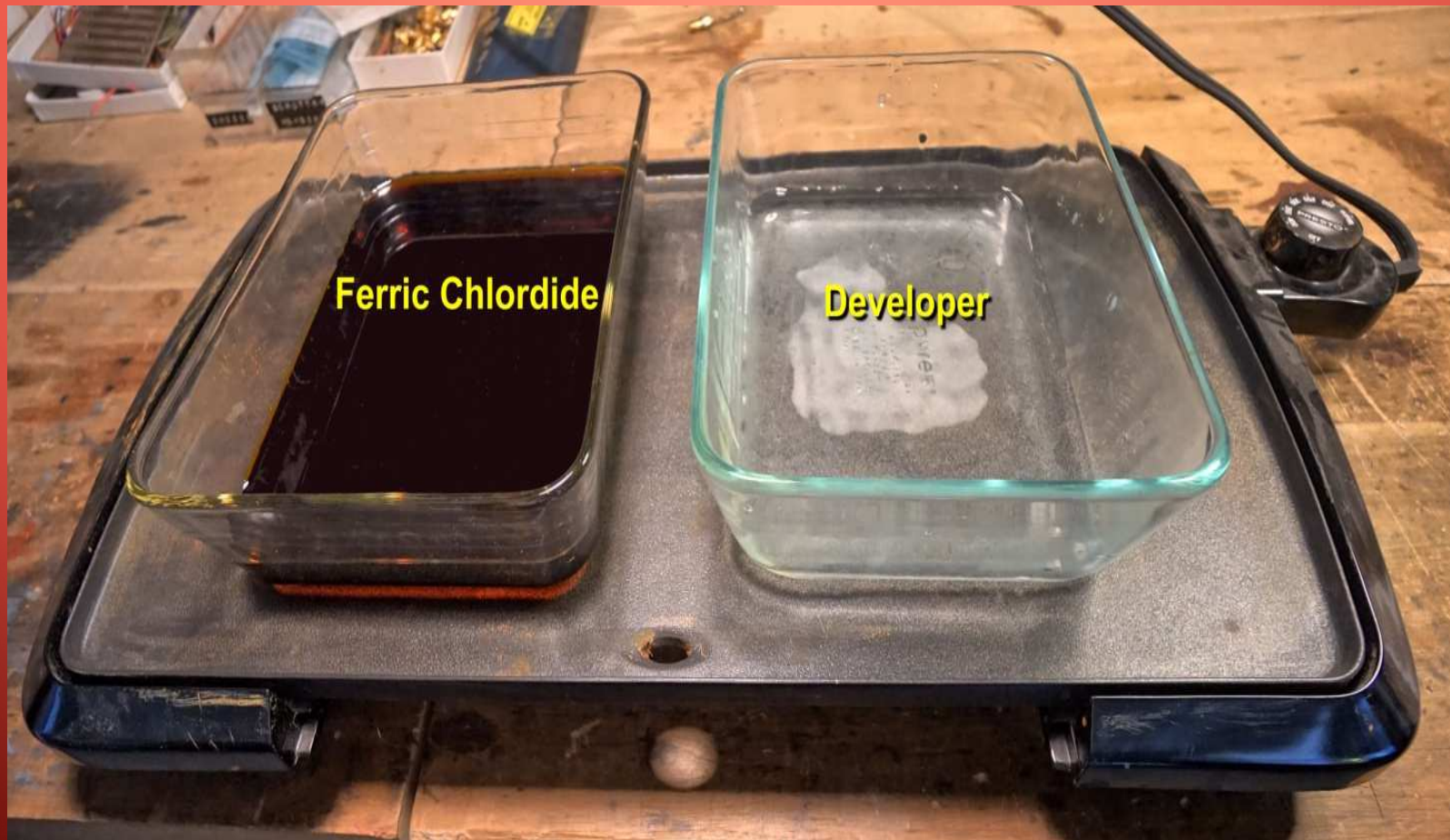
- **Heat Developer and Ferric Chloride to ~90 degrees F.**
- **Exposed PCB into glass tray with PCB Developer**
  - **DATAK Cat. No 12-404 Positive Type PCB Developer**
  - **Developer concentrate diluted 1:10 with water.**
- **Agitate until the Light Sensitive Film (Green color) is dissolved between traces.**
- **Wash the board**



