

PROJECTS THEORY APPLICATIONS CIRCUITS TECHNOLOGY

NUTS AND VOLTS

# NUTS AND VOLTS

www.nutsvolts.com  
April 2017

EVERYTHING FOR ELECTRONICS

A 50'S  
FERRIS  
WHEEL  
GETS AN  
ARDUINO  
UPGRADE



Sync A Digital Oscillator  
To An Analog Source

Build The Old School Cool CMOS Clock

Test Drive The PWI-928 RF Data Module





# PHANTOM

## X4 CHARGER



— CHARGES —  
DJI PHANTOM™ 3 / 4  
— SMART BATTERIES\* —

— FAST CHARGES —  
FOUR BATTERIES  
— SIMULTANEOUSLY —

CHARGES  
2 REMOTE CONTROLLERS  
TABLET DEVICES  
WHILE BATTERIES ARE CHARGING

# STAY CHARGED!

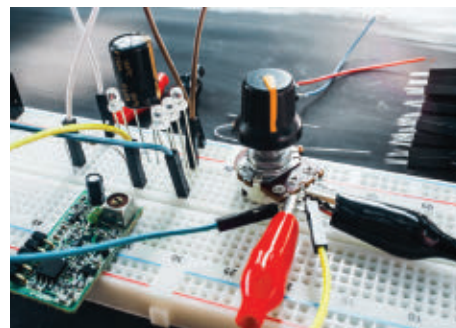


Charging your batteries shouldn't keep you away from what you really want to do: FLY!!!  
Our Phantom X4 Charger is the ultimate four-channel battery charger designed and built exclusively for charging, storing and deep cycling four DJI Phantom™ 3 and DJI Phantom™ 4 \* Smart Batteries simultaneously. And while your batteries are charging, plug into the USB ports and charge two remote controllers and two smart devices as well. That's right DJI Phantom™ Phans, stay charged and keep on flying!

\*DJI™ Phantom 4 Smart Batteries Require an Adaptor which is Sold Separately. ™DJI, DJI Phantom are trademarks of SX DJI Technology Co., Ltd.



## IF YOU BOUGHT AN ELECTROLYTIC OR FILM CAPACITOR DIRECTLY FROM CERTAIN DISTRIBUTORS SINCE 2002 YOU COULD GET MONEY FROM SETTLEMENTS TOTALING APPROXIMATELY \$15 MILLION



NEC TOKIN Corp. and NEC TOKIN America, Inc. (“NEC TOKIN”); Okaya Electric Industries Co., Ltd. (“OEI”); and Nitsuko Electronics Corporation (“Nitsuko”) (collectively “Settling Defendants”) have agreed to Settlements resolving claims that they allegedly fixed the prices of Capacitors. This may have caused individuals and businesses to pay more for Capacitors. Capacitors are electronic components that store electric charges between one or more pairs of conductors separated by an insulator.

### AM I INCLUDED?

You may be included if, from January 1, 2002, through July 15, 2016, you purchased one or more Capacitors from a distributor (or from an entity other than a Defendant) that a Defendant or alleged co-conspirator manufactured. “Indirect,” as that term is used below, means that you bought the product from someone other than the manufacturer, for example, from a distributor. A more detailed notice, including the exact Class definitions and exceptions to Class membership, is available at [www.capacitorsindirectcase.com](http://www.capacitorsindirectcase.com).

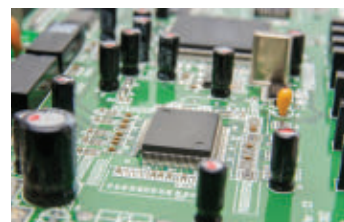
### WHAT DO THE SETTLEMENTS PROVIDE?

The Settlements provide for the combined payment of \$14,950,000 in cash to the Classes. The Settling Defendants have also agreed to cooperate in the pursuit of claims against other Defendants.

### HOW CAN I GET A PAYMENT?

Money will not be distributed to the Classes at this time. The lawyers for the Classes will pursue the lawsuit against the other Defendants to see if any future settlements or judgments can be obtained in the case and then be distributed together to reduce expenses. The lawyers anticipate that when the settlement proceeds are disbursed to the Classes, it will be done on a *pro rata* basis based on the value of your Capacitor purchases.

If you want to receive notice about the claims process or future settlements, you should register at [www.capacitorsindirectcase.com](http://www.capacitorsindirectcase.com).



### WHAT ARE MY RIGHTS?

Even if you do nothing, you will be bound by the Court’s decisions concerning these Settlements. If you want to keep your right to sue one or more of the Settling Defendants regarding Capacitor purchases, you must exclude yourself in writing from the Classes by May 31, 2017. If you stay in the Classes, you may object in writing to the Settlements by May 31, 2017. The Settlement Agreements, along with details on how to exclude yourself or object, are available at [www.capacitorsindirectcase.com](http://www.capacitorsindirectcase.com). The U.S. District Court for the Northern District of California will hold a hearing on July 6, 2017, at 10:00 a.m., at 450 Golden Gate Avenue, 19th Floor, Courtroom 11, San Francisco, CA 94102 to consider whether to approve the Settlements. Class Counsel may also request at the hearing, or at a later date, attorneys’ fees of up to 25% of the Settlement Funds, plus reimbursement of costs and expenses, for investigating the facts, litigating the case, negotiating the Settlements, providing notice to the Classes, and/or claims administration. The total amount of these costs shall be no more than \$2,558,454.00. You or your own lawyer may appear and speak at the hearing at your own expense, but you don’t have to. The hearing may be moved to a different date or time without additional notice, so it is a good idea to check the above-noted website for additional information. Please do not contact the Court about this case.

If the case against the other Defendants is not dismissed or settled, Class Counsel will have to prove their claims against the other Defendants at trial. Dates for the trial have not yet been set. The Court has appointed the law firm of Cotchett, Pitre & McCarthy, LLP to represent Indirect Purchaser Class members.

**FOR MORE INFORMATION: 1-866-217-4245 | [WWW.CAPACITORSINDIRECTCASE.COM](http://WWW.CAPACITORSINDIRECTCASE.COM)**

## 16 An Old-School Digital Clock

This timely clock project uses CMOS logic and seven-segment displays, which offer the builder many design variations for construction.

