Perfect PSK31 Signals, Automatically.

Building a digital field strength meter for your PSK31 station

by

George Rothbart, KF6VSG

Presented at the

Pacificon 2003 Conference

October 17-19, 2003 – San Ramon CA

No need to take notes! This entire presentation is available at www.ssiserver.com/info/pskmeter

What we will cover this hour…

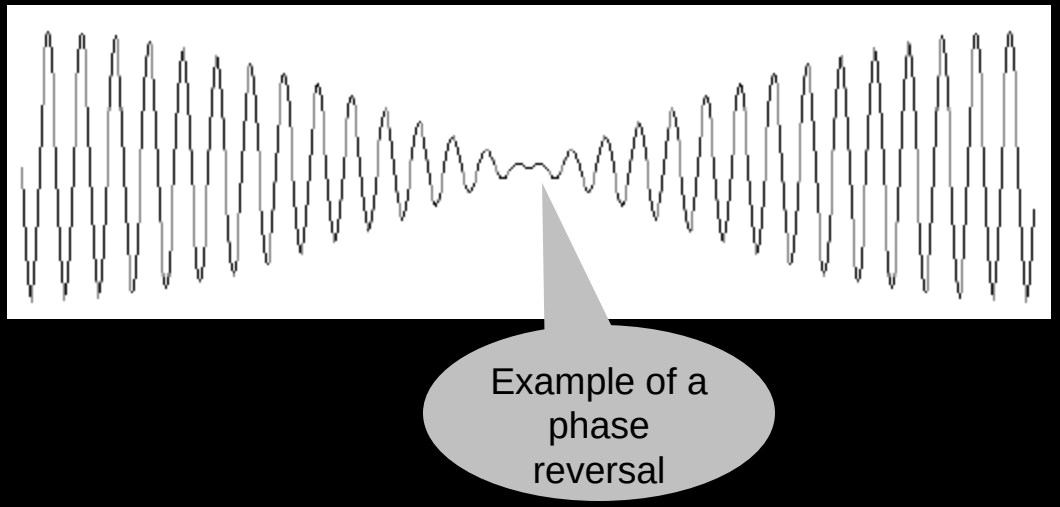
- PSK31 signalling (how it works)
- A PSK31 station and its components
- Tuning up in PSK31
- Over modulation: the audio level problem
- Building an RF monitor to solve the problem
- Circuit diagram and hardware details
- Firmware and Software routines
- Performance
- Where we go from here

Digital Modes (character- based data):

- ✓CW
- ✓RTTY
- ✓Packet
- ✓APRS
- ✓SSTV
- ✓***PSK-31 (and now PSK-63, too)***

Phase shift keying—the basics

Zero bits are signaled by 180^{o} phase reversals:



To avoid harmonics, the amplitude passes through zero at the instant of the reversal of phase. There is no need to change the amplitude when we don't change phase. Thus, we might see a signal amplitude like this:

PSK31 signaling

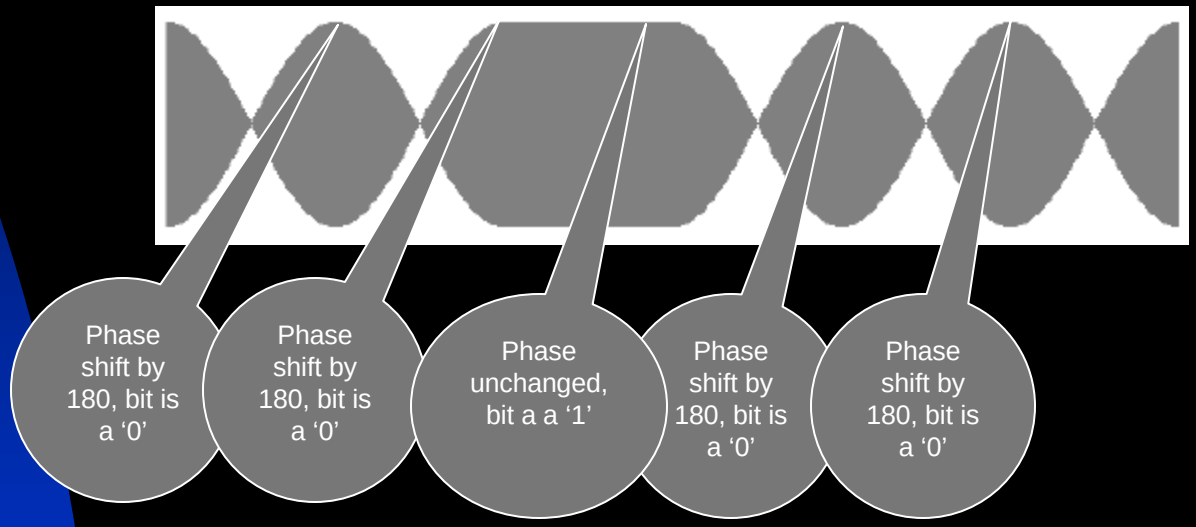
These phase changes occur every 32 milliseconds (31.25 Hz). Why this rate? Because:

- It can be easily derived from the 8KHz signal generated by your sound card (8000 Hz divided by 256 = 31.25 Hz), and
- 30 or so bits per second translates to about 50 wpm, the fastest you can type.

A zero (space) is defined as a 180 degree phase reversal from the prior sample,

A one (mark) is defined as no phase change from the prior sample.

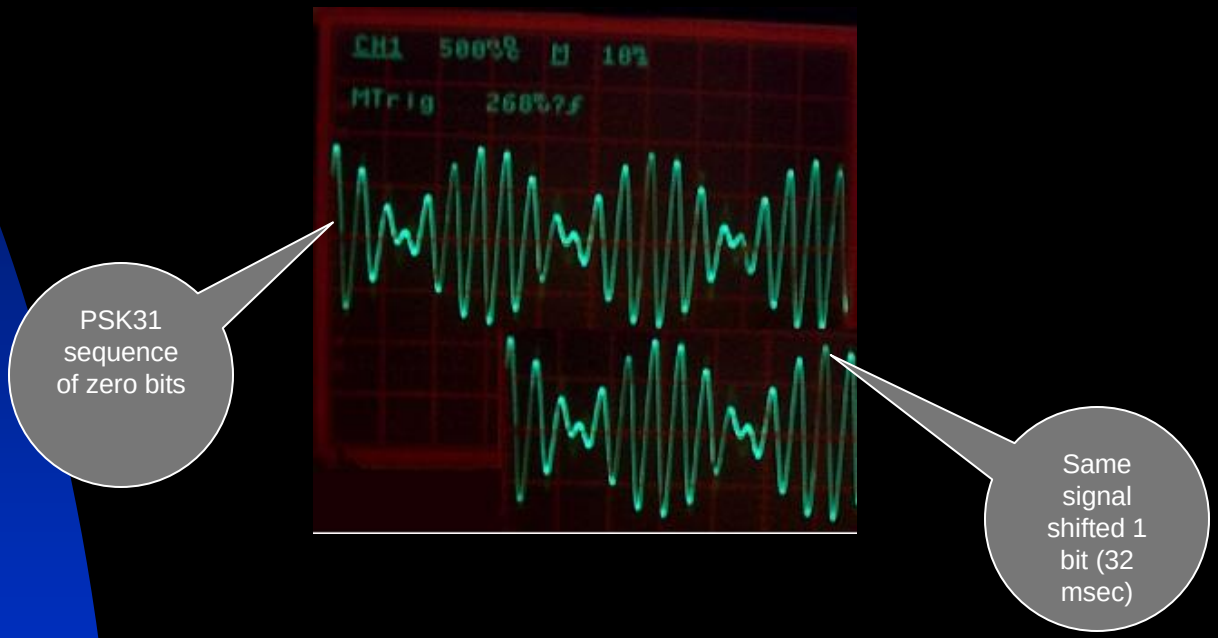
PSK31 signaling



Therefore, this signal is detected as 00100, which we will soon see is the representation of the character “SPACE”.

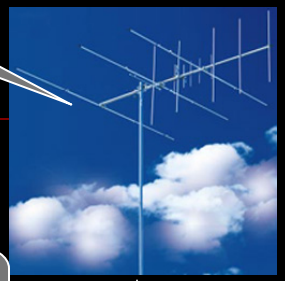
PSK31 signaling

Let’s look at PSK31 in the real world on an oscilloscope:



Notice that when we shift the signal one bit width (32 msec), and compare, the phase shift of 180 degrees is clear. This also gives us a clue for how PSK-31 is detected and converted back to the digital world.

The PSK-31 Station



Computer

Speaker Out

RF Out

a

Sound Card

b*

COM Port

Mic Input*

HF Transceiver

PSK-31 Application

DTR or RTS of, COMn*

a=Line In
b=Line Out

*=not needed to monitor

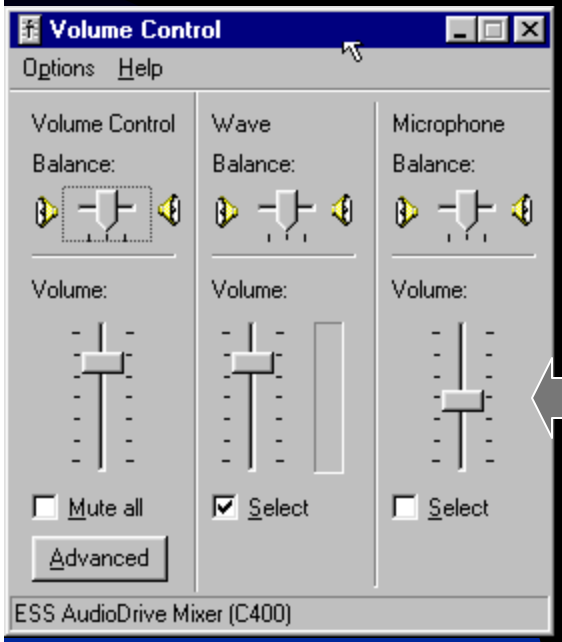
Tuning up in PSK-31

- Improperly tuned PSK-31 can result in
  - Undermodulation—your signal will be too weak to be received or copied
  - Overmodulation—your signal will have high Intermodulation Distortion (IMD) and splatter, and possibly will have poor copy also.

Tuning up in PSK-31

- Tuning up involves setting
  - Transceiver RF frequency
  - Transceiver microphone gain
  - Transceiver power setting
  - Turning off the speech processor
  - Turning off AGC
  - *Sound card line out audio level*

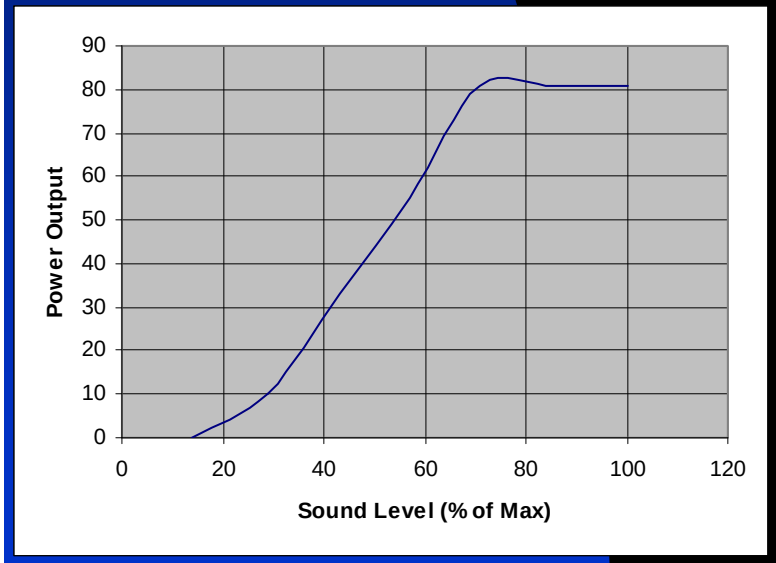
Setting Audio Levels



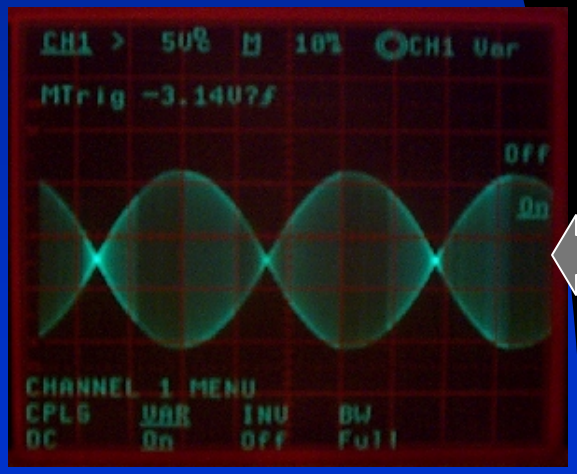
- It's all in the Window's "mixer."
- At the mixer's main display…
- De-select everything but "Wave" (that's your line out).
- Set the Wave volume to maximum.
- Now you can adjust the sound level by the Volume Control slider.

Setting the audio level

- Click “TX” button of your PSK-31 software, so that you are sending an idle PSK signal to the transceiver.
- Starting with the maximum sound level, observe the transmitter’s power output and back off the mixer’s audio level until the power level is at about 50%.
- You should observe an S-shaped curve like the one I measured at 1500 Hz.
- The proper audio level in this case would be about 50% of maximum.



Setting the audio level



- Avoid the temptation to crank up the audio level to get more power!
- You will only produce horrendous splatter, have very poor IMD, and be less copiable!
- Properly tuned, you will have a low IMD figure (-25 to –30 dB), and a clean spectrum with maximum copy—a signal you can be proud of.

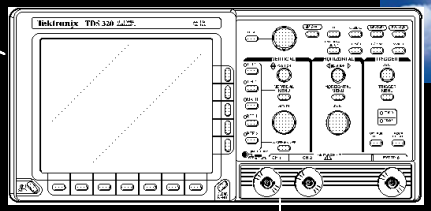
Setting the Audio Level is *not* a one-time Adjustment!

- The ideal audio level is a function of audio frequency. What works at 1500 Hz will not be a good setting at 300 Hz or 2500 Hz.
- Your rig's audio frequency response is not flat over the audio range but peaks in the mid-range from 1000-2000 Hz.
- Audio levels set by your Windows mixer will change as you use other applications.

The Oscilloscope Solution

Antenna

Scope



Speaker Out

Computer

RF Out

Feed line T'd to scope input

a

Sound Card

b*

COM Port

Mic Input*

HF Transceiver

PSK-31 Application

DTR or RTS of, COMn*

a=Line In
b=Line Out

*=not needed to monitor

What's the downside?

- Poor use of a costly resource
- Bulky, power consumptive (not a good solution for a portable PSK-31 station)
- No place to get a good trigger signal
- Still a manual operation (you fiddle with the volume control slider).

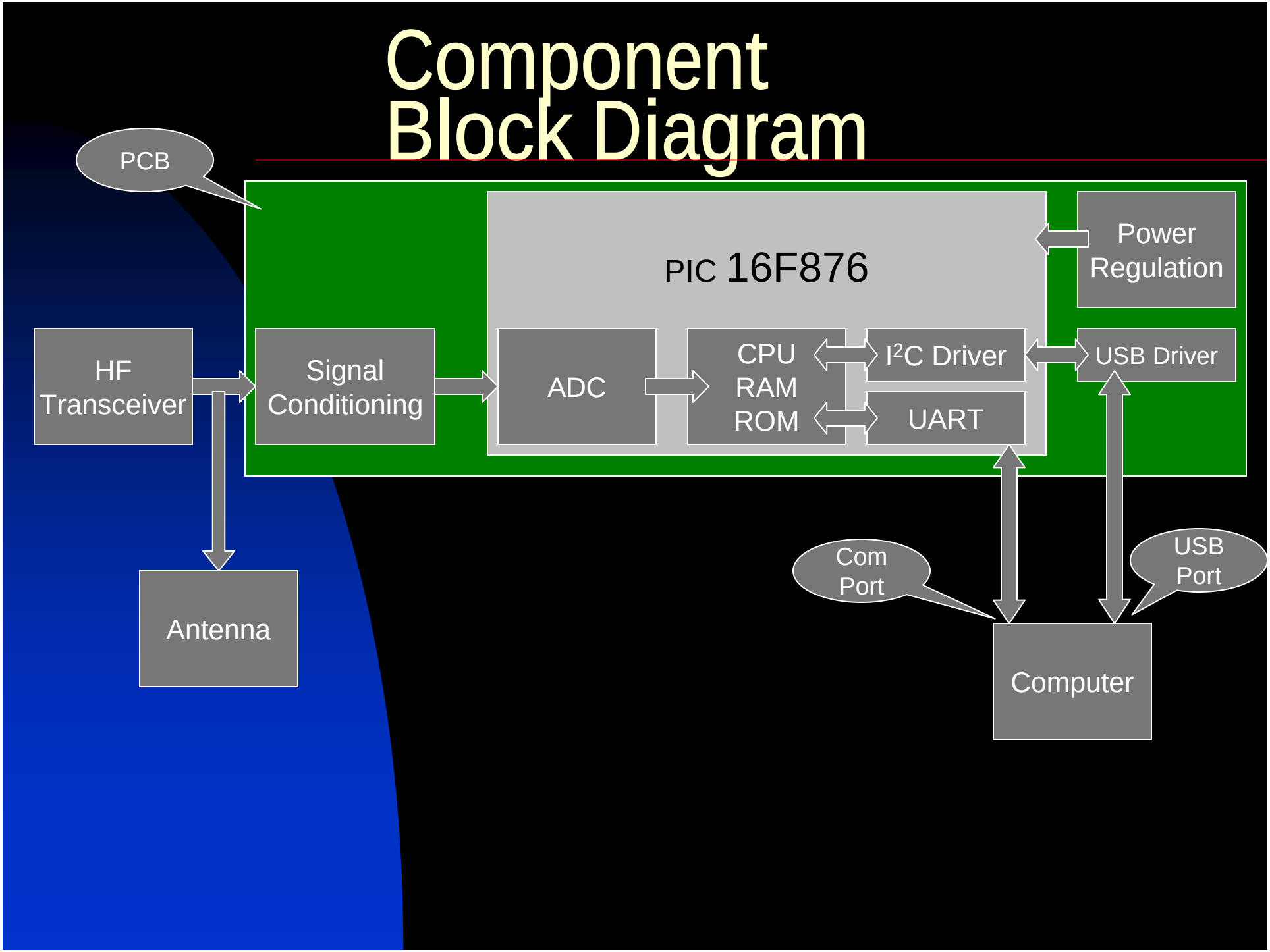
An alternate solution: let your computer be your scope

- A computer with display is guaranteed to be available
- Automatic graphical interface (if running a Windows OS)
- Only need to add an Analog to Digital Converter (ADC)
- Can do more than monitor RF—can dynamically set your audio for perfect output!