## PCB FABRICATION PROCESS (CONT.)

- **Board into Ferric Chloride Etchant Solution bath**
  - **MG Chemicals #415-1L,945 ml bottles Ferric Chloride**
  - **Typically 15 minutes**
- **Periodically agitate**
- **Inspect and remove when the copper between traces has been dissolved**
- **Wash off Ferric Chloride with fresh water**


## PCB FABRICATION PROCESS (CONT.)





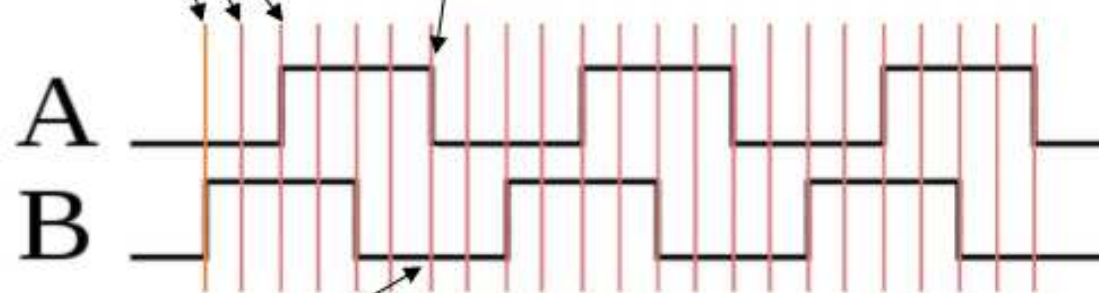
# QUADRATURE ROTARY ENCODER

SPARKFUN ELECTRONICS; PN COM-11102

Rotation (Clockwise) 

The routine samples Channel A every loop cycle every 15 ms.

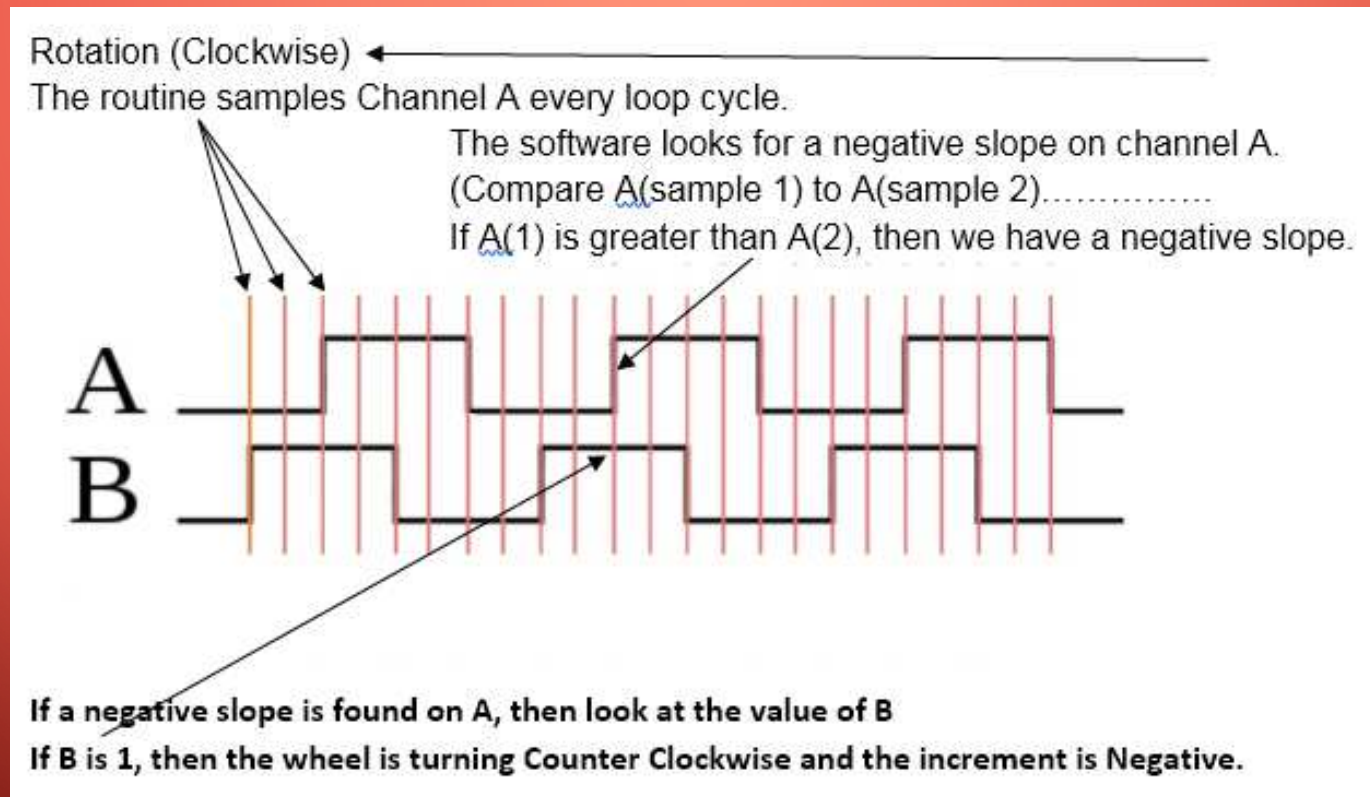
The software looks for a negative slope on channel A.  
(Compare A(sample 1) to A(sample 2).....  
If A(1) is greater than A(2), then we have a negative slope.