■ By Bryant Julstrom

## 23 Computer Control and Interfacing with the NI MyRIO

**Part 5: A Temperature Control System and Data Logger.**

We'll take everything we've done so far in this series on using National Instruments' MyRIO (a real time embedded evaluation board) and put together a complete system for temperature control.

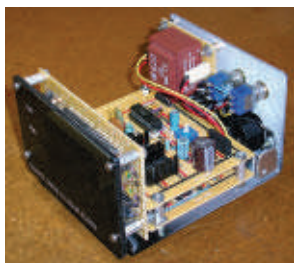
■ By David Ward

## 29 Take a CAN Bus for a Spin

**Part 3: CAN Controller Interrupts and Flags Give Us an Edge.**

This final article explains how to use the interrupts and flags available in an MCP2515 CAN controller IC, and describes how device-to-device acknowledgments work on the bus. The acknowledgment bit in a CAN frame plays a key part in successful communications and identification of errors.

■ By Jon Titus KZ1G



## 35 The Magic of the Gilbert Erector Set

Erector sets helped to spark lifetime passions for building things. Take a trip down memory lane and follow the construction of a classic Ferris wheel with some new bells and whistles added from today's technology.

■ By David Goodsell

## 40 Synchronizing a 60 Hz Crystal Controlled Oscillator to the Line

Though the science of synchronizing one analog signal to other analog signal is somewhat well-trod ground, the introduction of digital systems has created new and surprisingly interesting challenges for synchronization. Here is just one of many possible ways to solve this particular problem.

■ By Dana Geiger

## 46 Translational Reality Interfaces — Part 2: Control the World by Playing Pong

This article builds on "Translational Reality Interfaces" (N&V, March 2017) in which a toaster equipped with a translational reality interface (TRI) enables a user to control an irrigation pump. This time, I'll show you how a classic video game can be adopted for use as a TRI to enable the player to monitor and operate a remote device by simply playing the game.

■ By Bryan Bergeron

## 06 Q&A

### Reader Questions Answered Here

Asked and answered this month are topics on modernizing an old radio's power and monitoring a tankless hot water heater.

## 10 Near Space

### Approaching the Final Frontier

**A BalloonSat Photometer Flight Computer**

An explanation on how to create a recording photometer for BalloonSats that allows near space missions to record the colors of the sky and sunlight in eight different frequencies of the electromagnetic spectrum simultaneously.

## 51 Open Communication

### The Latest in Networking and Wireless Technologies

**Personal Radio Technologies**

Cell phones aren't the only option for communication with your family and friends. Here are some choices you may not have thought of or knew about.



## 54 The Design Cycle

### Advanced Techniques for Design Engineers

**Get "Patriot"ic with Lemos' LoRa Radio Module**

If you need a long-range RF data link that's easy to set up and use, you may be interested in this month's installment of Design Cycle. We'll be turning the knobs and pushing the buttons on a new custom-engineered LoRa radio module from Lemos International.



## Departments

### 05 DEVELOPING PERSPECTIVES

*My First Time Machine*

### 14 NEW PRODUCTS

### 15 SHOWCASE

### 60 NV WEBSTORE

### 62 TECH FORUM

### 64 CLASSIFIEDS

### 64 ELECTRO-NET

### 64 AD INDEX



Published Monthly By  
**T & L Publications, Inc.**

430 Princeland Ct.  
Corona, CA 92879-1300  
**(951) 371-8497**

FAX **(951) 371-3052**

Webstore orders only **1-800-783-4624**

**www.nutsvolts.com**

## Subscription Orders

Toll Free **1-877-525-2539**

Outside US **1-818-487-4545**

P.O. Box 15277

North Hollywood, CA 91615

## FOUNDER

Jack Lemieux

## PUBLISHER

Larry Lemieux

**publisher@nutsvolts.com**

## ASSOCIATE PUBLISHER/ ADVERTISING SALES

Robin Lemieux

**robin@nutsvolts.com**

## EDITOR

Bryan Bergeron

**techedit-nutsvolts@yahoo.com**

## VP OF OPERATIONS

Vern Graner

**vern@nutsvolts.com**

## CONTRIBUTING EDITORS

Fred Eady

Lou Frenzel

David Ward

Jon Titus

Bryant Julstrom

Kristen McIntyre

Paul Verhage

David Goodsell

Dana Geiger

## CIRCULATION DEPARTMENT

**subscribe@nutsvolts.com**

## SHOW COORDINATOR

Audrey Lemieux

## WEBSTORE MARKETING COVER GRAPHICS

Brian Kirkpatrick

**sales@nutsvolts.com**

## WEBSTORE MANAGER/ PRODUCTION

Sean Lemieux

**sean@nutsvolts.com**

## ADMINISTRATIVE STAFF

Re Gandara

Copyright © 2017 by T & L Publications, Inc.

All Rights Reserved

All advertising is subject to publisher's approval. We are not responsible for mistakes, misprints, or typographical errors. *Nuts & Volts Magazine* assumes no responsibility for the availability or condition of advertised items or for the honesty of the advertiser. The publisher makes no claims for the legality of any item advertised in *Nuts & Volts*. This is the sole responsibility of the advertiser. Advertisers and their agencies agree to indemnify and protect the publisher from any and all claims, action, or expense arising from advertising placed in *Nuts & Volts*. Please send all editorial correspondence, UPS, overnight mail, and artwork to: **430 Princeland Court, Corona, CA 92879**.

# DEVELOPING PERSPECTIVES

by  
**Bryan  
Bergeron,  
Editor**

## My First Time Machine – Well, at Least My First Flux Capacitor

**W**ith a circuit simulator and a basic knowledge of components and circuit theory, it's possible to simulate just about any electronic device that can be built. Trouble is, circuit simulators and the underlying models are inherently limiting. You're not likely to come up with a simulation of a time machine because your library of components is limited to the same resistors, capacitors, inductors, etc., that designers have used for decades.

The great thing about simulation is that you can define your own universe in software. Let's say you want to work with flux capacitors and time resistors. You define time resistors as delay lines of sorts — increase time resistance and time slows; decrease time resistance and time accelerates. Perhaps you define flux capacitors as gravity distortion devices. It's up to you, as long as the math is consistent across devices and within your universe. The math can be particular to your universe as well; perhaps division by zero always results in one, for example.