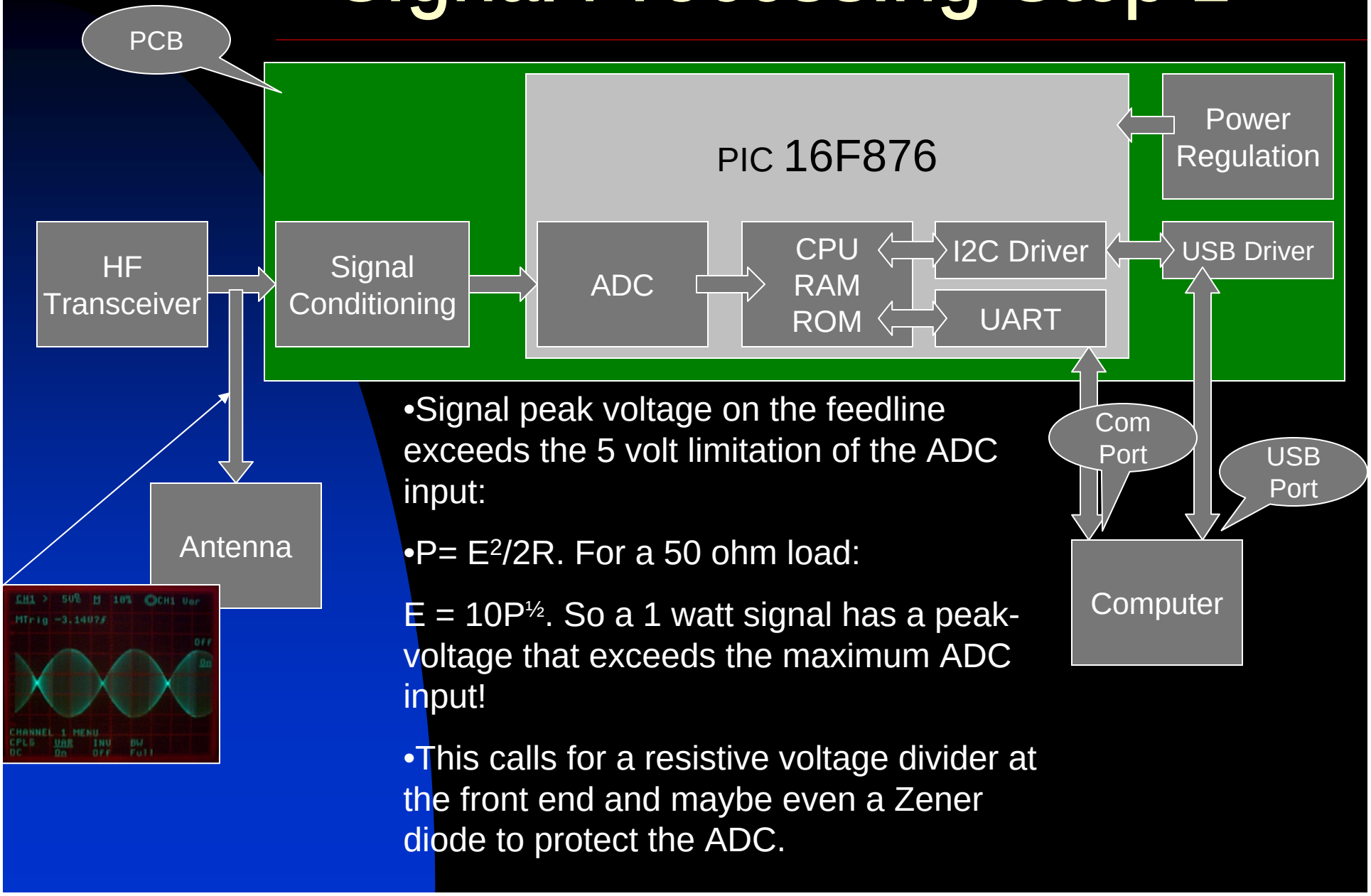
Adding an ADC

- Let's use a PIC microcontroller as an inexpensive way to add an 8-bit ADC to your computer.
- The PIC 16F876 comes with a built-in UART that can communicate with your computer's COM port
- Has a built in I^2C (2 wire) bus interface, so you can
  - Add more memory
  - Interface to a USB chip that can communicate with your computer's USB port

Component Block Diagram



Signal Processing-Step 1



- Signal peak voltage on the feedline exceeds the 5 volt limitation of the ADC input:
- $P = E^2/2R$. For a 50 ohm load:

$E = 10P^{1/2}$. So a 1 watt signal has a peak-voltage that exceeds the maximum ADC input!

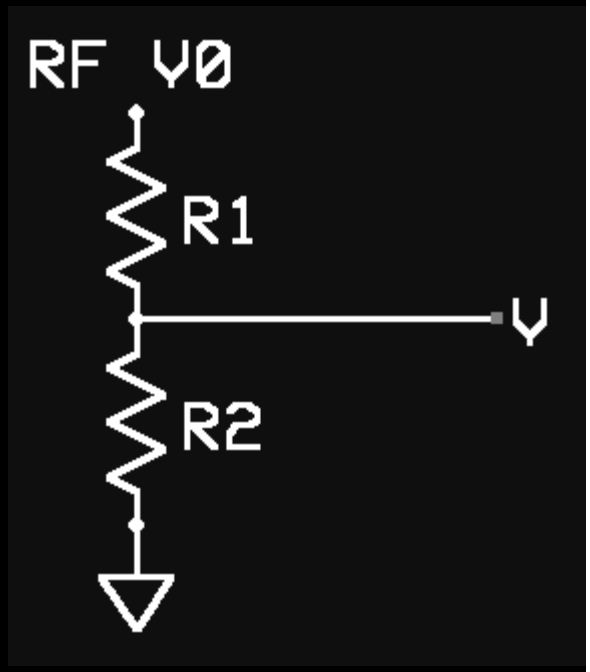
- This calls for a resistive voltage divider at the front end and maybe even a Zener diode to protect the ADC.

Signal Processing-Step 1

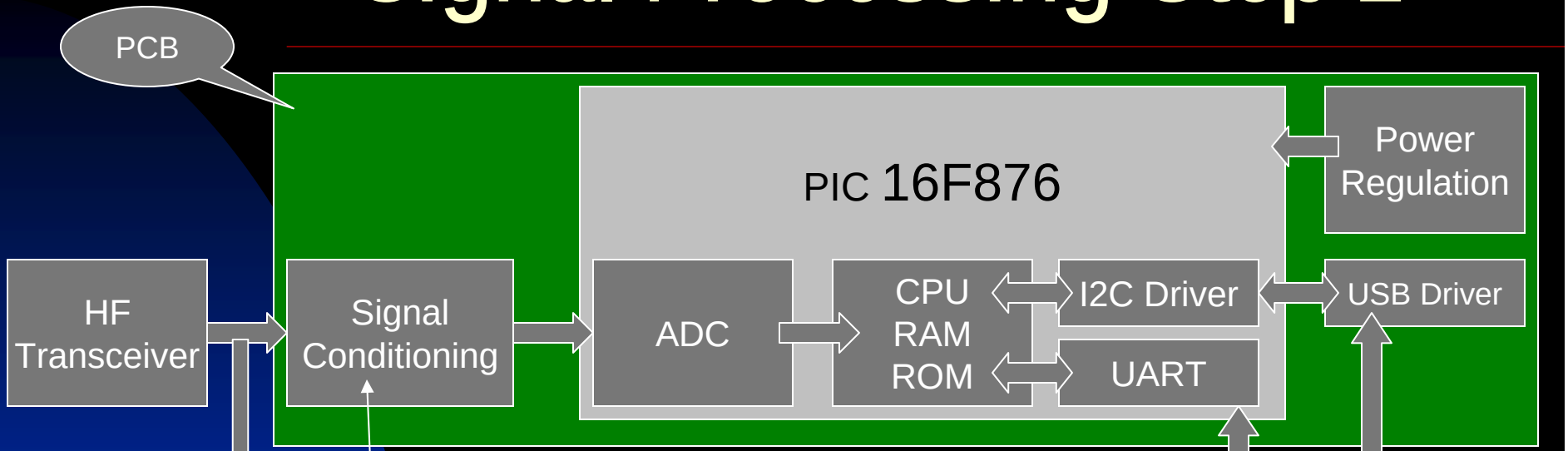
- Peak RF voltage is given by
- $\mathbf{V_0} = (2R_AP)^{1/2} = \mathbf{10P^{1/2}}$ where R_A=50Ω, the antenna impedance
- Voltage applied to the ADC will be
- $\mathbf{V = V_0\ R_2/(R_1+R_2)}$
- Setting V=5 (the ADC limit), substituting for V_0 and solving for R_2 / R_1 gives:
- $\mathbf{R_1 / R_2 = 2P^{1/2} - 1}$
- Examples: if R_2 = 2 KΩ then

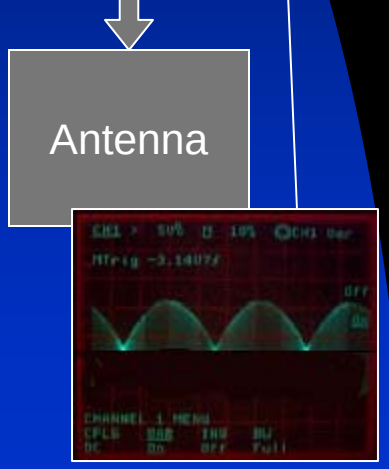
  P = 4 watts, R_1 = 6KΩ

  P = 100 watts, R_1 = 38KΩ



Signal Processing-Step 2





Antenna

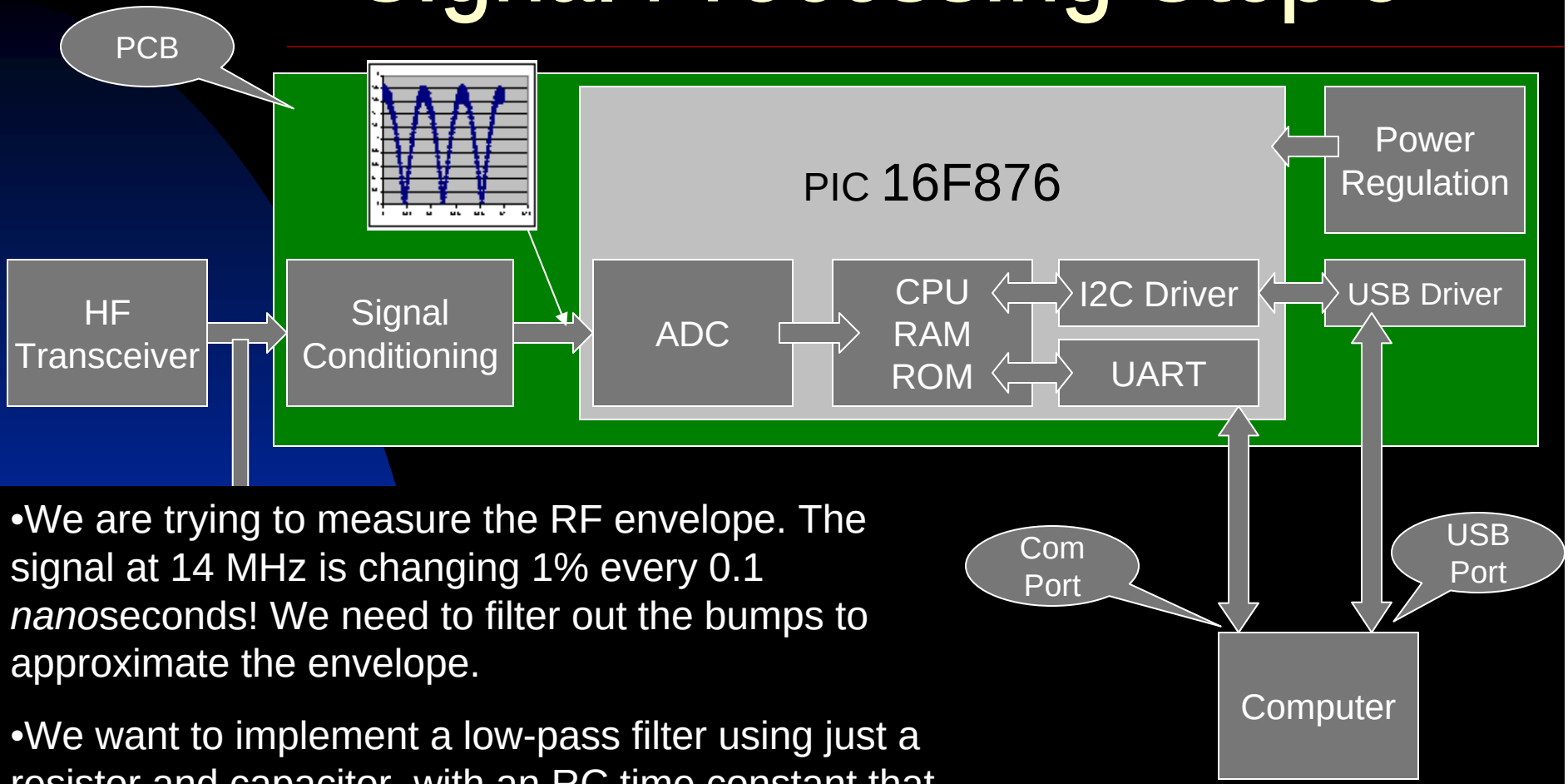
Com Port

USB Port

Computer

- Signal is AC. ADC sampling will not occur in any definite phase, so we will get an undesirable mix of both positive and negative samples.
- Solution: half-wave rectify the signal prior to sampling using a germanium diode. Why half-wave and not full-wave? Why germanium and not silicon?

Signal Processing-Step 3



- We are trying to measure the RF envelope. The signal at 14 MHz is changing 1% every 0.1 *nano*seconds! We need to filter out the bumps to approximate the envelope.
- We want to implement a low-pass filter using just a resistor and capacitor, with an RC time constant that allows the 31 Hz signal to pass, but blocks the RF.
- The PIC's ADC has an internal input capacitance of 5 pf. So the question is: how to determine the value of the resistor?

Signal Processing-Step 3

The cutoff frequency in this low pass filter is

$f_c = 1/(2 \pi RC)$

And the “gain” of the filter is given by

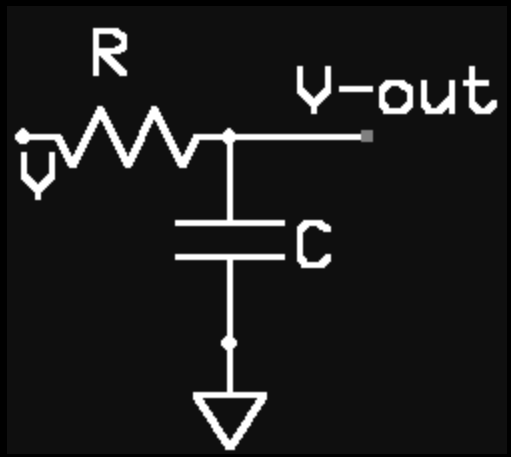
$G = 1/(1 + (f/f_c)^2)^{1/2}$

A good cutoff to choose would be the geometric mean between 31 Hz (the signaling envelope) and 14 MHz (the carrier frequency). This works out to be 21 KHz.

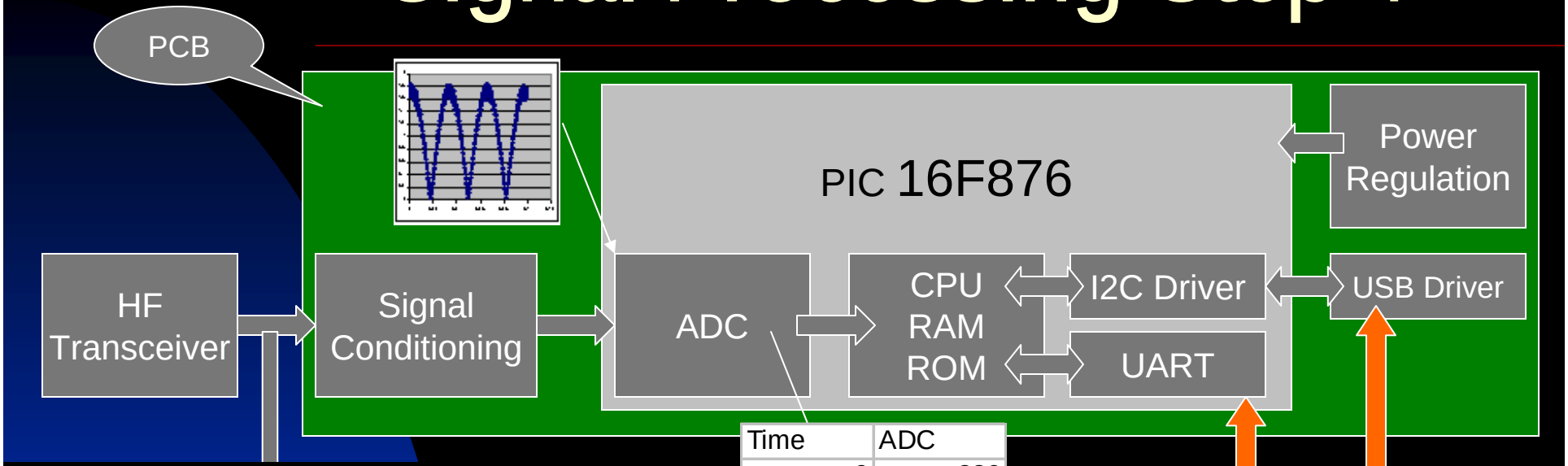
Then G for the RF component is 1.5×10^{-3} and for the PSK31 component is $1 - 10^{-6}$.