If a negative slope is found on A, then look at the value of B

If B is 0, then the wheel is turning Clockwise and the increment is positive.

# QUADRATURE ROTARY ENCODER (CONT.)



An RC time network was added to Ports A and B for Debounce.  $R=10k$  and  $C=.01\mu f$ .



# Quadrature Rotary Encoder (cont.)

In the following code;

A1=Sample 1 of Port A

A2=Sample 2 of Port A

B=Sample Port B

```
//-----Rotarty Encoder-----
```

```
A1=digitalRead(Encoder_Pin1);
```

```
delay(15); //Delay next sample by
```

```
15 ms
```

```
A2=digitalRead (Encoder_Pin1);
```

```
B=digitalRead(Encoder_Pin2);
```

```

if (A1>A2) //A1(sample 1) is greater
than A2(sample 2)
{
  if (B==HIGH)
  {
    f_rx=f_rx+increment[inc_pointer]; //Increase frequency
by increment
    freq_low=(f_rx%10000);
    freq_high=(f_rx-freq_low)/10000;
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 2, freq_low);
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 3, freq_high);
    load1();
  }

  else //A(sample 1) is less than
A(sample 2)
  {
    f_rx=f_rx-increment[inc_pointer]; //Decrease frequency
by increment
    freq_low=(f_rx%10000);
    freq_high=(f_rx-freq_low)/10000;
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 2, freq_low);
    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 3, freq_high);
    load1();
  }
}