Why "waste" your time defining components that don't exist in the real world, and define a world that doesn't abide by our current understanding of physics? Because the exercise can free your mind to explore what would be required (for example) to build a real time machine, and what physical constraints of our world would have to be relaxed.

Furthermore, defining components in software that don't have a parallel in the real world doesn't mean that they can't be used as software components. For

example, many filter operations are performed by DSP techniques running on computer hardware, with the various filter components defined functionally in software.

Aside from defining a new universe complete with physical laws, a major challenge is getting buy-in from a large enough number of users to create a critical mass of contributors. It takes a following for technology to replicate and grow. Look at the parallels in computer software. There are dozens of language compilers introduced every year, and yet few attract enough followers from the C and C++ crowd to survive. The exceptions tend to be languages tied to specific microcontrollers or specific problem domains.

If you're new to circuit simulation, it's a good idea to get some experience working with the free software packages that are out there. Once you understand how circuit simulators work, you can try your hand with a generic simulation language, including something as simple as an Excel spreadsheet. You don't need fancy graphics and icons to develop your simulations. Better to focus on the math and interrelationships, and worry about aesthetics later on.

So, go ahead, define that flux capacitor. We're ready for someone to invent a time machine. **NV**

### ERRATA

In my March 2017 article on "A Totally Retro Mode Makes a Comeback," Figure 3 should be Figure 4 and vice versa (my error ... sorry).

John Thompson  
K3MD

In this column, Kristen answers questions about all aspects of electronics, including computer hardware, software, circuits, electronic theory, troubleshooting, and anything else of interest to the hobbyist. Feel free to participate with your questions, comments, or suggestions. **Send all questions and comments to: Q&A@nutsvolts.com.**

## Modernizing an Old Radio's Power

**Q** I am an 85 year old electronics repair person. My expertise, however, is in vacuum tube equipment. I belong to an antique and classic car club and work on the old radios for the members. These old radios use a vibrator to chop up the DC in order to step it up for the tubes; the vibrators are no longer available. I have repaired several of them and got them working again. However, a lot of them are beyond repair.

I was wondering if you could furnish me with a circuit for a 1,500 cycle oscillator using a 555 or 556 that I could put in the old vibrator can and drive the transformer; solid-state and ICs are all new to me. Any help would be greatly appreciated.

**Dave Freeman**

**A** I remember working on old radios with vibrators back in the 1960s. The contacts would often be pitted and occasionally fuse together. While they are an amusing way to generate the high voltages needed to bias tube plates, they always seemed problematic to me. Fortunately, while we have to cheat a bit with modern semiconductors, we can fix this problem.

A vibrator generates the AC (or more precisely, the changing magnetic flux) needed to be able to use a transformer to increase something like a car's 12V input to something that can bias a tube. Typically, that's at least a few hundred volts. Since there were no practical semiconductors at the time when most of these radios were made, engineers used mechanical oscillators to create square wave inputs by alternately grounding the primary inputs to a step-up transformer, connecting the center tap of the transformer to the power supply.

This allowed the transformer to see alternately a magnetic field of one polarity, followed by a switch to the other polarity, as each side of the transformer primary was connected to the so-called ground side of the supply. In cars, typically the negative battery terminal is connected to the chassis of the vehicle, and we call this "ground." Sometimes it's reversed, though, with the positive terminal

## • Modernizing an Old Radio's Power

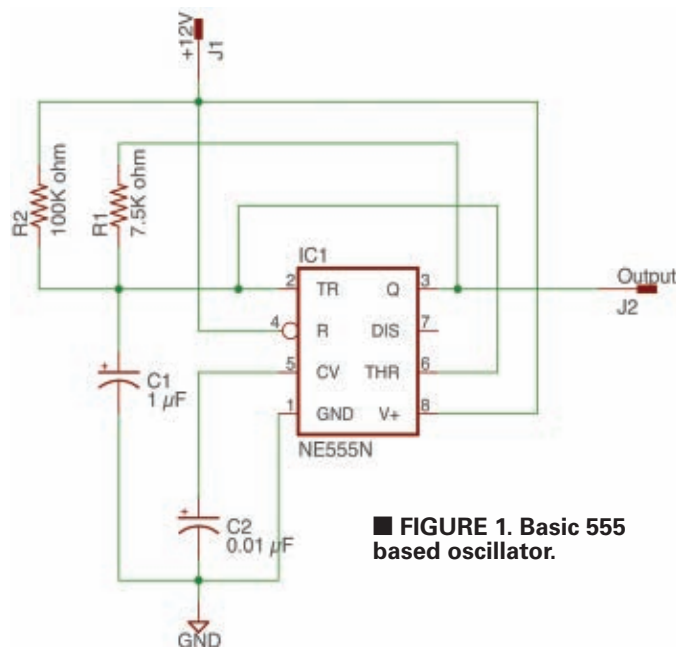
## • Monitoring a Tankless Hot Water Heater

connected to the chassis. This is an example of why the whole concept of "ground" is muddled, in my opinion.

We don't care too much about the integrity of the input waveform in this application. The transformer's magnetic flux — which is proportional to the current — will look like a triangle wave, as the primary integrates the input voltage square waves, according to the voltage and current relationship for an inductor. There will be some harmonic energy at higher frequencies from this kind of waveform and subsequent losses in the core of the transformer, but they are minimal for this purpose.

The vibrator itself is built much like a relay, with an electromagnet that pulls the contacts back and forth. The contacts themselves are used to create positive feedback and make the relay-like device oscillate at its natural frequency. We'd like to eliminate that kind of a system and replace it with a 555 oscillator driving some kind of transistor switches. In this case, we'll only oscillate the low voltage and current portion of the circuit, leaving the heavy lifting for a pair of output devices. I'm going to assume a negative ground system, but this circuit could easily be inverted for positive ground.

One thing that is a bit confusing to me in your



■ **FIGURE 1.** Basic 555 based oscillator.