Taking a value of C to be 0.1 uf, we have

$R = 1/(2 \pi f_c C) = 113$ Kohm ~100K

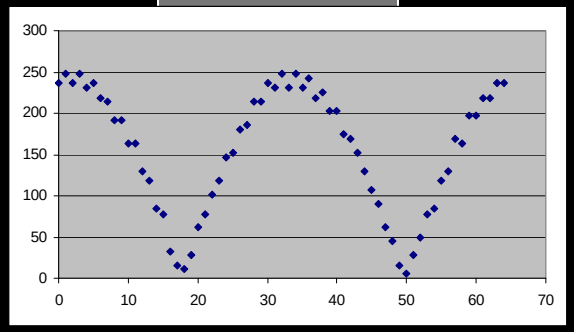


Signal Processing-Step 4

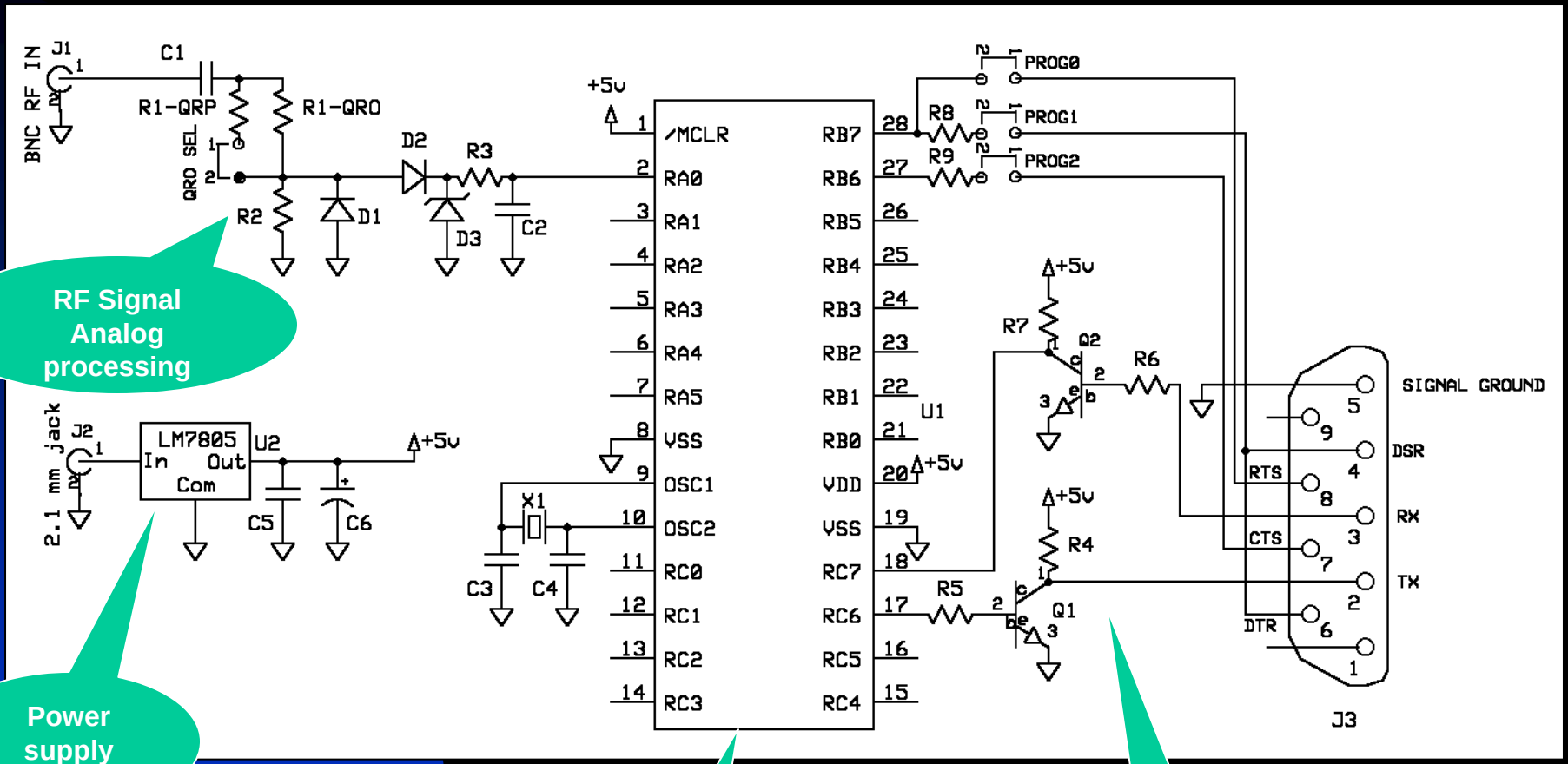


- The firmware now waits for a request from the computer, then instructs the ADC to sample the signal, stores the digital result in a table in RAM memory, then transfers the buffered bytes to the computer.
- How often to sample? Recall that the PSK31 envelope has a frequency of 31.25 Hz (1 bit or cycle per 32 msec). By sampling the waveform every millisecond we obtain exactly 32 samples for each PSK31 bit. In 64 msec, we can acquire 2 PSK31 bits in 64 samples.

Time	ADC
0	236
1	248
2	236
3	248
4	231
5	236
6	219
7	214
8	191
9	191
10	163
11	163
12	129
13	118
14	84
15	78



PSKMeter Circuit Diagram



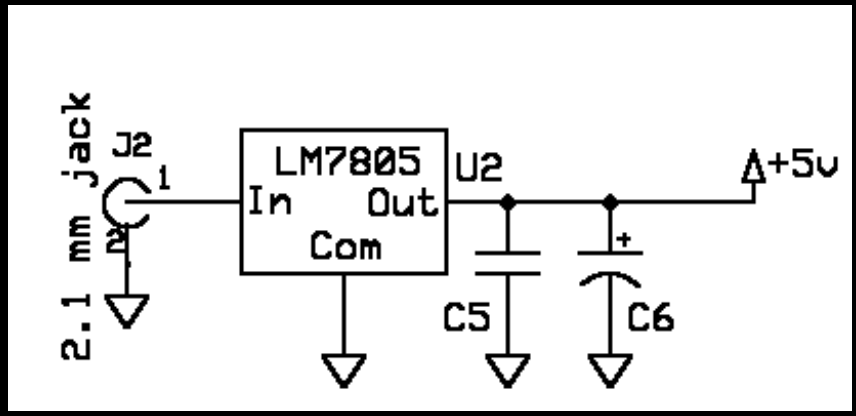
RF Signal Analog processing

Power supply

PIC Microcontroller

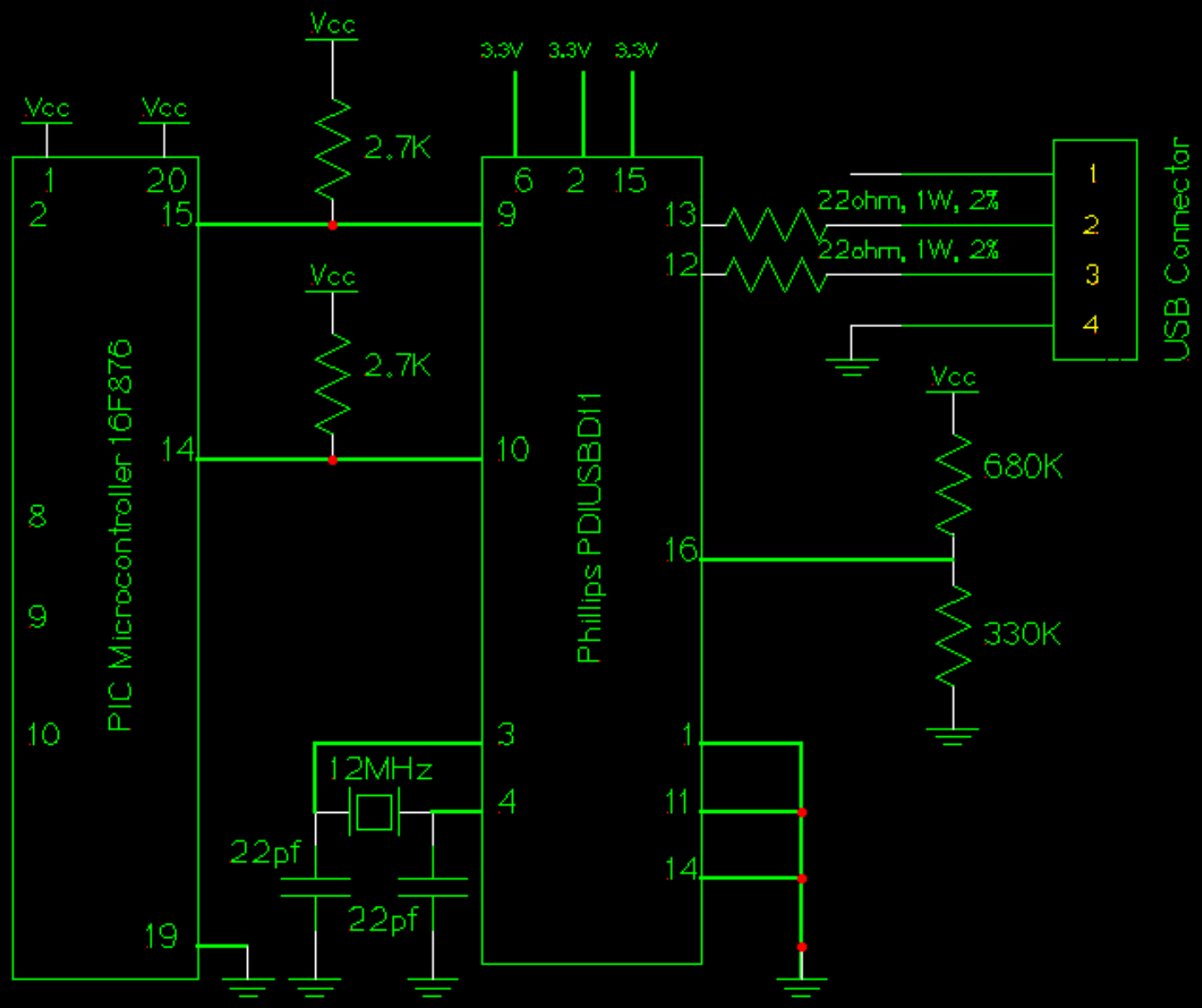
Serial I/O to computer

Adding the Power Supply Circuit

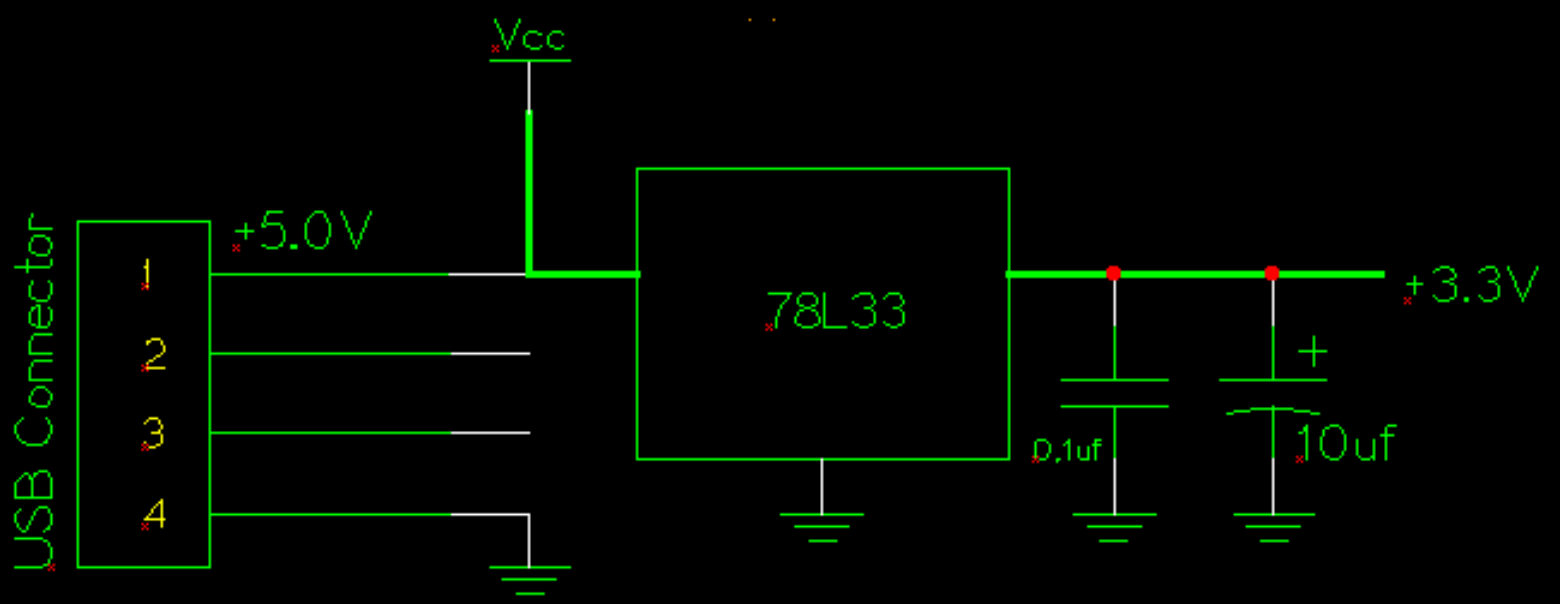


- Accepts any positive voltage from 7.5 to 35 volts and outputs +5.0 volts. Package: TO-92.

PSK meter with USB interface



Revised Power Circuit for USB

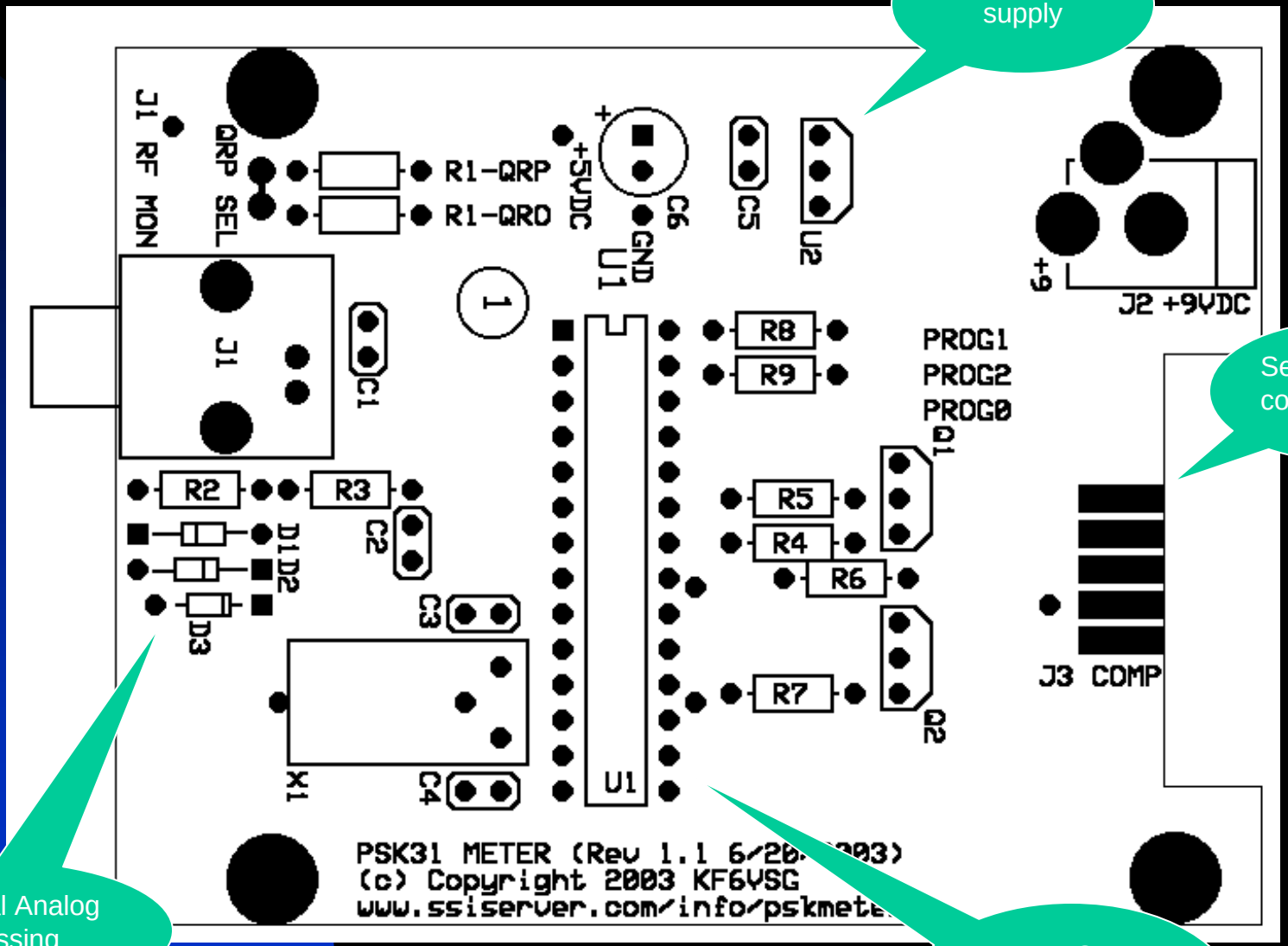


- Other enhancements:
- Option for either RS232 or USB interface on the same card
- Diode isolation between USB power and external power
- On board PIC programming

PCB Layout

- In theory the circuit should and does work, *but…*
- You have RF signals and digital signals on the same board…*so what*?
- Strong RF signals at 14 MHz will mix with the 20MHz crystal and shift the CPU's clock, and the firmware will crash
- RF should be isolated from any signals that could creep into your computer, or you will have software problems
- The solution is to *be careful with the physical layout* of the board:
  - Keep all RF traces as short as possible.
  - Use plenty of ground plane on the copper side of the board, and even the component side also.
  - Keep the PIC and its crystal away from the RF signal conditioning components and traces.
  - Use ferrite beads (toroids) on the cables to your computer.