```

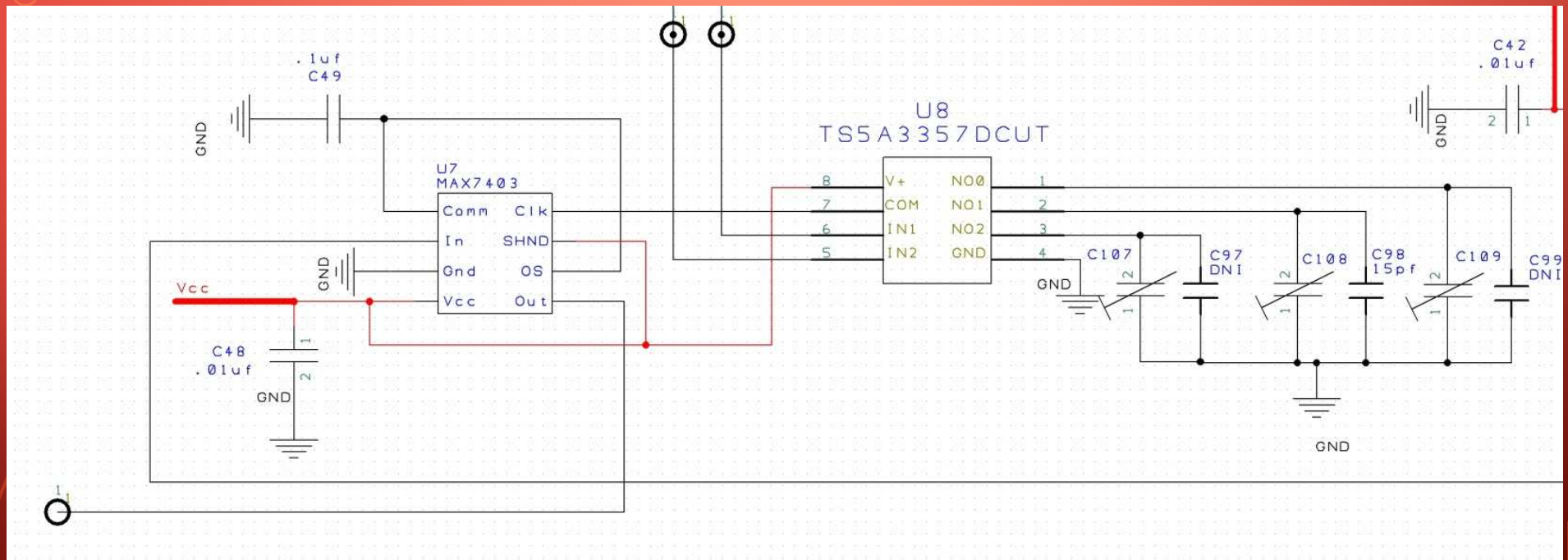
**More work is required, sometimes the encoder will skip a count when turning the wheel too fast.**



# VARIABLE BANDPASS AUDIO FILTER

- MAX7403 is a 8th-order, lowpass, elliptic, switched-capacitor filters
- These devices draw 2mA of supply current and allow corner frequencies from 1Hz to 10kHz
- Two clocking options are available
  - self-clocking through the use of an external capacitor
  - external clocking for tighter cutoff-frequency control
- Cutoff Frequency=Clock/100.  $F_{osc}(kHz) = (38 * 10000) / C_{osc}$ ;  $C_{osc}$  is in pf.
- For my application, I used external capacitors (self clocking) switched with a solid state single pole three throw switch (TS5A3357)