Post comments on this article and find any associated files and/or downloads at  
[www.nutsvolts.com/magazine/article/April2017\\_QA](http://www.nutsvolts.com/magazine/article/April2017_QA).

$$f(\text{Hz}) = \frac{1}{0.693(2R_2)C}$$

## ■ FIGURE 2. Frequency equation for simple 555 oscillator.

question is 1,500 cycles. I'm fairly sure that most vibrators oscillate at around 100 cycles per second, or Hz in modern units. I think the magnetic material in those transformers is laminated iron, which won't be too happy at that frequency. I'm going to attempt a design that runs at 100 Hz, but it could easily be tweaked to run at a higher frequency.

Let's start with the oscillator, using a 555 (**Figure 1**). This is a very simple design cribbed from the application notes. We want the output square wave to be symmetric, so no special provisions are needed. We use the easy equation for a 50% duty cycle oscillator to calculate some R and C values (**Figure 2**). Since a 555 can be powered from a variable power supply voltage that nicely includes most 12V (really 13.8V at full charge) automobile batteries, we can just put a filter capacitor in and use the input voltage as-is.

For the switching transistors, we will need an inversion so that we can switch one on while the other is off. We'll use a simple bipolar transistor for that. In order to handle significant current and switch quickly, we can use a pair of N-channel MOSFETs as our switch devices. I think NPN bipolar transistors could work here as well, but MOSFETs, with their high input impedance at the gates will ensure that the devices switch on fully and quickly with minimal

drive. **Figure 3** shows an attempt to design the entire circuit.

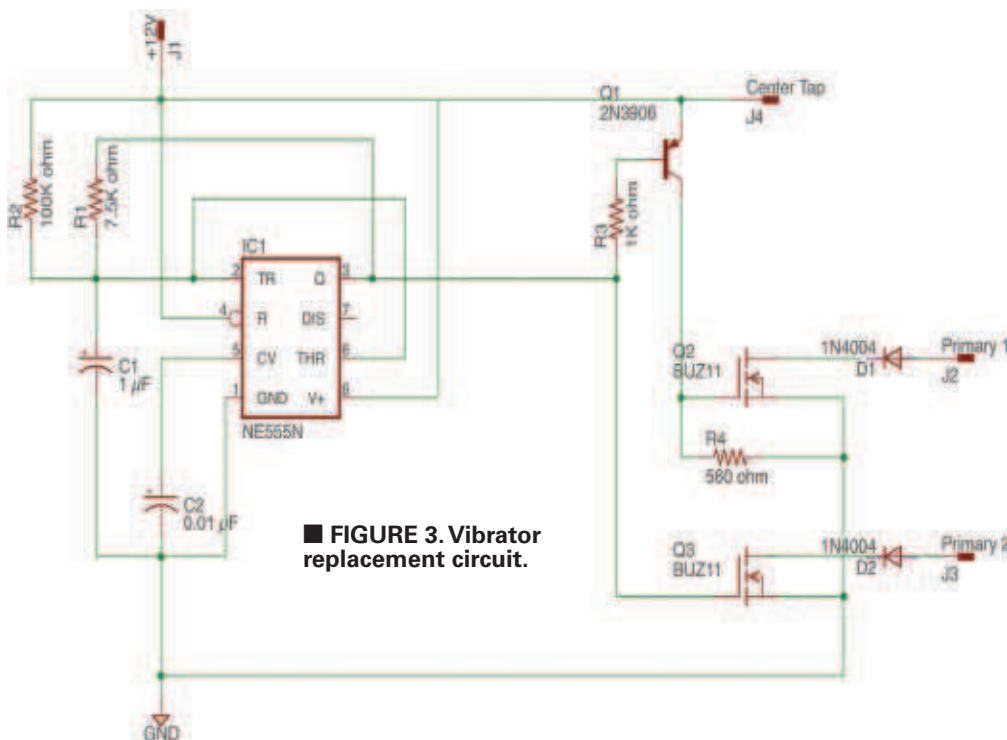
Please note that I have not tested this circuit, so some refinement may be necessary. It might even be wrong! I think it's close, though. Also, this will switch much more quickly than the original vibrator, so the output voltage on the transformer is likely to be a little high because of the longer time that the voltage will be applied to the primary. This will give it more time to integrate, and thus more change in magnetic flux. Dropping the primary voltage a bit with a diode or two will bring that back down. I put one set of diodes (D1 and D2) in there for good measure.

## Monitoring a Tankless Hot Water Heater

**Q** I am looking for a simple circuit to monitor all the normally closed and normally open controls on my hydronic heating system with a tankless hot water heater. I want a separate green LED to light when the control is in the normal mode, and an intermittent sound and red blinking LED to light when a control opens in a trouble mode and then stay on until it is manually reset. One control is a single-pole double-throw device with a normally open and normally closed contact, depending on whether it is in the heating or the hot water mode.

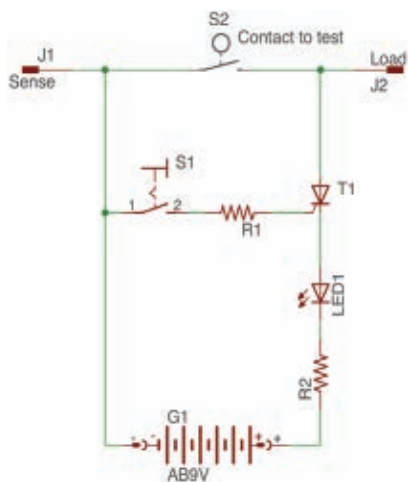
I have a device which is battery operated that I place across the contacts of a normally open control switch in a 24 volt AC circuit to monitor it. If the switch malfunctions, an SCR will lock out and light a red LED. There is a button to reset it back to the monitor mode. I can only use the device for a single control.

**George Forcino**  
Cranston, RI



■ FIGURE 3. Vibrator replacement circuit.