PCB Layout



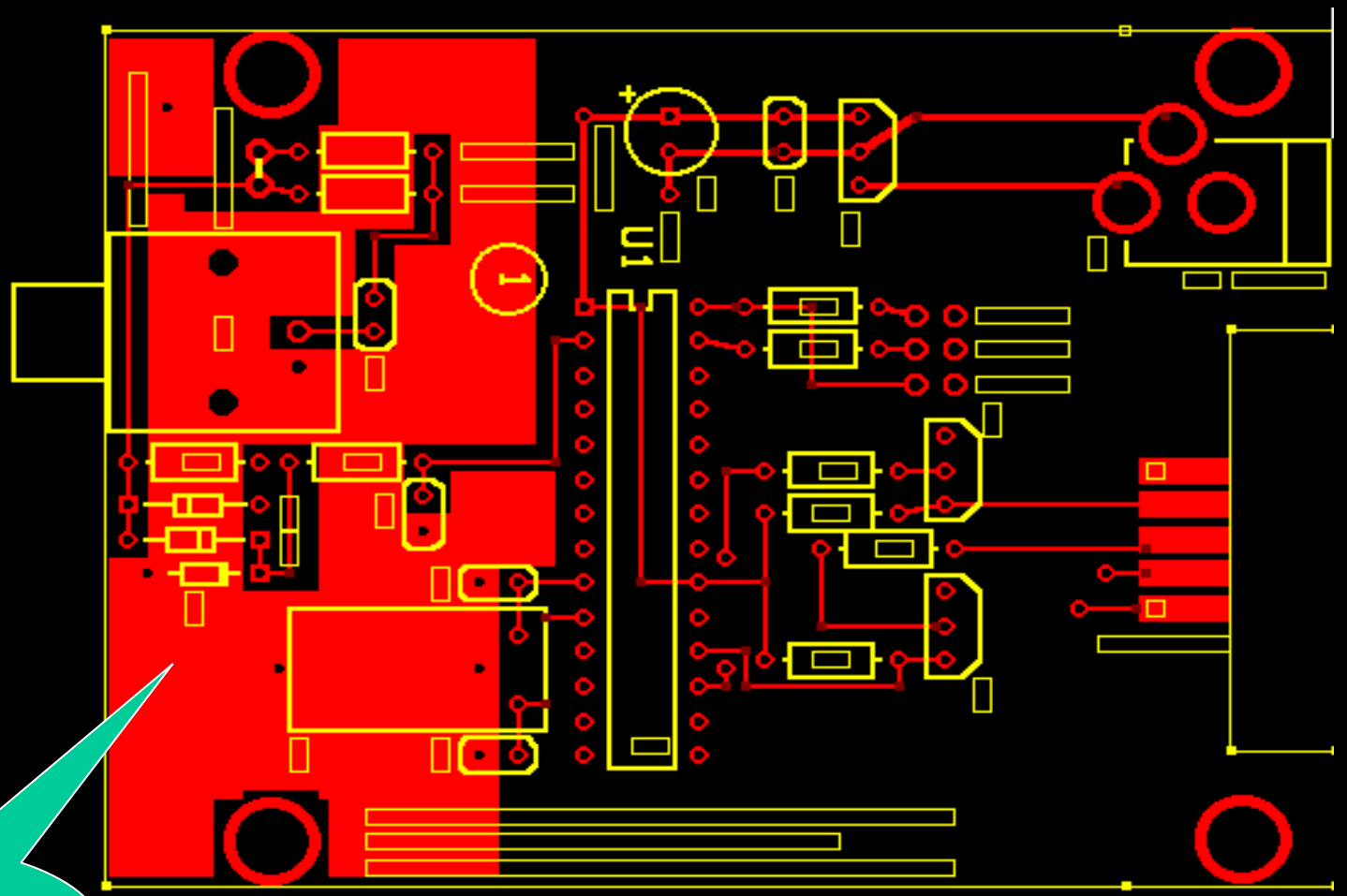
Power supply

Serial I/O to computer

RF Signal Analog processing

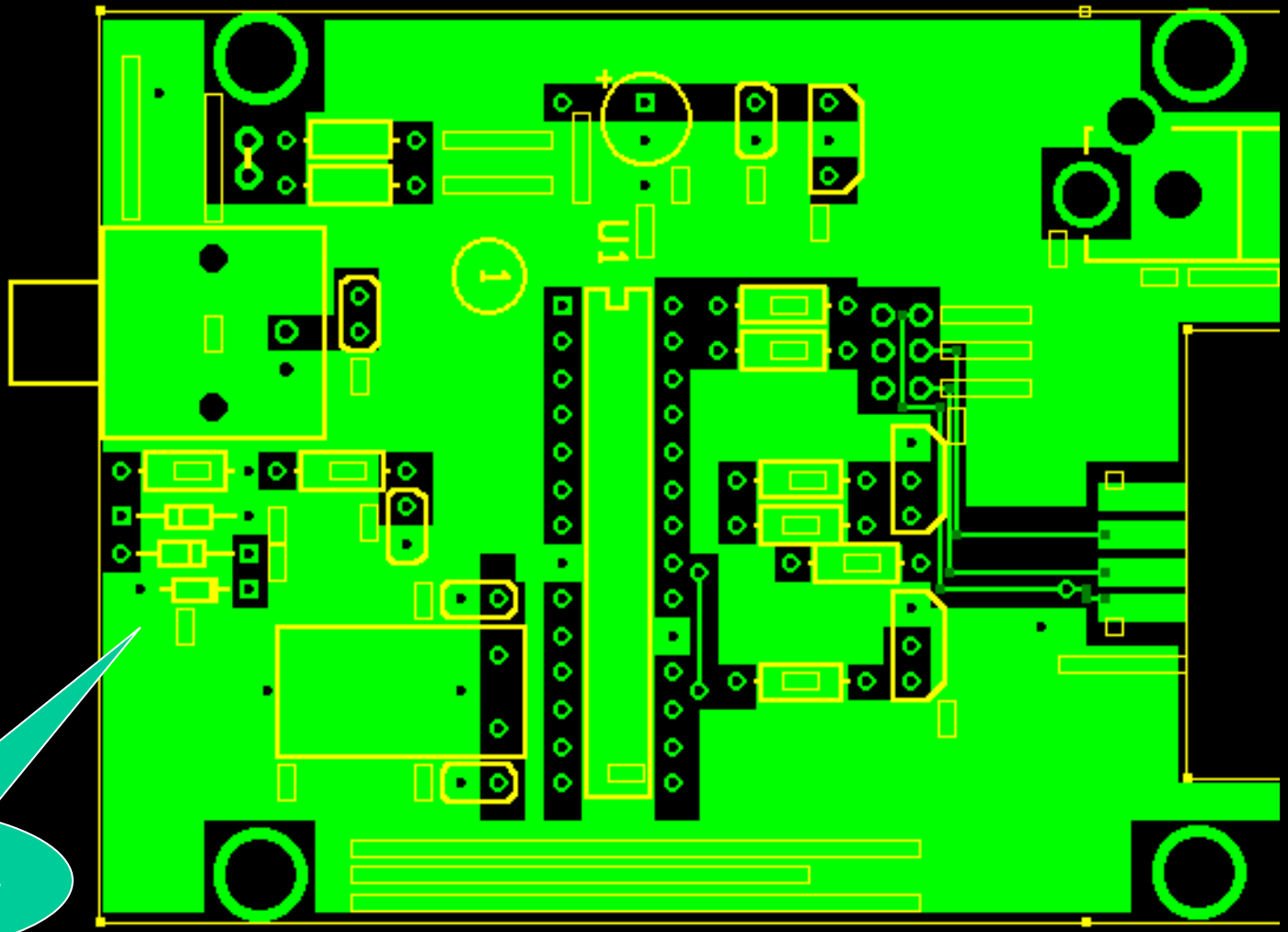
PIC Microcontroller

Component Side Ground Plane



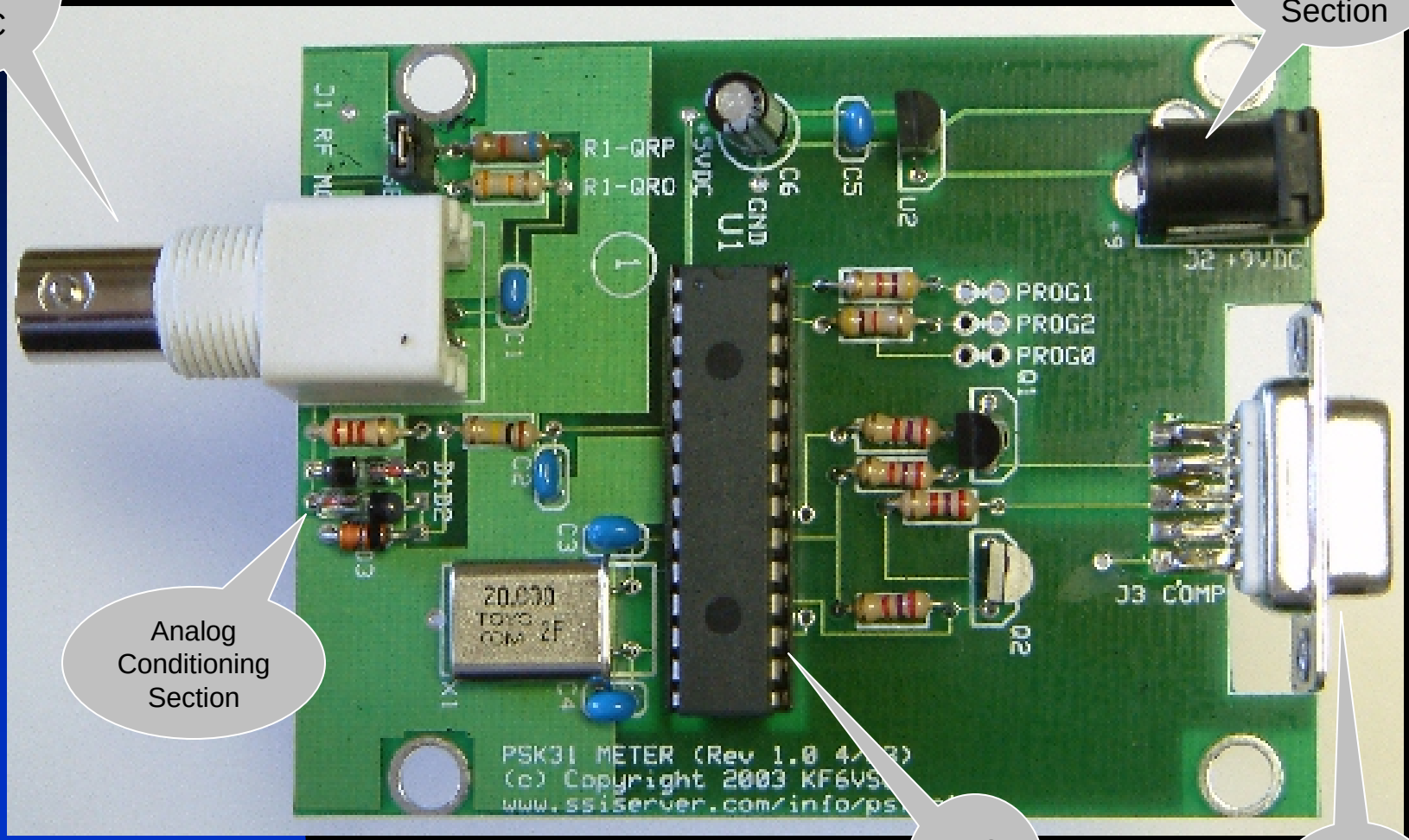
Red indicates component side copper ground plane