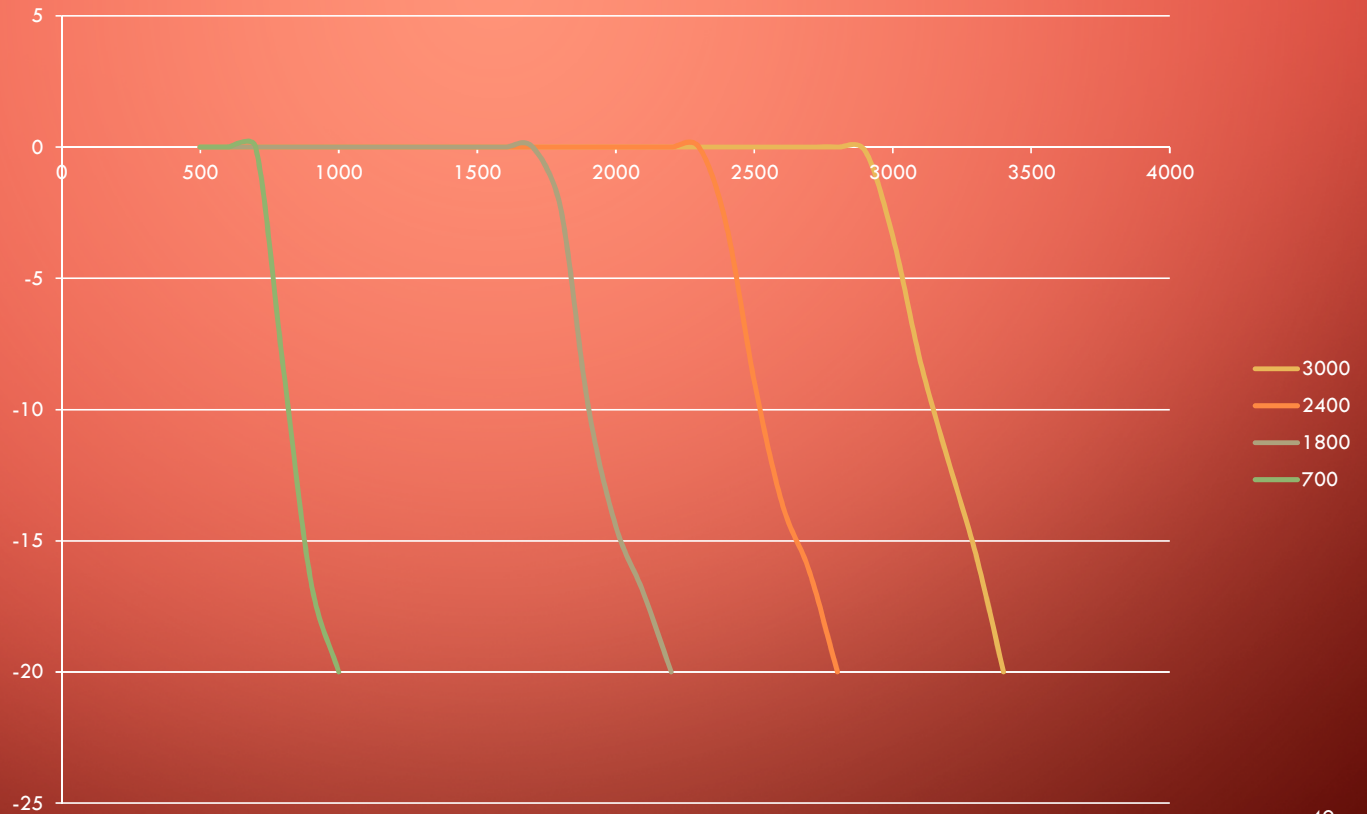
# VARIABLE BANDPASS AUDIO FILTER (CONT.)