**A** While I'm a big fan of the analog circuit approach to most designs, when it comes to complicated behavior, I tend to favor the modern approach of using inexpensive embedded computer boards, many of which have numerous inputs and outputs that can be exploited to monitor inputs and drive outputs in a multitude of ways. That being said, there are some analog solutions that



■ **FIGURE 4. George's latching indicator.**

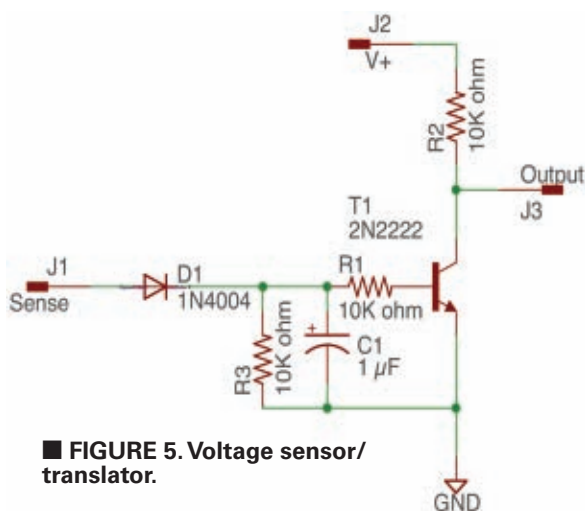
work for part of the problem.

I've reproduced your simple circuit based on an SCR that gives an indication that a particular relay is in the state you want, and that must be reset to turn the indicator off in **Figure 4**. It certainly gets the job done. I have one concern, though, and that is it does inject some voltage and current into the circuit that is being monitored. It's hard to tell whether that might damage (unlikely) or more likely modify the behavior of the circuit that it is monitoring. It also might not work for all things that need to be monitored.

The real key to getting this to work (I think) is to be able to sense those states in the most unobtrusive way possible, and convert the states of the contacts into high and low indications that can be read by either an embedded controller or some more complicated digital logic. If you're lucky, you can get all of the states to come together as the logical AND or OR of the needed states, in which case an analog solution will work. This is probably what most modern implementations do to be able to report status.

As I said earlier, though, this particular problem seems to be best suited for a hybrid analog and embedded computer solution. **Figure 5** shows a simple circuit that could be used to monitor for an AC voltage developed at a node in the circuit with a contact that might open or close (on one side or the other) and have a voltage present there. I believe many of these heaters have a 24 VAC system, but it might be higher or lower in your heater. The voltages should be carefully checked, and please watch out for full line voltage at points in the circuit. What is shown should work over a wide range of voltages, though, and be able to survive.

Sensing closure of a contact through injecting current is a trickier business. It really depends on where the contacts are in the circuit and what's going on. **Figure 6** gives you a general idea of how one might do this for something that is ultimately connected to something near the chassis ground when the contact is closed, but how



■ **FIGURE 5. Voltage sensor/translator.**

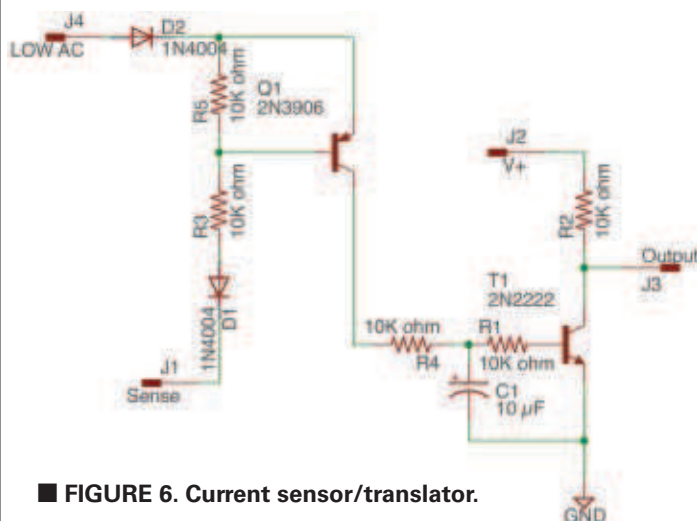
well it works when the contact is open is entirely dependent upon what's on either side of the contacts when they are open. It requires a high impedance on one of the sides to be able to sense closure.

It actually works in similar fashion to your circuit with the SCR, but ultimately results in a DC control signal referenced to the chassis ground that can be at any voltage (like +5V), and scavenges voltage from the 24 VAC power supply internal to the device. You can invert the sense part of this circuit on the

left side to get it to look for current flow toward the LOW AC side (by which I mean the low VAC supply – perhaps 24V – terminal that's not connected to the chassis) as well. Both supplies (V+ and LOW AC) should be referenced to the same chassis ground.

Once the control signals are assembled, they can be sent to the PIO pins of an Arduino, and then some very simple code can be written to handle giving it a sticky state, requiring a reset button, flashing lights, sounding buzzers, or anything else you desire. Doing this detection in software will also give you an opportunity to debounce contacts, handle reset conditions, avoid false triggering with transient states, and add things like timeouts.

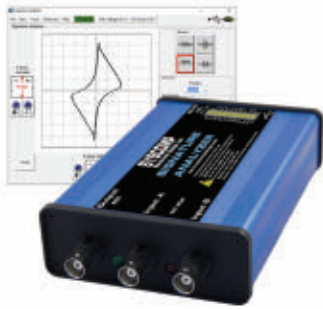
Again, though, I'd like to emphasize that sensing this internal state of an unknown device without provisions for doing that can be difficult. There is no general solution, so carefully studying the schematics and measuring voltage and impedance will be essential to getting a good monitoring system working. **NV**



■ **FIGURE 6. Current sensor/translator.**



## More Unique Products



### Signature Analyzer

- Controlled via USB
- 5-in-1 Functions
- Includes Oscilloscope AWG, I/O
- WIN, MacOS, Linux
- Under \$400

SIG-101

### Oscilloscope, Waveform Generator, Digital I/O Port - All in One!



- Terrific educational tool with lessons/experiments
- AWG, I/O, PWM, FFT
- \$99.95

CGM-101

### SDS1102X Scope Free Serial Decode Option



- 1GSa/s Sampling
- 14 Mpts Memory
- FFT
- 8" LCD
- 60,000 wfm/s
- \$424

### Wide Selection of Spectrum Analyzers



- Desktop to USB stick versions cover up to 12GHz to find signal qualities or interference
- Starting at \$518



### Power Supplies - 30W to 1500W



- Desktop to rack-mount power supplies to fit almost any need.
- Programmable on/off voltage level cycles
- From \$56

### Automotive Diagnostic Equipment



- Picoscope test kits are designed to make diagnosing & testing vehicle electronics, components easy. Find faults fast!

### EMC Precompliance Testing



- Check your PCB design before expensive formal testing to make sure you'll pass EMC specs with our affordable solutions.