Solder Side Ground Plane



Green indicates solder side copper ground plane

PSK Meter Printed Circuit Board



RF Input BNC

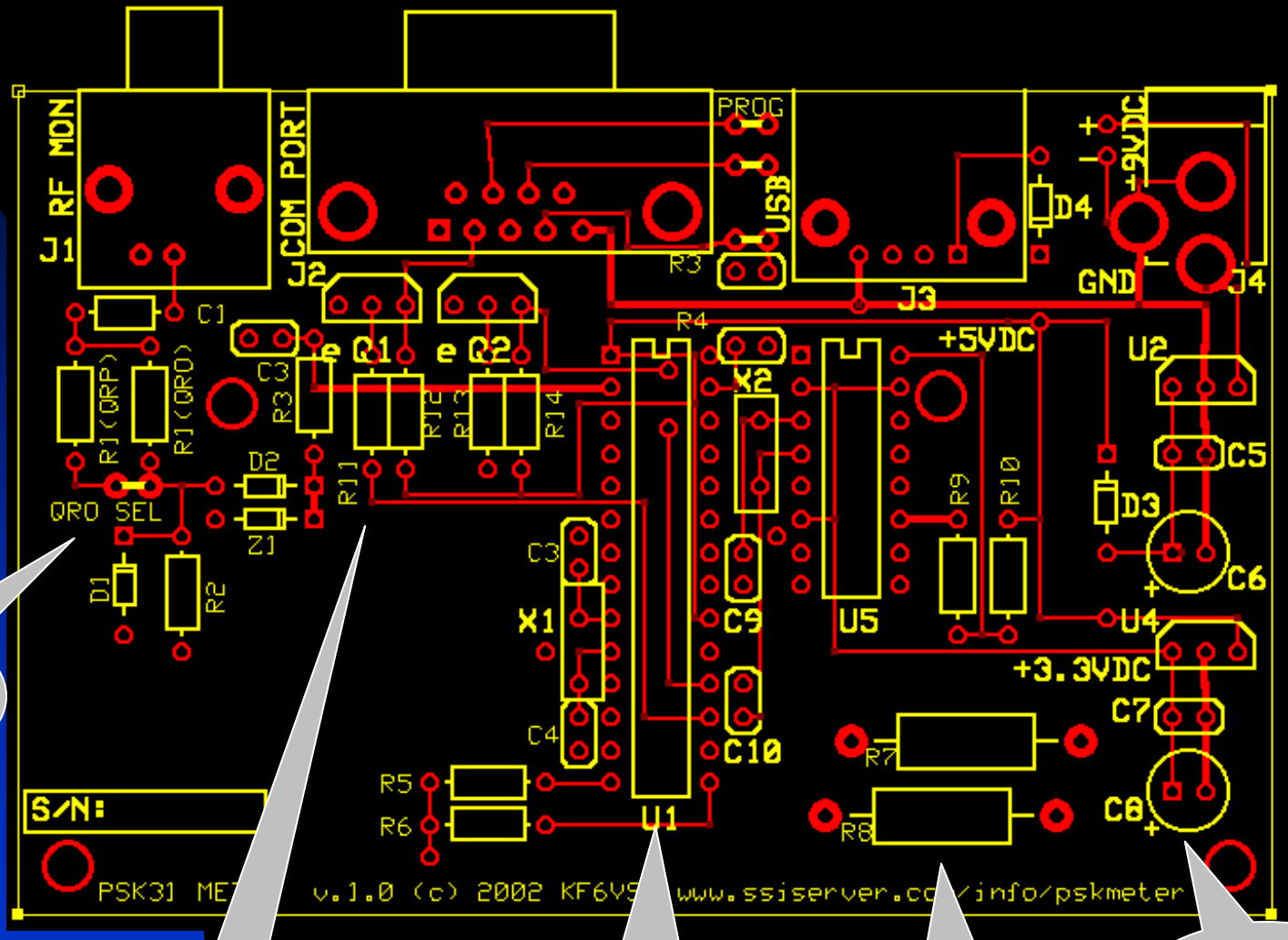
Power Supply Section

Analog Conditioning Section

PIC MPU

RS232 to COM port

PSK Meter Printed Circuit Board



Analog Conditioning Section

RS-232 Section

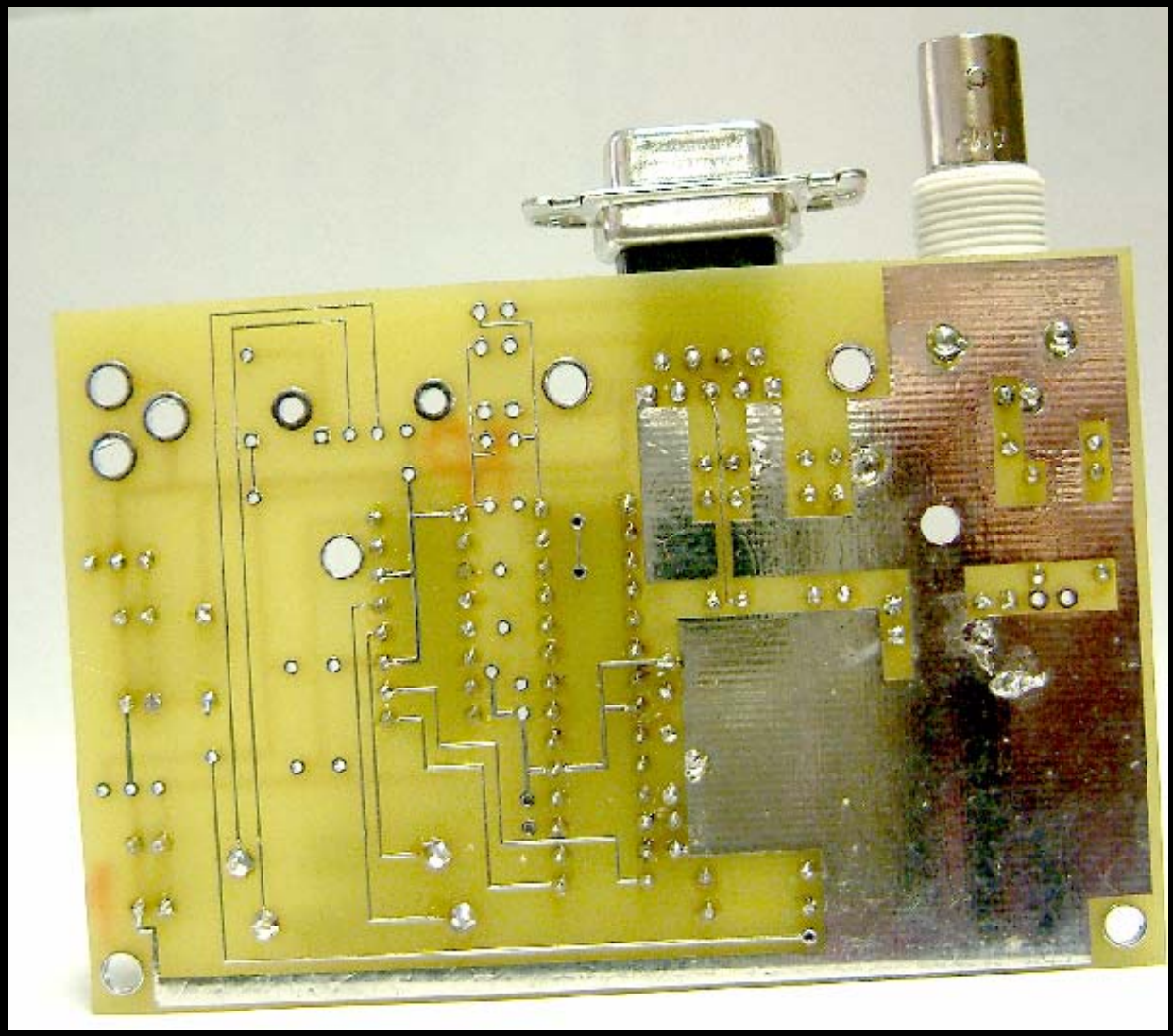
Microprocessor Section

USB Section

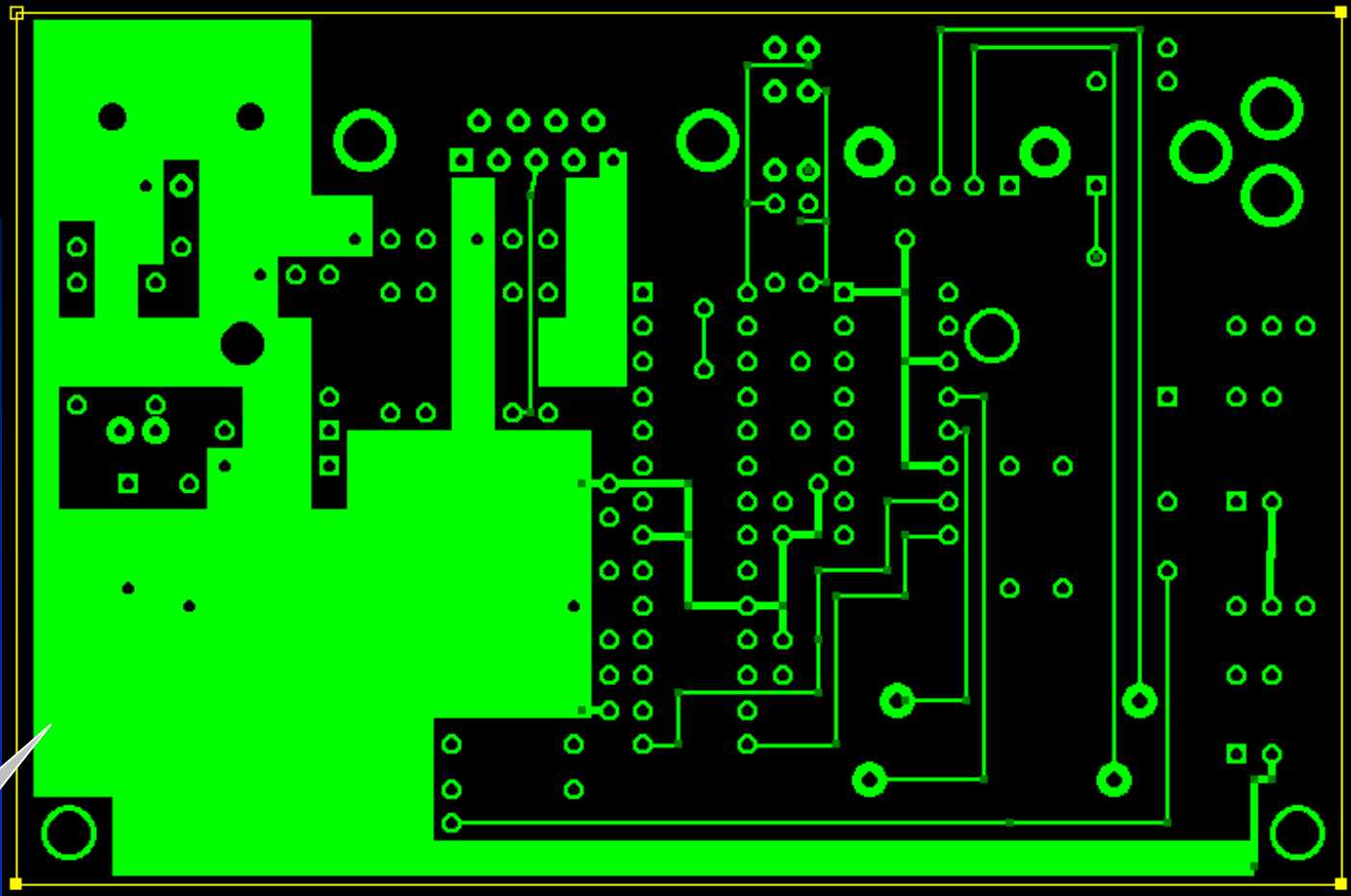
Power Supply Section

PSK Meter Printed Circuit Board

- Solder, partially assembled
- Prototype board is missing silk screen and solder mask
- Notice lots of ground plane used at RF input and signal conditioning section. This is a must for the RF to stay out of the MPU!



PSK Meter Printed Circuit Board



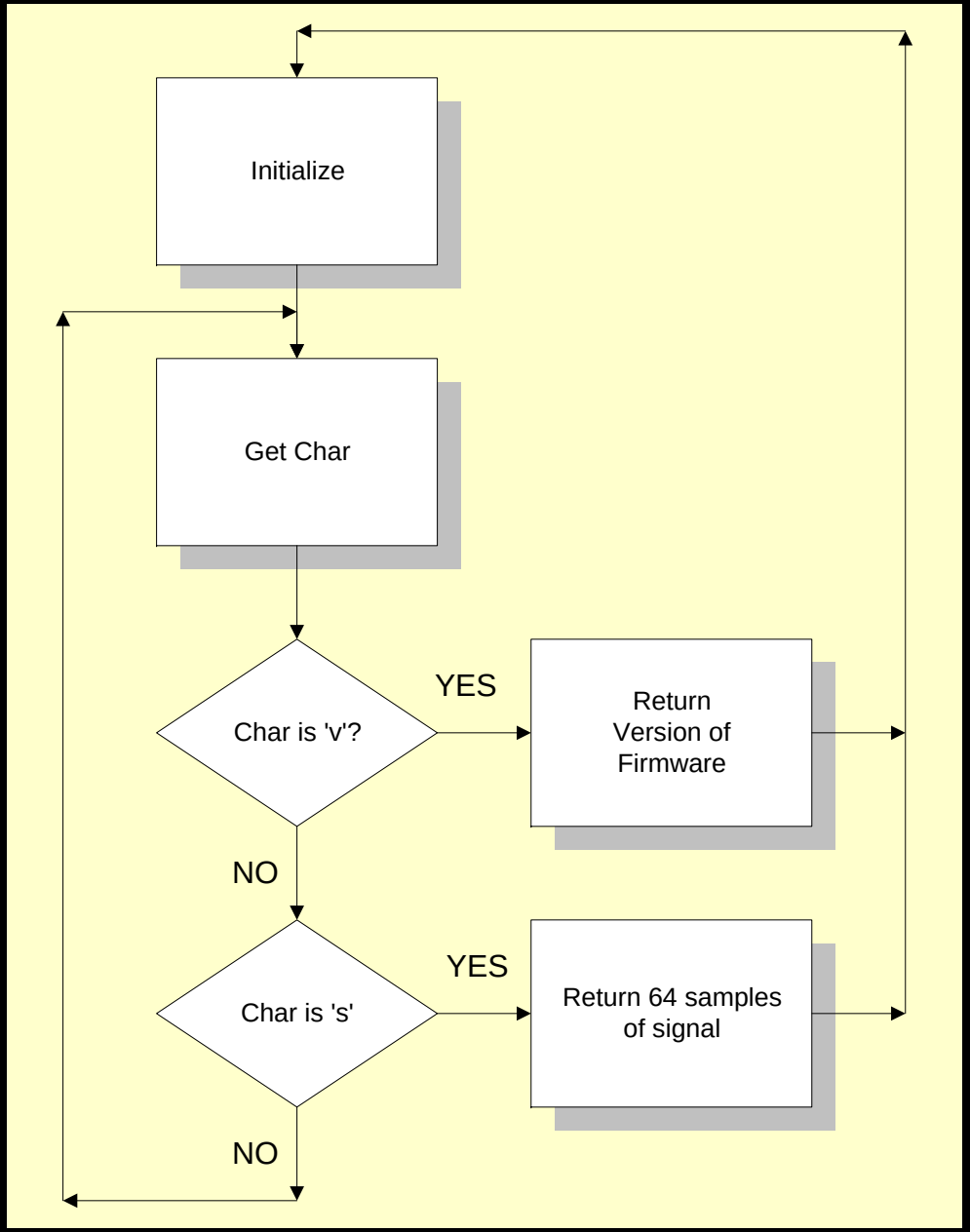
Ground
Plane

Parts List—COM Port version

Item	Count	Circuit Part Number	Description
1	1	R1-QRP	7KΩ ¼ watt 10%
2	1	R1-QRO	38KΩ ¼ watt 10%
3	1	R2	2.2KΩ ¼ watt 10%
4	1	R3	1.5MΩ ¼ watt 10%
5	2	R4,R5	2.5KΩ ¼ watt 10%
6	2	R6,R7	1.0KΩ ¼ watt 10%
7	2	C1,C2	0.1uf
8	2	C3,C4	22 pf
9	1	C5	10uf electrolytic
10	2	D1,D2	Germanium diodes 1N34A or OA47 or NTE109 or ECG109 or SK3090
11	1	Z1	4.7V ½ watt Zener diode, 1N4624 or NTE5009A
12	2	Q1,Q2	2N4401 NPN transistor
13	1	X1	20 MHz xtal, Mouser part 73-XT49U2000-20
14	1	U1	PIC 16F876
15	1	U2	7805 5V reg, TO-92
16	1	28 pin DIP socket	
17	1	J1	BNC edge connector, Amphenol 31-5431-2010
18	1	J2	DB9 female edge connector, Amphenol 745781-4, Mouser part 152-3409
19	1	J4	Power jack, 2.1 mm, edge connector, Digikey CP-202A-ND,JDR # PC21S

Firmware

- Firmware: software that runs on the PIC in ROM
- Initializes hardware and variables
- Waits for a character command from the PC
- Returns either
  - information about itself, or
  - takes 64 samples of the RF envelope at 1 msec intervals



Firmware Source Code

```
#include <16f876.h>
#include <string.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7)

void showversion()
{
 printf("PSKMETER Firmware Version 1.00.\n\r");
 printf("(c) 2002 KF6VSG
         www.ssiserver.com/info/pskmeter\n\r");
}

void senddata()
{
 int i;
 #define MAXSAMPLES 64 //exactly two psk31 cycles
 int sample[MAXSAMPLES];
 // take MAXSAMPLES samples separated by 1 msec:
 for (i=0;i<MAXSAMPLES; i++)
    {
     sample[i] = Read_ADC();
     delay(960); // wait .96 millisecond
    }
 // upload the samples to the windows app
 for (i=0; i< MAXSAMPLES; i++)
   printf("%c", sample[i]);
}
```

Copyright & Version Info

Take & report data

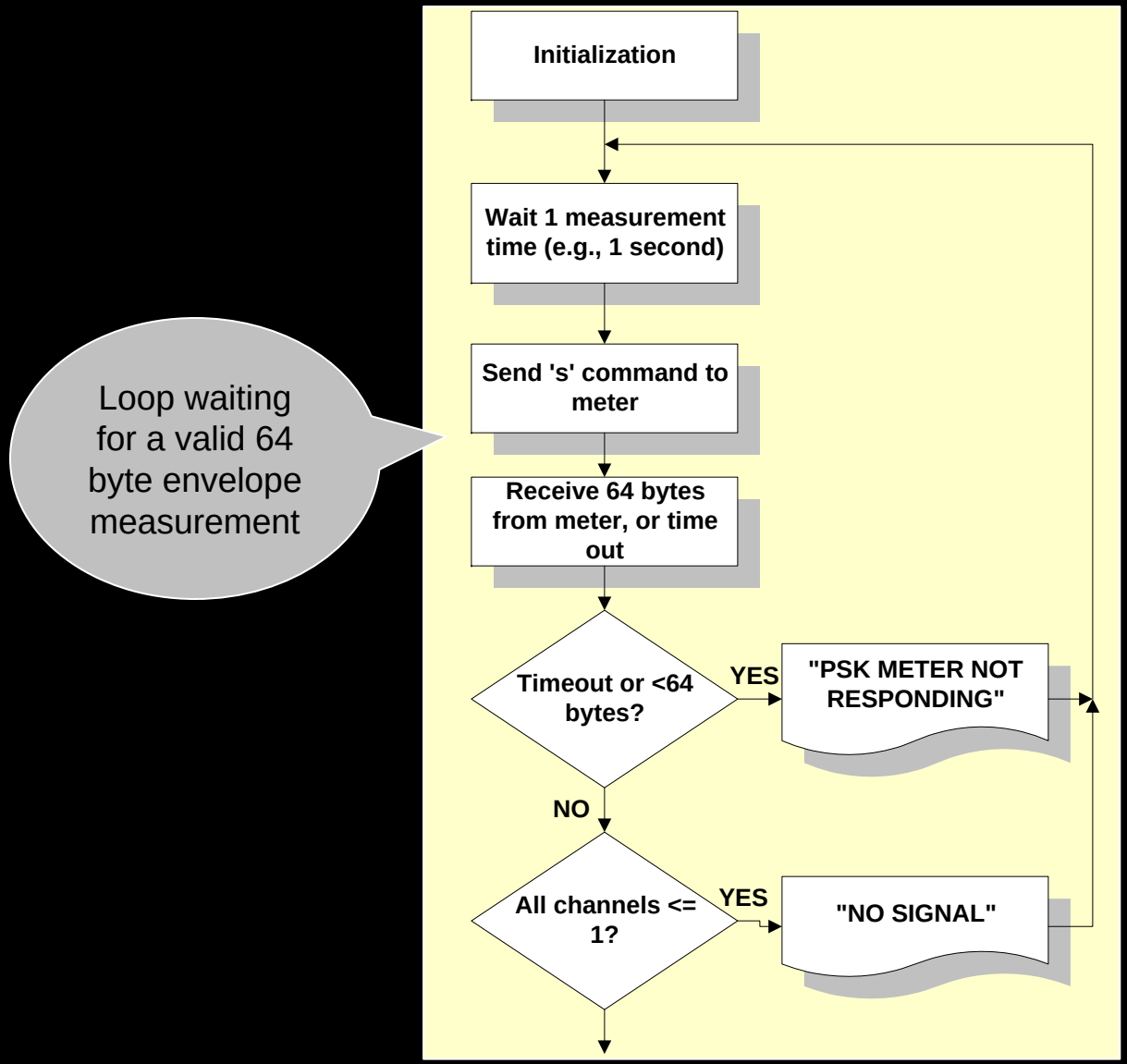
```
main()
{
 char command;
 showversion();
 setup_port_a( ALL_ANALOG );
 setup_adc( ADC_CLOCK_INTERNAL );
 set_adc_channel( 0 );

 do
   {
      /* wait for the command byte 's' or
        'v' to be received from the windows
        application */
   do
     command = getc();
   while ((command != 's') &&
          (command != 'v'));

   if (command == 'v')
        showversion();
      else
        senddata();
   } while (TRUE);
}
```

Initialization

PC Software Flow Diagram—Part 1



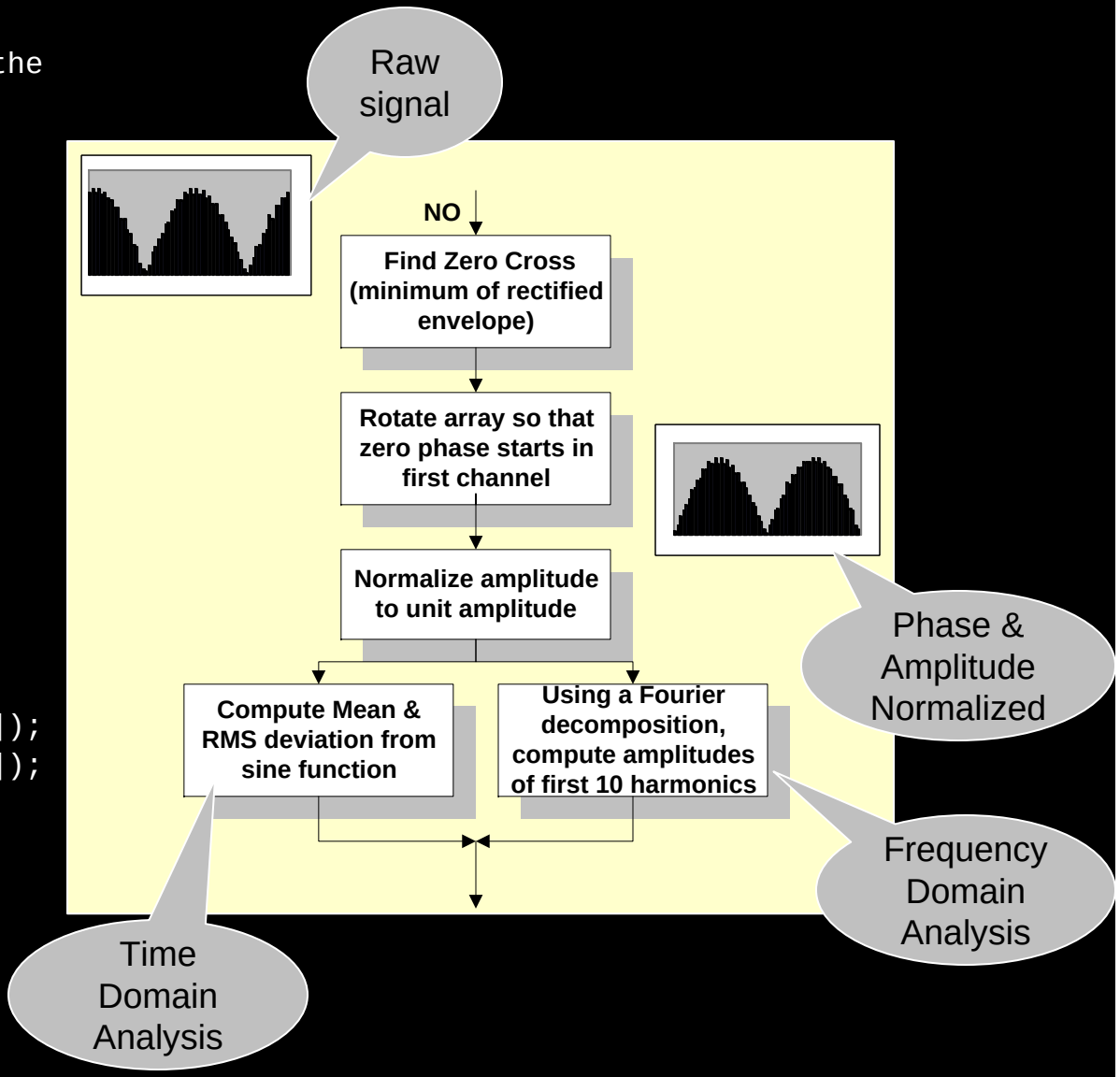
PC Software Flow Diagram—Part 2

Computing RMS and Mean deviation in the *time domain*:

```
for i:=1 to 32 do  begin
  error := s[i] – sin(theta[i]);
  rms := rms + sqr(error);
  mean := mean +error;
end;
rms := sqrt(rms)/(32);
mean := mean/32;
```