# VARIABLE BANDPASS AUDIO FILTER (CONT.)

- Audio Lowpass Filter Frequency Response



# DDS (DIRECT DIGITAL SYNTHESIZERS (AD9850))

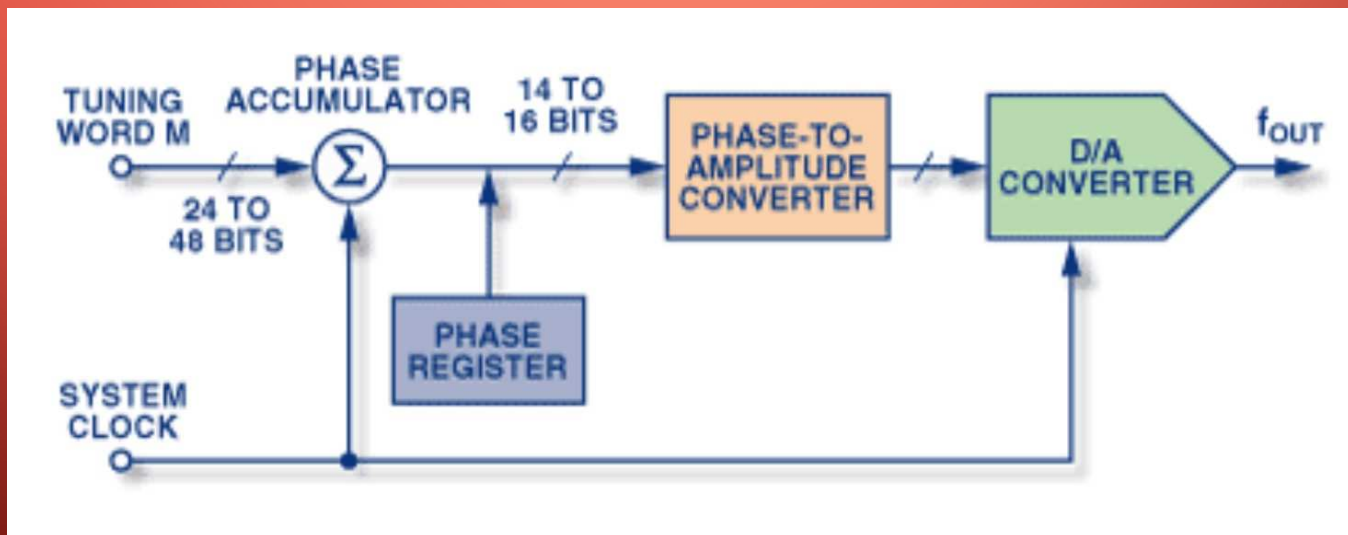
## DDS TUTORIAL

- Traditional designs of high bandwidth frequency synthesizers employ the use of a phase-locked-loop (PLL).
- A direct digital synthesizer (DDS) provides many significant advantages over the PLL approaches
  - Fast settling time
  - Sub-Hertz frequency resolution
  - Continuous-phase switching response
  - Low phase noise



## DIRECT DIGITAL SYNTHESIZERS (CONT.)

- A DDS produces a sine wave at a given frequency
- The frequency depends on two variables
  - *Reference-clock* frequency
  - Binary number programmed into the frequency register (*tuning word*)



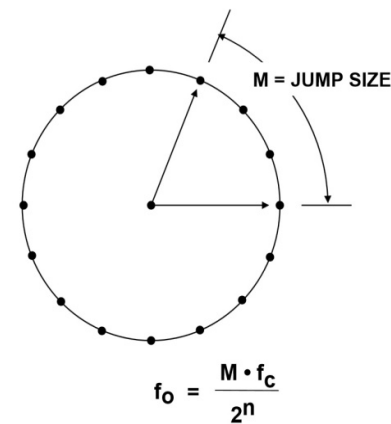
## DIRECT DIGITAL SYNTHESIZERS (CONT.)

- Binary number in frequency register provides main input to phase accumulator
- Sine look-up table used
- Phase accumulator computes phase (angle) address for look-up table
  - Outputs the digital value of amplitude
  - Corresponding to the sine of that phase angle—to DAC
- DAC converts number to a corresponding value of analog voltage or current



# DIRECT DIGITAL SYNTHESIZERS (CONT.)

- To generate a fixed-frequency sine wave, constant value (the phase increment – from binary number added to the phase accumulator with each clock cycle.
  - If the phase increment is large, the phase accumulator will step quickly through the sine look-up table and thus generate a high frequency sine wave.
  - If the phase increment is small, the phase accumulator will take many more steps, accordingly generating a slower waveform



n	Number of Points = $2^n$
8	256
12	4,096
16	65,536
20	1,048,576
24	16,777,216
28	268,435,456
32	4,294,967,296
48	281,474,976,710,656

Figure 3: Digital Phase Wheel

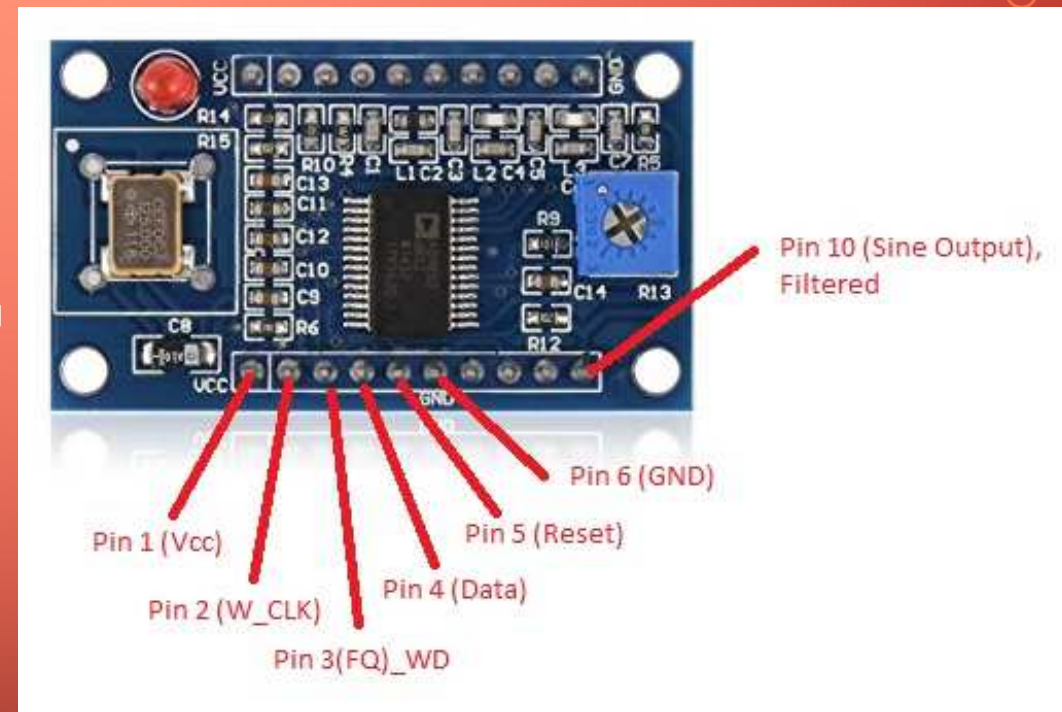
## DIRECT DIGITAL SYNTHESIZERS (CONT.)