### PCB Repair Equipment



- Repair, don't throw away costly PCBs. BoardMaster systems are easy-to-use and can be used by anyone to find and repair PCB faults.

# A BalloonSat Photometer Flight Computer

**I've flown a few photometers into near space on BalloonSats (model satellites carried by weather balloons) in the past. They used LEDs as their sensing element because as Forrest Mims discovered in 1992, LEDs behave as color specific light detectors. On each mission, I switched out the LED in order to determine how the intensity of the new color varies in near space. However, switching out LEDs between flights prevents me from comparing multiple colors simultaneously. So, this month, I'll explain how I created a recording photometer for BalloonSats that allows near space missions to record the colors of the sky and sunlight in eight different frequencies of the electromagnetic spectrum simultaneously.**

## LED Photometers

If you've read some of my past articles, you'll know that I love the work Forrest Mims has done using LEDs as color-specific photometers. For readers unfamiliar with this, Mims wrote an article titled "Sun Photometer with Light-Emitting Diodes as Spectrally Selective Filters" in 1992.

The article described how LEDs can be used as the detector in a narrow-band photometer. What Mims discovered is that not only does an LED emit a specific color of light, it also detects the amount of the same color of light shining on it.

You see, when electrons jump

across the doped PN junction of an LED, the electron combines with a hole on the other side to produce a photon of light. The energy and therefore the frequency and color of the light depend on the energy barrier

formed by the creation of the PN junction. Solar cells and photodiodes have similar PN junctions, but behave in the opposite way.

When a photon of light shines on their PN junctions, it frees an electron (and makes a hole) with enough energy to jump across the PN junction and create a current. Mims discovered that the LEDs behave like photodiodes and solar cells when light shines on them.

The only difference is that photodiodes and solar cells respond to a wide range of photon wavelengths while LEDs are very particular about the photons of light they'll make a current for.

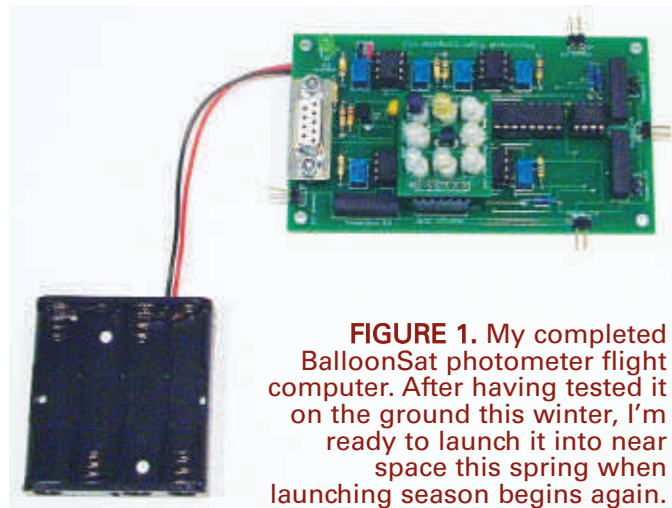
Therefore, an LED that emits

green light at 500 nanometers (nm) only produces a current when light of around 500 nm shines on it. Even better, the amount of current is directly proportional to the intensity of light, or the number of green photons shining on the LED's PN junction per second.

It's so much more affordable to use LEDs in a photometer than the expensive narrow band filters used in professional photometers (although the increased cost of a narrow band photometer does result in better resolution in the data).

The only problem remaining is that microcontrollers can only measure voltages (if even that). Current — which is produced by an LED — is something that needs to be converted into voltage prior to digitization by the microcontroller. Operational amplifiers (op-amps) are one way to make the conversion. The circuit designed to do this is called a transimpedance amplifier.

Current entering the inverting input of the op-amp is converted into a voltage equal to the current times the feedback resistor (recall that Ohm's Law says  $V = IR$ ). In the case of an LED photometer, the current



**FIGURE 1.** My completed BalloonSat photometer flight computer. After having tested it on the ground this winter, I'm ready to launch it into near space this spring when launching season begins again.



Post comments on this article and find any associated files and/or downloads at [www.nutsvolts.com/magazine/article/April2017\\_NearSpace\\_BalloonSat-Photometer-Flight-Computer](http://www.nutsvolts.com/magazine/article/April2017_NearSpace_BalloonSat-Photometer-Flight-Computer).

source is an LED and it's the cathode that's connected to the inverting input of an op-amp.

In the case of the LED photometer I am using, the LED is not biased. In other words, no voltage is applied to it. This is called photovoltaic mode and is the same situation occurring in a solar cell.

The op-amp represents a low impedance load for the LED and it can therefore create a current that is proportional to the light intensity incident on its PN junction. The capacitor that's parallel to the feedback resistor keeps the op-amp stable and prevents its output voltage from oscillating.

The value for the feedback resistor depends on the desired output voltage and the amount of current generated by the current source.

In my case, I experiment with different values to find what works well. I expect that the proper resistor values for each LED will necessarily change as the intensity of the LEDs I am using is changed.

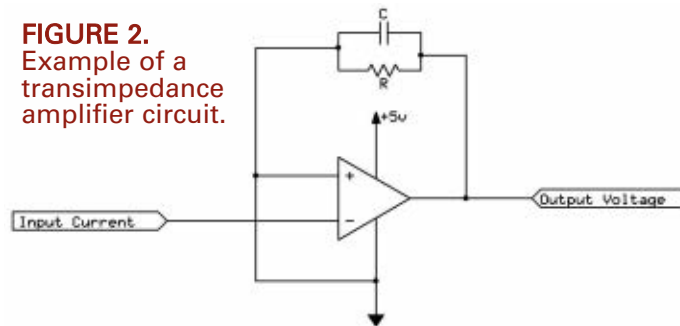
## Putting It All Together

So, all I have to do is create a printed circuit board (PCB) that supports multiple LEDs, a transimpedance amplifier for each LED, and a microcontroller to digitize and record the output from each amplifier. Right? Not quite.

The output of LEDs is also affected by their temperature. Therefore, I need to add a suitable temperature sensor to the circuit and surround it with the LEDs in order to measure their temperatures.