Computing harmonics in the *frequency domain*:

```
for j:=1 to 10 do
  for i:=1 to 64 do begin
   a[j] := a[j] + s[i]*sin(j*theta[i]);
   b[j] := b[j] + s[i]*cos(j*theta[i]);
   end;
```



Metrics of Distortion

- RMS is a time domain metric of the departure of the signal from a perfect sinusoidal waveform. An RMS of zero indicates a perfect signal. Greater RMS values indicate clipping and other forms of distortion that result in splatter.
- Mean deviation is another time domain metric that is used to establish if the signal is overdriven (over-modulated) or underdriven (under-modulated). If the mean >0, the signal is over modulated. If the mean <0, the signal is under modulated.
- Software can make decisions regarding the quality of the signal based on these two metrics.
- However, we usually think in frequency space…

Metrics of Distortion-Part 2

- The discrete Fourier coefficients, a_n and b_n are the real and imaginary amplitudes of the signal's fundamental frequency (31.25Hz) and its harmonics (62.5, 93.75, etc.)
- The absolute amplitude at each frequency is sqrt($a_n^2 + b_n^2$).
- A perfect signal would consist only of the fundamental (n=1). The amplitude of all the harmonics (n>1)would be exactly zero.
- The amount of energy in the harmonics gives us the *Inter-modulation Distortion*, or IMD. Since the energy is proportional to the square of the amplitude, we have
- IMD = 10 Log_{10}(H/E).

```
for j:=1 to 10 do
  E := E + sqr(a[j])+ sqr(b[j]);
H := E - (sqr(a[1]) + sqr(b[1]));
F := H/E;
IMD := 10*log10(F);
```

Metrics of Distortion—Part 3

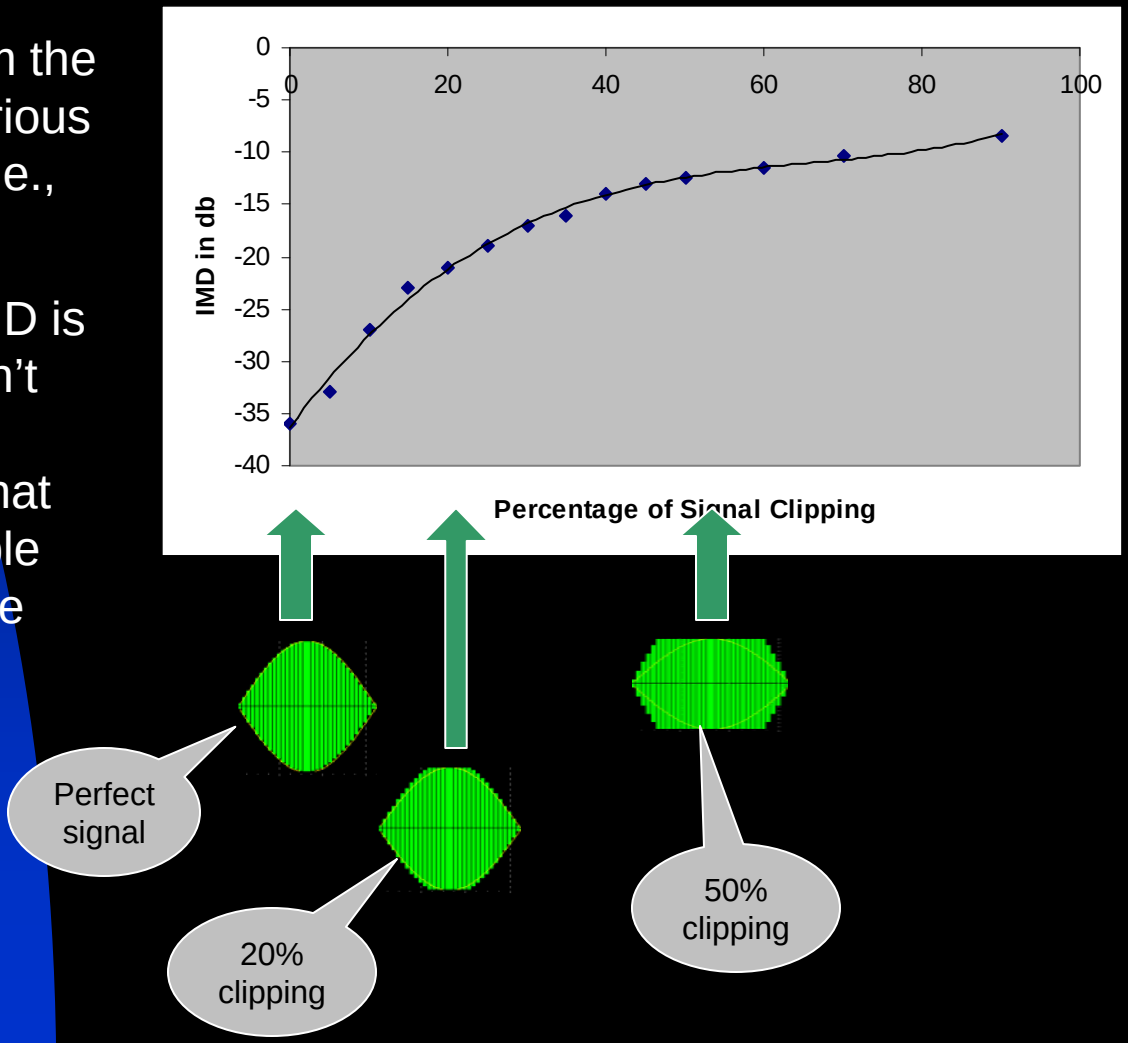
- IMD = 10 Log_{10}(H/E)
- Since the energy in the harmonics will always be less than the total energy, H/E will always be a fraction, so IMD will always be negative.
- IMD is quoted in units of dB.
- Large absolute IMD is better. Smaller absolute IMD is worse.

Example: IMD vs. Clipping

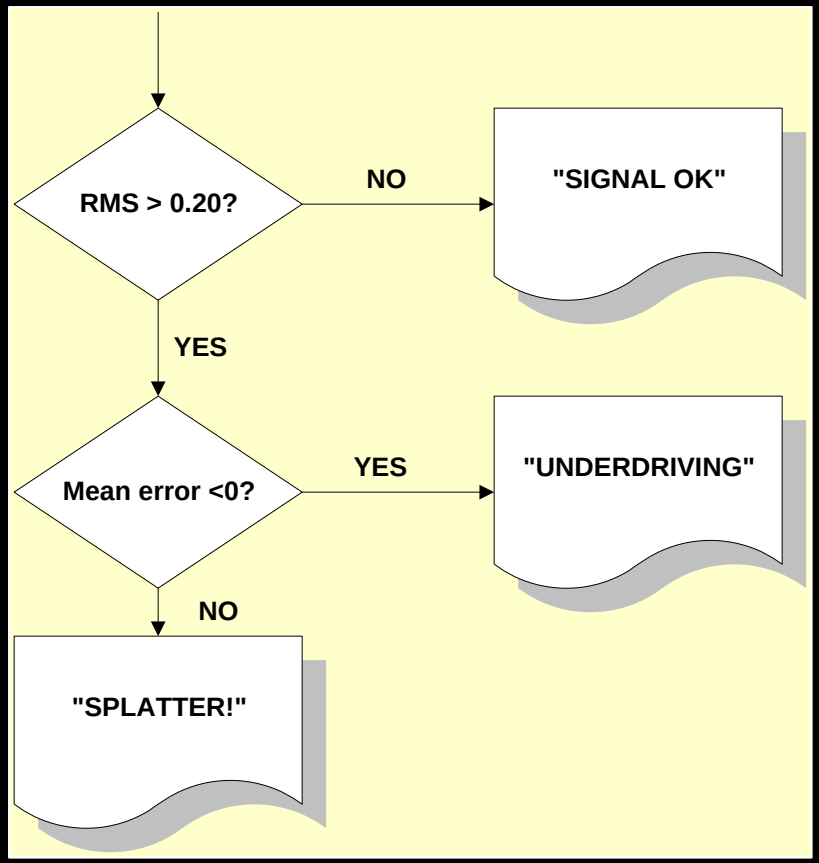
To create this example, I simulated a PSK31 signal with a small amount of white noise, and then computed the IMD from the Fourier analysis for various degrees of “clipping”, I.e., flat-topping the signal.

In the real world, an IMD is –30db is excellent. Don’t expect to do better. An acceptable IMD (one that won’t result in noticeable splatter) would be in the range of –20 to –30.

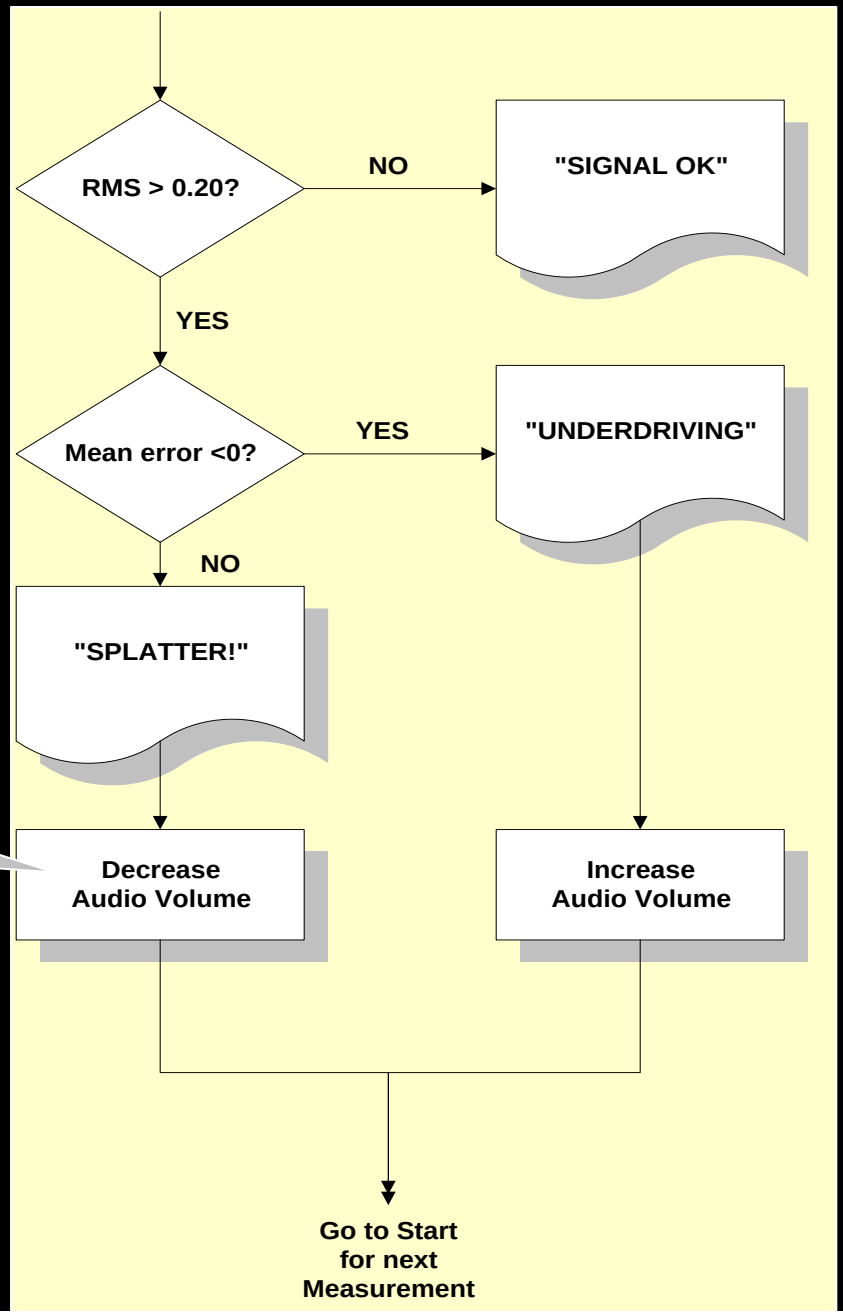
If you are getting IMD reports of –10, you are splattering all over the band!



PC Software Flow Diagram—Part 3



PC Software Flow Diagram—Part 4



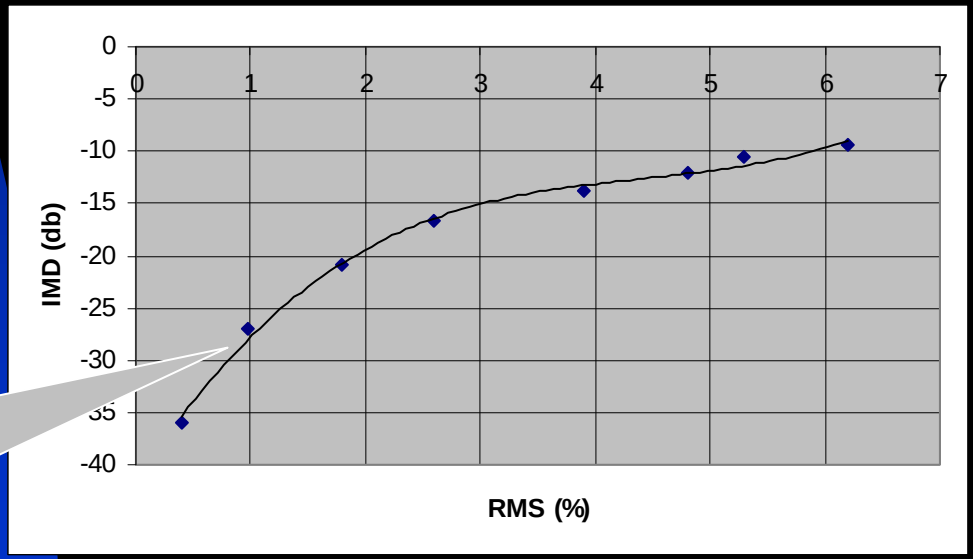
Automate the system: Add a proportional negative feedback loop!

Monitor and Control

- The PSK Meter monitors your RF output and provides a visual, scope-like display to confirm that your signal is clean.
- However, the software also controls the audio level of your sound card to obtain the largest signal without splatter—automatically!

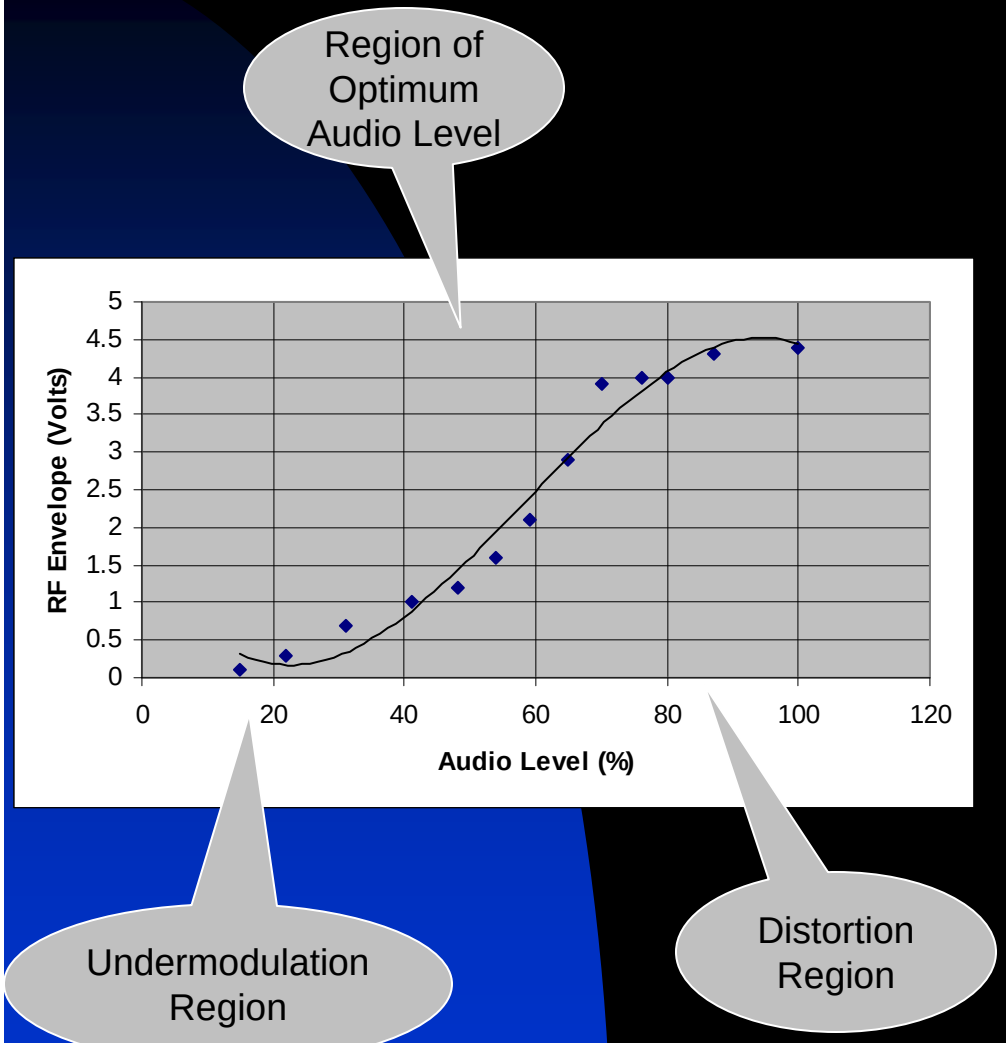
Control variables—what to use?