- Consider the case for  $n = 32$ , and  $M = 1$ . The phase accumulator steps through each of  $2^{32}$  possible outputs before it overflows and restarts. The corresponding output sinewave frequency is equal to the input clock frequency divided by  $2^{32}$ . If  $M=2$ , then the phase accumulator register "rolls over" twice as fast, and the output frequency is doubled. This can be generalized as follows.
- For an  $n$ -bit phase accumulator ( $n$  generally ranges from 24 to 32 in most DDS systems), there are  $2^n$  possible phase points. The digital word in the delta phase register,  $M$ , represents the amount the phase accumulator is incremented each clock cycle. If  $f_c$  is the clock frequency, then the frequency of the output sinewave is equal to:
  - $f_{OUT} = (\Delta Phase \times CLKIN) / 2^{32}$
  - $\Delta Phase = (f_{OUT} \times 2^{32}) / CLKIN$



# DIRECT DIGITAL SYNTHESIZERS

- AD9850 is a highly integrated device that uses advanced DDS technology coupled with an internal high speed, high performance D/A converter and comparator to form a complete, digitally programmable frequency synthesizer and clock generator function.
- Premounted on a small PCB online from either Amazon or Digikey.



# DIRECT DIGITAL SYNTHESIZER (AD9850)

- The frequency tuning, control, and phase modulation words are loaded into the AD9850 via
  - Parallel byte or
  - Serial loading format (I used the serial loading format)
- In serial load mode, subsequent rising edges of W\_CLK shift the 1-bit data on Pin 25 (D7) through the 40 bits of programming information.
- After 40 bits are shifted through, an FQ\_UD pulse is required to update the output frequency (or phase).



The Arduino Code for loading the DDS synthesizer:

```
//-----DDS Loader-----  
long function_delta_phase1()  
{  
    freq_dds1=f_rx+if_center-offset;  
  
    //From the equation above  $\Delta Phase = (f_{OUT} \times 2^{32}) / CLKIN$   
    long delta_phase1 =(((freq_dds1)*pow(2,32))/clock);  
  
    return (delta_phase1);  
  
void load1()  
{  
    long delta_phase1=function_delta_phase1();  
    digitalWrite (LOAD1_Pin, LOW);  
    shiftOut(DATA_Pin, CLOCK_Pin, LSBFIRST, delta_phase1);  
    shiftOut(DATA_Pin, CLOCK_Pin, LSBFIRST, delta_phase1 >>  
8);  
    shiftOut(DATA_Pin, CLOCK_Pin, LSBFIRST, delta_phase1 >>  
16);  
    shiftOut(DATA_Pin, CLOCK_Pin, LSBFIRST, delta_phase1 >>  
24);  
    shiftOut(DATA_Pin, CLOCK_Pin, LSBFIRST, 0x0);  
    digitalWrite (LOAD1_Pin, HIGH);  
}
```