Since this is designed as a flight computer for BalloonSats, I further expanded the circuit with

**FIGURE 2.** Example of a transimpedance amplifier circuit.



two relays to run cameras and an I<sup>2</sup>C memory for a datalogger. Finally, I placed the LEDs and temperature sensor in a separate PCB that plugs into the main board. That way, the main electronics can remain inside the warm airframe while the LEDs are exposed to the cold unfiltered sunlight. **Figure 3** shows the circuit I came up with.

The components I used were:

C1 (C2): 220 pF  
C3: 22  $\mu$ F  
D1: LED  
R1: 4K7

R2 (R3): Value depends on the LED  
R4: 10K  
R5: 22K  
R6: 1K  
R7: 4K7  
R8: 4K7  
R9: 680 ohms  
RL1 (RL2 and RL3): Reed Relays  
U1: TLC272 (Op-amp)  
U2: LM2940 (LDO Voltage

Regulator)

U3: 24LC128 (I<sup>2</sup>C Memory)

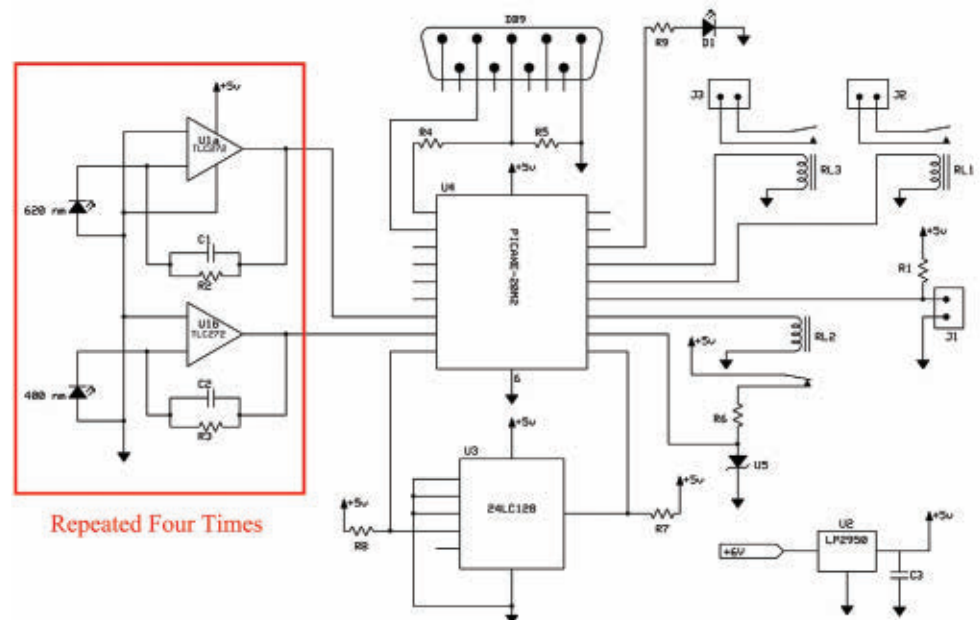
U4: PICAXE-20M2 (Programmable Microcontroller)

U5: LM335 (Temperature Sensor)

## Some Construction Notes

**D1** is an indicator. The PICAXE illuminates it to signal that it's collecting data.

**J1** is the Commit Port. The PICAXE checks to see if this port is shorted to ground (with a two-pin shorting block). As long as it is, the



Repeated Four Times

**FIGURE 3.** It took two iterations to get it right, but I finally came up with this circuit. The result is a BalloonSat flight computer that operates two cameras and records the output from eight different LEDs and their temperature. *Nuts & Volts* readers are free to build their own recording LED photometer for themselves.

LED Color	Wavelength	Resistance
IR	940 nm	100K
IR	890 nm	100K
Red	660 nm	1M
Orange	620 nm	1M
Yellow	590 nm	1M
Green	500 nm	1M
Blue	470 nm	10M
Violet/UV	400 nm	10M

**Table 1.**

PICAXE will not start collecting data or triggering the cameras.

**J2** and **J3** are Camera Ports. Two modified cameras (wires soldered to their shutter buttons) plug into these ports. The PICAXE triggers the cameras to take pictures regularly during a mission by using the HIGH command.

**RL2** controls power to the temperature sensor (**U5**). The LM335 generates heat, and I've noticed its temperature is greater than the air temperature at high altitudes (where there's very little air). This relay is an experiment to see if shutting down the LM335 between readings will improve its accuracy in near vacuum conditions.

There are eight photometer circuits in the flight computer, of which two are shown in the schematic. **Table 1** shows the recommended resistors for each

color of LED. Be warned, however, that your mileage may vary depending on the intensity and sensitivity of the LEDs you select.

Use the PICAXE's *READADC* and *READADC10* commands to digitize the output from the LEDs. Which command you use depends on the LED and its current output.

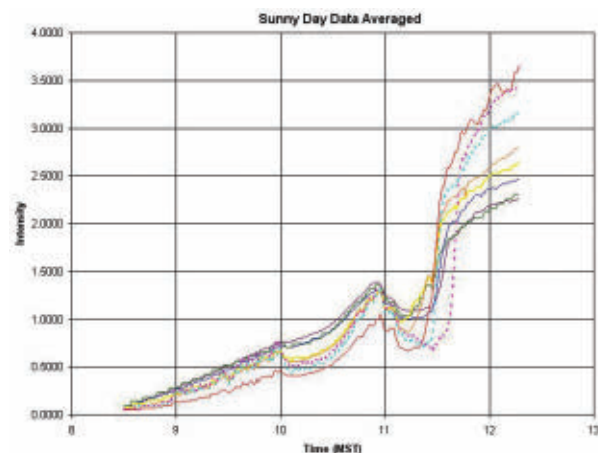
Experiment with this before trying to collect data on a near space mission.

## Results

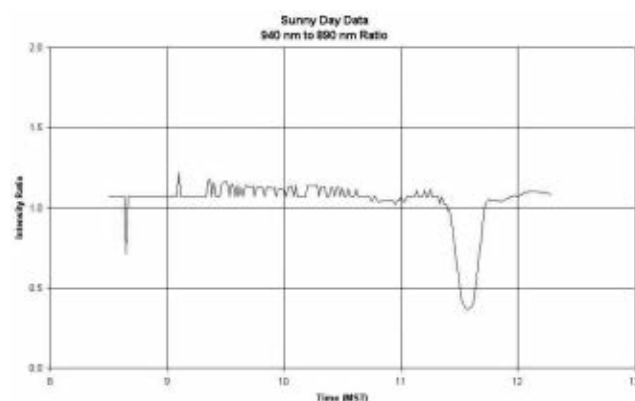
It's not near space season quite yet, so I've been testing the BalloonSat Photometer on my patio. The two graphs in **Figures 4** and **5** are the results of letting the photometer record data for several hours during the morning.