- RMS (deviation from ideal signal), in percent
- IMD (harmonic disortion), in dB
- Average deviation—discerns between overmodulation and undermodulation



In most cases, RMS and IMD can be used interchangably as the control variable

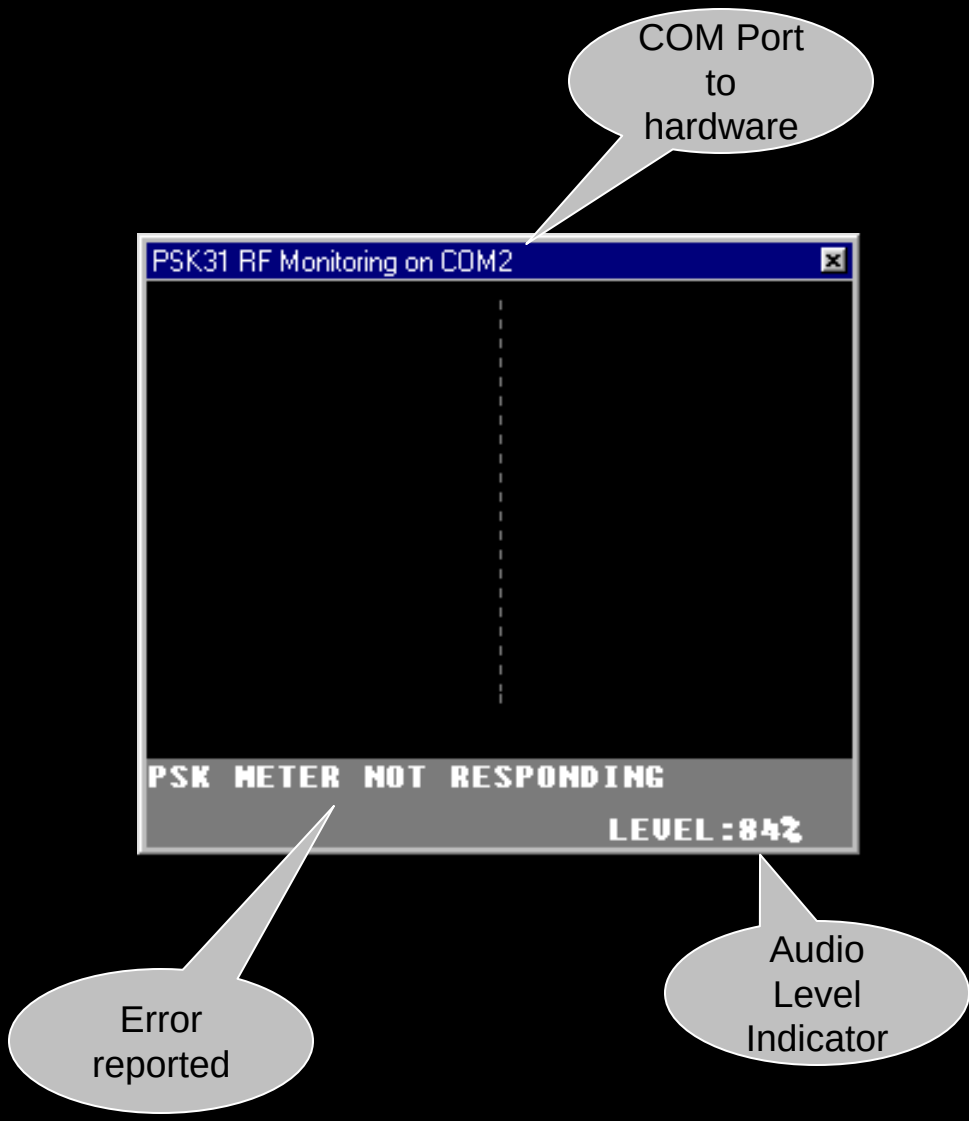
A graphical look at feedback



- Plot the peak PSK31 envelope vs audio level.
- Notice that clipping occurs for the top 1/3 in audio level.
- Define underdriving as the bottom 1/3 in audio level.
- The feedback loop in the software attempts to find and place the audio level in the middle third, tending to the largest signal without distortion.

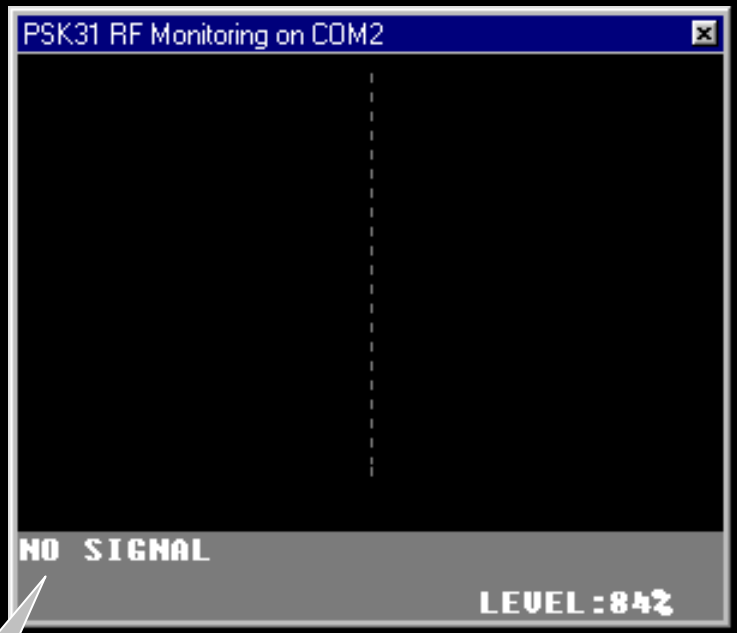
A look at the PSKMeter Software

- When PSKMeter is launched, it displays the COM port it uses to connect to the PSKMeter hardware.
- If the hardware fails to respond, a message is displayed.
  - Wrong COM Port
  - No serial cable
  - No PSKMeter hardware
  - PSKMeter hardware not powered up



A look at the PSKMeter Software

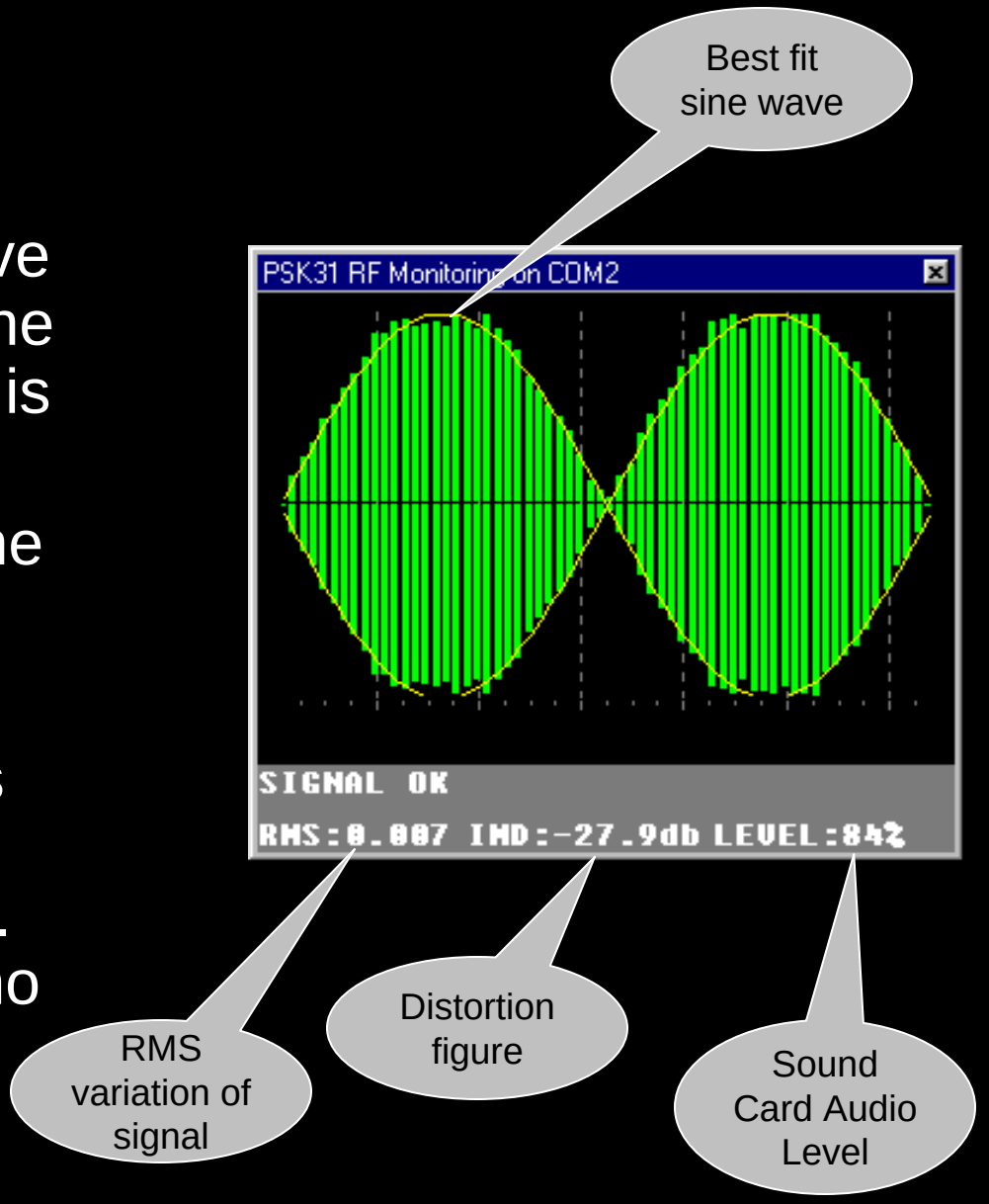
- If the hardware connects, the software periodically issues a request for waveform sample.
- When you are not in transmit mode, no signal is returned.



Absence of Signal

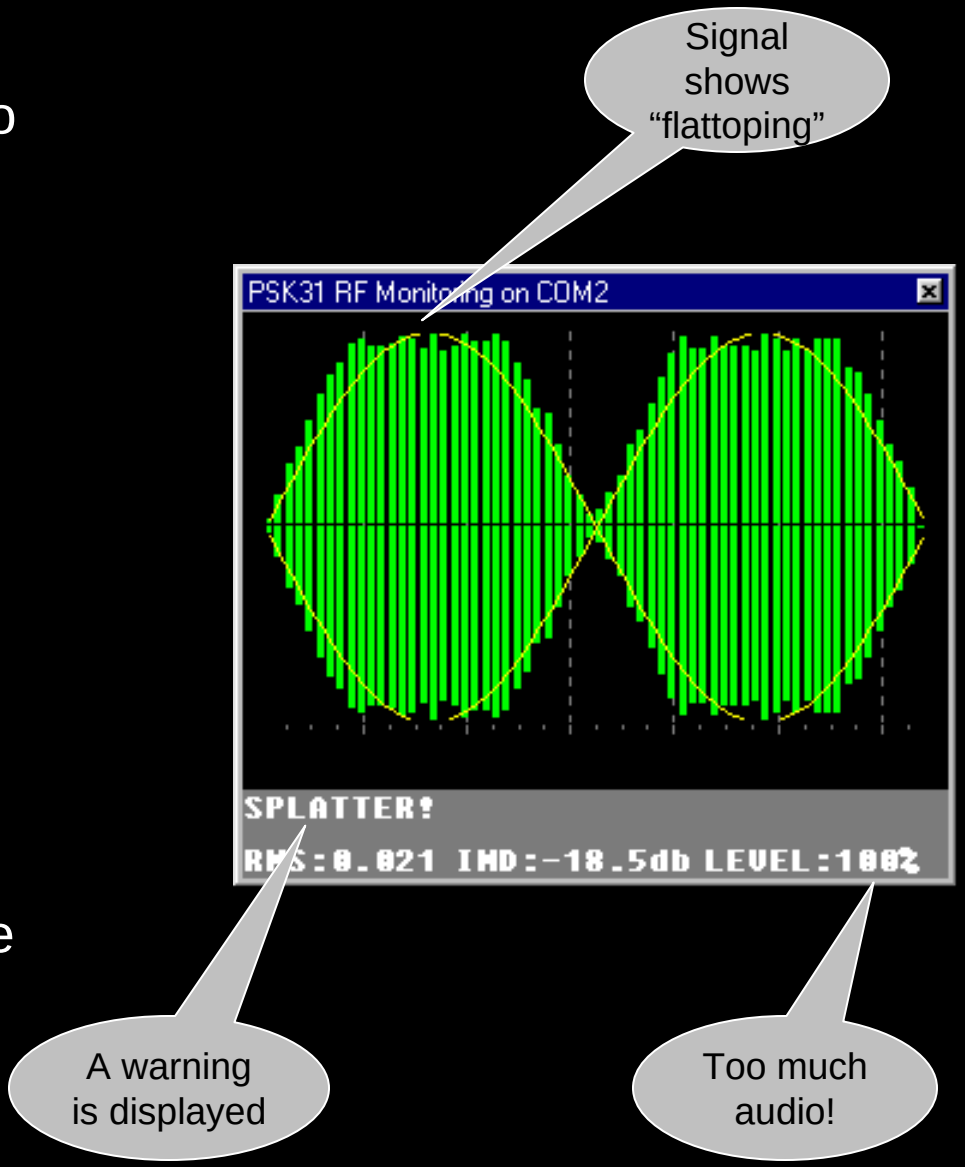
A look at the PSKMeter Software

- In transmit mode, the RF output is periodically sampled and displayed.
- The best-fit sine wave curve is displayed. Variation of the signal from this sine wave is a measure of distortion.
- RMS variation from the sine wave is displayed.
- IMD is displayed.
- If RMS is small and IMD is large, the indication “SIGNAL OK” is displayed. Your signal is clean, and no adjustment (automatic or manual) is necessary.



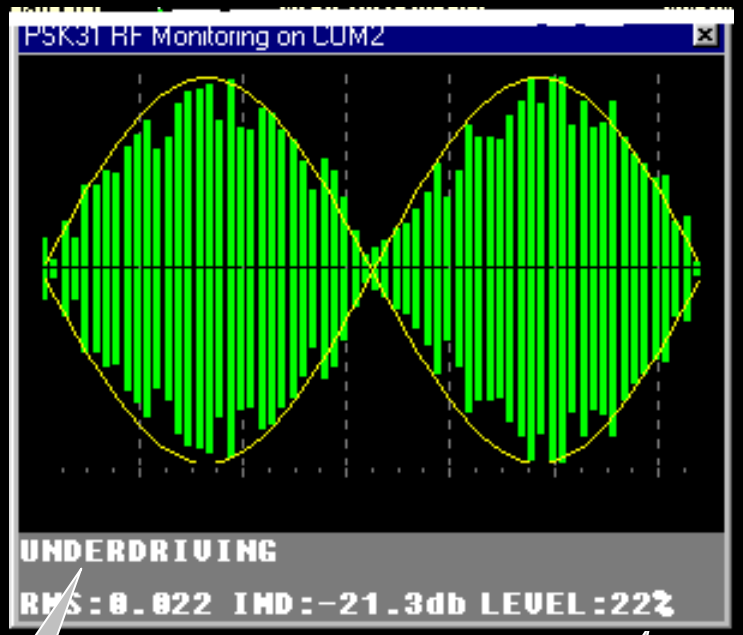
A look at the PSKMeter Software

- If you are over-driving your transmitter, your signal is going to show visible evidence of distortion.
- Example: flat-topping (clipping).
- The RMS will be large and the IMD will be small.
- The result is splatter, as indicated. You are causing interference to other portions of the band, and your signal will not be as copiable!
- When in automatic mode, the software will automatically reduce the sound card's audio level to completely remove the splatter.



A look at the PSKMeter Software

- If you are under-driving your transmitter, your signal will be weak and noisy, but not distorted.
- Because your signal is weak, you will not be as copiable.
- When in automatic mode, the software will automatically increase the sound card's audio level, boosting your output power without introducing distortion.

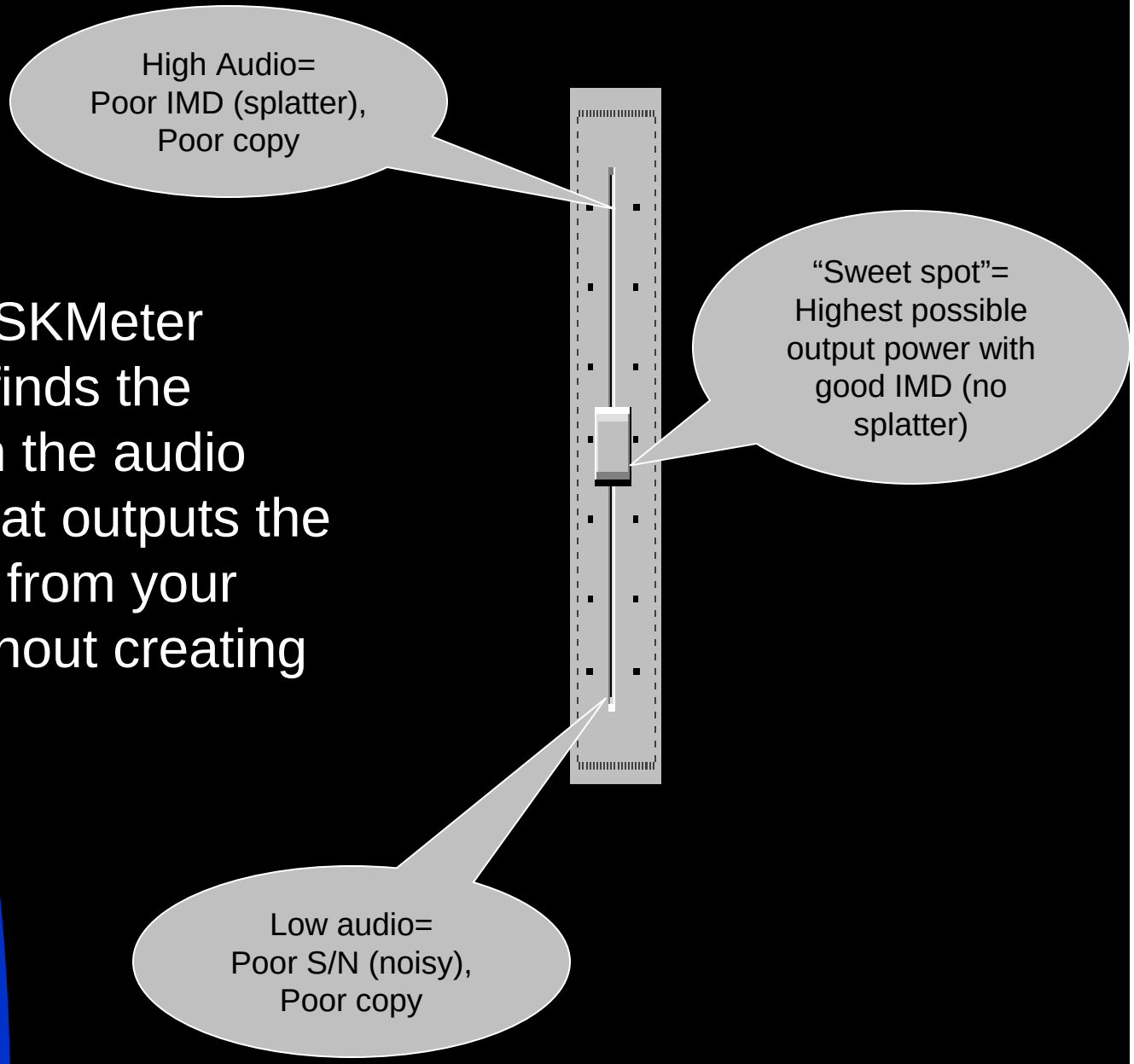


A warning is displayed

Feeble audio

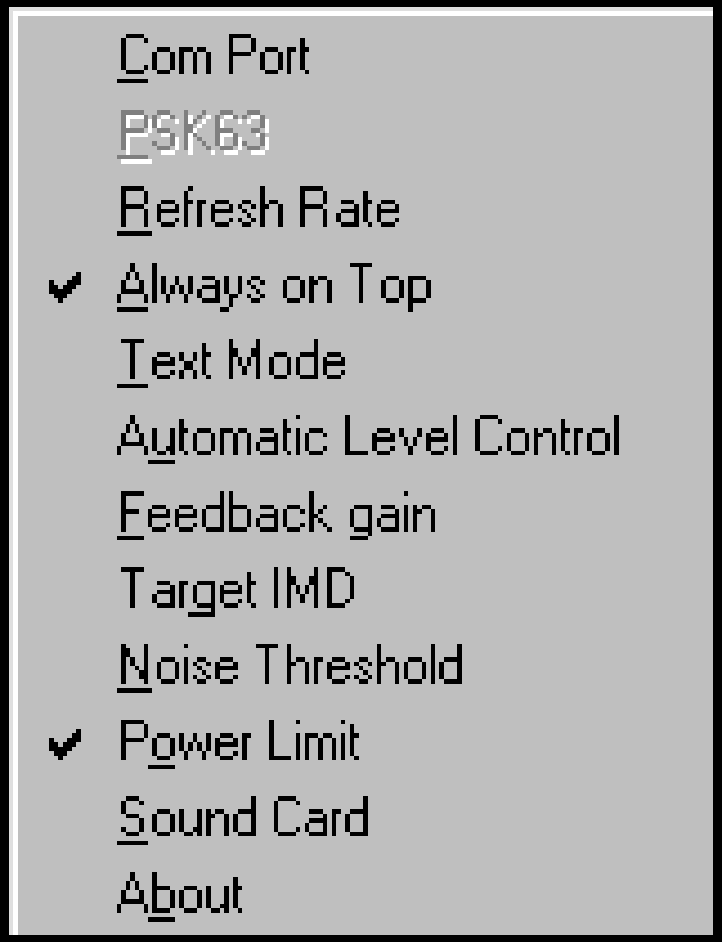
PSKMeter Benefit:

- Bottom line: PSKMeter *automatically* finds the "sweet spot" in the audio level setting that outputs the highest power from your transmitter without creating splatter.

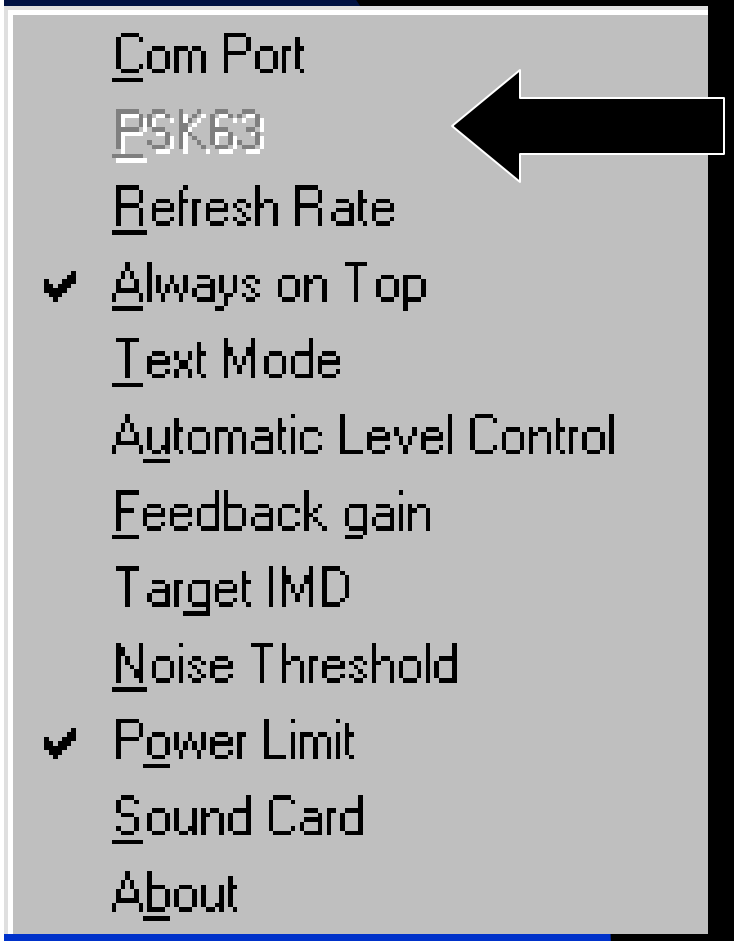


PSKMeter User Interface

- Right-click on the meter brings up the user interface menu:
- Com port selection
- Mode selection (PSK31/PSK63)
- Refresh (sampling) rate
- Force window to be always visible
- View data numerically
- Force automatic control of audio level
- Gain selection
- Desired IMD selection
- Noise (floor) threshold
- Maximum power emission selection
- Sound card selection

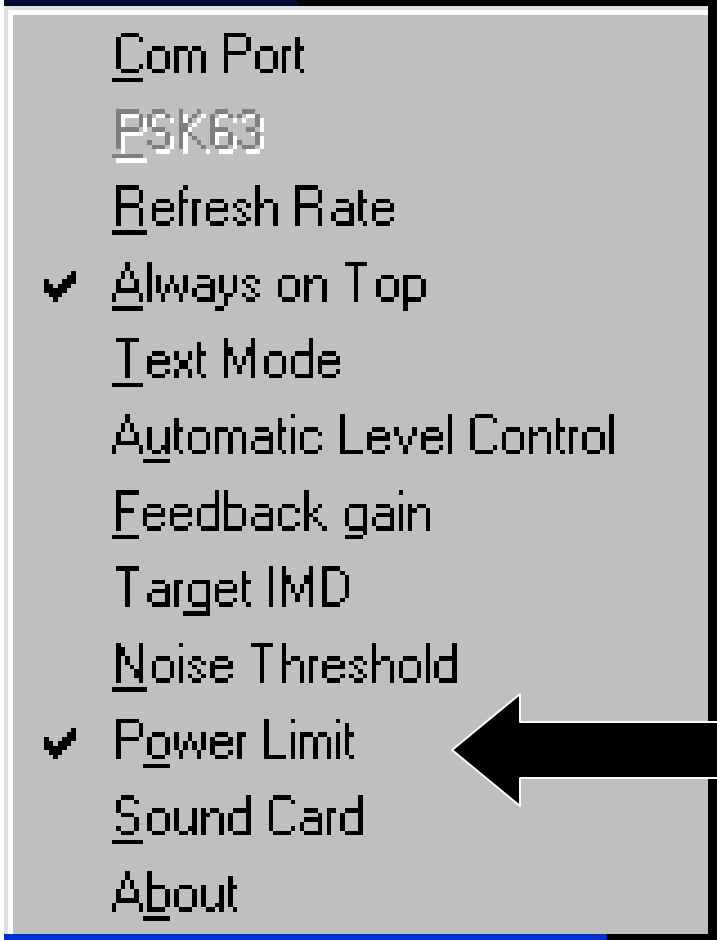


Mode Selection



- "Standard" mode is PSK31 (31.25 Hz = 32 msec per bit)
- Recently we have the ability to transmit PSK63 (62.50 Hz = 16 msec per bit)
- By selecting PSK63, a command it sent to the firmware that instructs it to use ½ msec sampling time instead of 1 msec. Everything else works the same way.

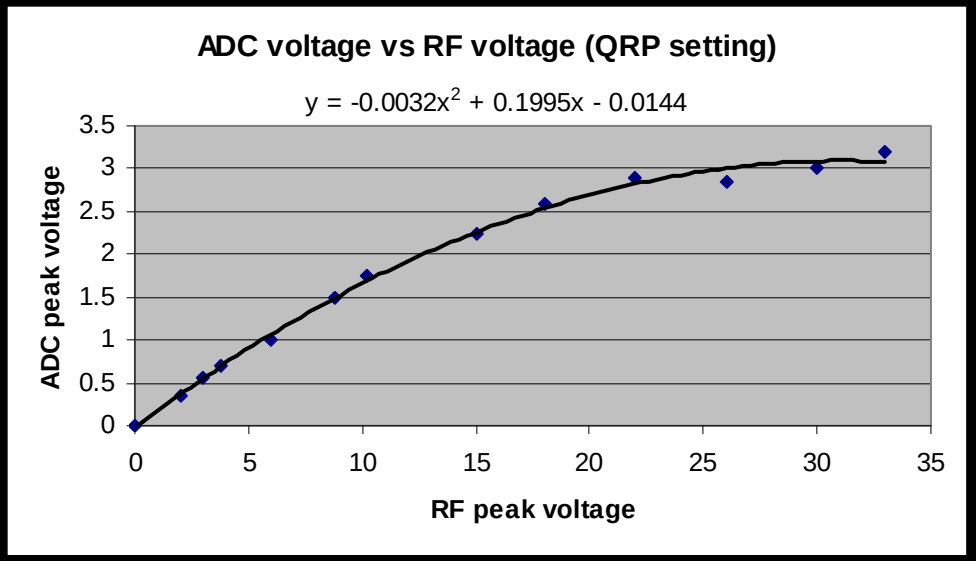
Maximum Power Selection



- PSKMeter is capable of controlling the power output to a specified level
- From $P=E^2/2R = E^2/100$, we know the peak RF voltage given an average power level (i.e., E=10 sqrt(P)).
- But what is the relationship between the peak voltage seen at the ADC to the peak RF voltage?

ADC voltage vs RF voltage

- Shown below is a measurement of the signal at the ADC as a function of the RF peak voltage:

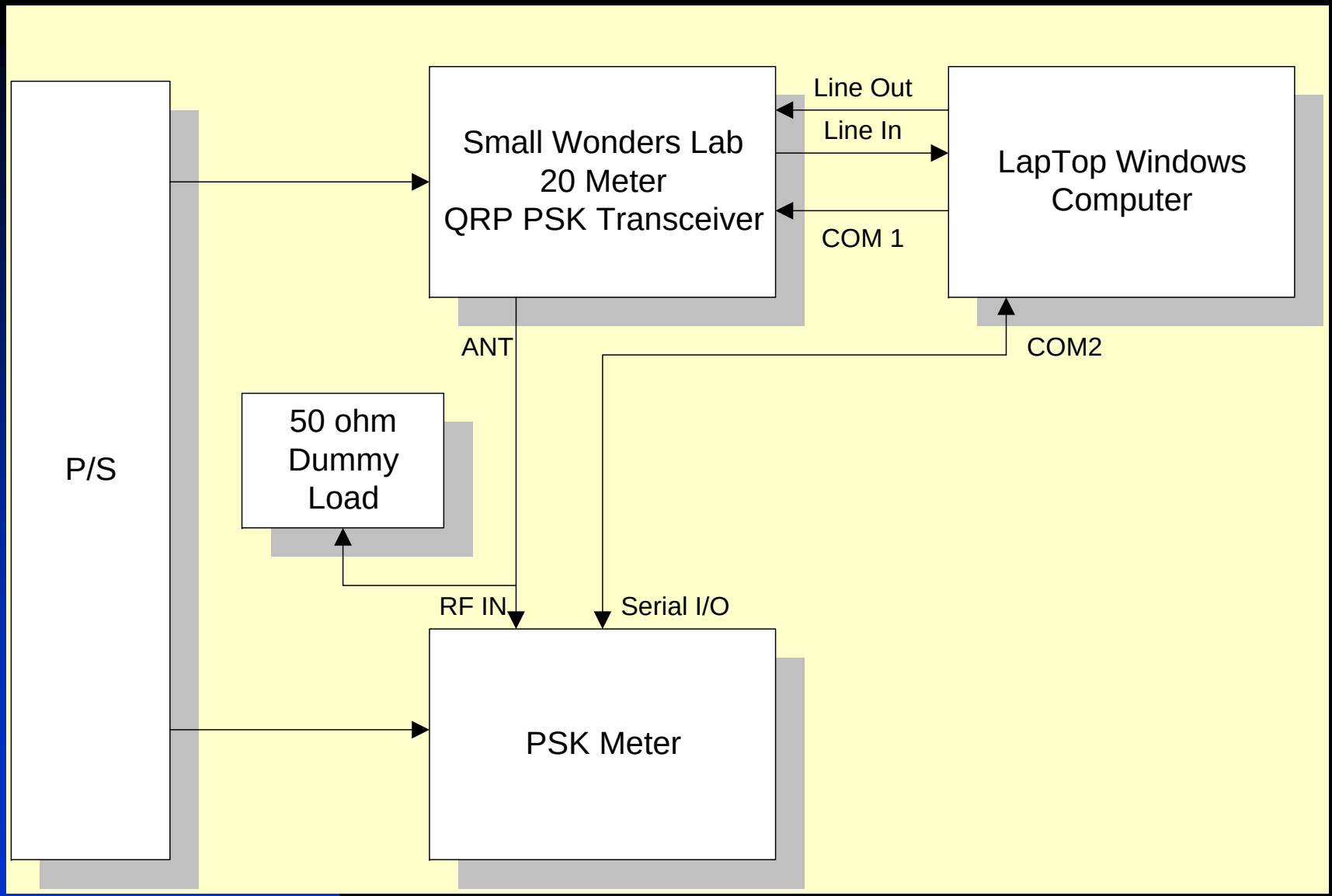


- Notice that the relationship is linear only for low power. The function is well approximated by a cubic polynomial over a wide range of power.

Maximum Power Selection

- Thus, given a desired power level, the software computes the RF peak voltage, then using this polynomial fit to the empirical data determines the voltage at the input to the ADC.
- The software then sets the audio level to obtain this voltage, thereby achieving the desired radiated power.

Demonstration Time!



Building one for yourself

- Circuit diagram, parts list, assembly instructions, pskmeter.exe and firmware hex file are available at no charge at http://www.ssiserver.com/info/pskmeter.
- Alternatively, PSKMeter is available in kit form; ordering can be done via a link at the same web site page shown above (or, I have some kits with me today).
- Questions and comments can be sent to george@softsci.com.

Future Enhancements

- Use one of the RS232 control lines to power the circuit—eliminates the need of a power adapter (wall wart)(adds dependencies on specific com port implementations—will it work for all computers and laptops?)
- Use an op-amp IC with diodes to obtain precision rectification—increases accuracy of the IMD calculation (714 lacks bandwidth, LM318 is fussy and tends to oscillate; high speed op-amp will add to cost)

Future Enhancements

- Port the existing pskmeter.exe application to a DLL (with a published API), to be accessed at first by Digipan and MixW and would allow Digipan and MixW to embed and incorporate PSKMeter functionality.
- Modify hardware to pass the PTT control signal to the rig interface--only a single COM port would be required. PSKMeter would be “in line” between the computer and the rig interface. Also, PSKMeter would know when the transmitter is keyed up eliminating the need to poll the hardware when in receive mode.

Future Enhancements

- Include rig interface circuitry on the PSKMeter printed circuit board, further reducing the complexity to the user (and the need for a separate rig interface) (adds parts and costs, but would be optional)

Where to go for more info

www.ssiserver.com\info\pskmeter. Download a copy of this presentation in PDF format (starting Monday), assembly instructions, circuit diagrams, firmware and software, software upgrades, order PSKMeter kits, read FAQs, etc.

Attend Steve Ford's WB8IMY talk "Introduction to PSK31" tomorrow at 1pm in Windor I.

Wireless Digital Communications: Design and Theory, Tom McDermott, N5EG

www.arrl.org (members only, search for PSK31)

www.psk31.com

See you on 14.070 PSK-31!

You've been a great audience!

73's from
George, KF6VSG