Because each LED responds differently to light intensity, I divided each LED reading by the average of all that LED's readings. This normalizes the curves to an extent and makes it easier to compare them to one another.

The thing to notice in **Figure 4** is not that the sky became brighter as the sun rose higher (as expected), but that it shows that clouds passed over the sun twice; first at between 10:00 and 10:30, and again at 11:00 to 11:30. The second cloud had a larger impact on sky brightness than the first.



**FIGURE 4.** This is photometer data collected from 8:30 AM to 12:20 PM.



**FIGURE 5.** Now we're looking at just the two infrared wavelengths. As mentioned by Mr. Mims, water vapor absorbs 940 nm IR more than it absorbs 890 nm IR.

**I'd like to thank Forrest Mims for helping me to understand how LED photometers work. He has always been a great encouragement for amateur scientists everywhere.**

More importantly, you'll notice that the 940 nm IR intensity lagged behind the 890 nm IR intensity during the second cloud passage. The next chart looks just at the ratio of 940 nm to 890 nm IR intensities.

Notice the large drop of 940 nm IR compared to 890 nm IR between 11:20 and 11:40 AM. Most likely, this happened because of the water vapor contained within the cloud.

Therefore, it appears that sunlight was shining through more water vapor in the cloud during this time because more 940 nm IR was absorbed than 890 nm IR. I don't know of a better way to detect water vapor in clouds than this.

Interestingly enough, this data was collected in mid-December when I would think most clouds are water ice and not water droplets. I wish I were better prepared for this result because I would have photographed the cloud in long-wave infrared to see if this cloud was significantly warmer compared to other clouds.

The BalloonSat photometer flight



computer has been a lot of fun to play with on the ground. I'd like to experiment just a little more with it on the ground and then again during the total solar eclipse on August 21<sup>st</sup>.

Between the start of launch season and the solar eclipse, I plan to fly the photometer with it pointing up and then pointing down. I expect to see a difference in the data between looking at light emitted by the sun and sky, and looking at light scattered by the atmosphere.

Onwards and Upwards,  
Your near space guide **NV**

## MEASUREMENT COMPUTING

### Wide Selection of Ethernet Devices for Remote Monitoring and Control

**NEW**



#### Multifunction DAQ

**E-1608 - Only \$499**

- Eight 16-bit analog inputs
- 250 kS/s sample rate
- 2 analog outputs
- 8 digital I/O, 1 counter

#### Thermocouple Input

**E-TC - Only \$479**

- 8 thermocouple inputs
- 24-bit resolution
- 4 S/s per channel sample rate
- 8 digital I/O

#### Digital I/O

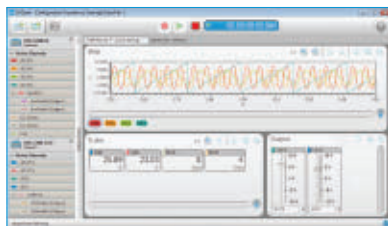
**E-DIO24 - Only \$289**

- 24-channel digital I/O
- $\pm 24$  mA high-drive capability
- Digital output alarm
- One 32-bit event counter

#### DAQ Companion Software

**DAQami™ 4.0 - Only \$49**

- Easy-to-use interface
- Acquire and generate data from analog, digital, and counter signals
- Export acquired data to a .csv file
- Try DAQami free for 30 Days



Ethernet devices also include comprehensive drivers and support for C++®, C#®, Visual Basic®, DASyLab®, and NI LabVIEW™

**MCCDAQ.COM**

**MEASUREMENT  
COMPUTING™**

**Contact us  
1.800.234.4232**

©2017 Measurement Computing Corporation, 10 Commerce Way, Norton, MA 02766 • info@mccdaq.com

# PoLabs

For more information or  
software download please visit



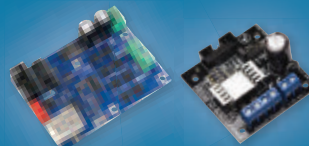
**www.poscope.com**

### PoKeys Connect, Control



- USB
- Ethernet
- Web server
- Modbus
- CNC (Mach3/4)
- IO
- PWM
- Encoders
- LCD
- Analog inputs
- Compact PLC

### Stepper motor drivers Drive



- up to 256 microsteps
- 50 V / 6 A
- USB configuration
- Isolated
- up to 32 microsteps
- 30 V / 2.5 A

### PoScope Mega1+ PoScope Mega50 Measure



- up to 50MS/s
- resolution up to 12bit
- Lowest power consumption
- Smallest and lightest
- 7 in 1: Oscilloscope, FFT, X/Y, Recorder, Logic Analyzer, Protocol decoder, Signal generator

# NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

## 4" OMNI WHEEL

**M**obilize a robot chassis with the Actobotics 4" omni wheel available from ServoCity. Omni wheels can be used just like regular drive wheels but have the advantage of being able to roll freely perpendicular to the drive direction.

The 10 rubber rollers around the circumference of the wheel provide excellent grip to excel in moving in the forward direction. Each roller is mounted on a stainless steel sleeve which rests on a stainless steel axle to allow the rollers to rotate freely and independently from one another. Multiple wheels can be mounted together by installing an 1/8" spacer between the wheels to provide adequate clearance between rollers.

The wheels can be clocked so that the rollers are staggered evenly which smooths out the circumference and increases load-bearing capabilities. The wheels have a 1/2" center hole, and numerous mounting options are provided including the standard 0.77" and 1.50" hub patterns for using with other Actobotics parts.

These wheels can reduce the

amount of torque necessary to turn while decreasing the turning radius of your chassis. Price is \$9.99.

## BALL BEARING WHEEL MOUNT

**T**his wheel adapter is intended to be used with ServoCity's five spoke pneumatic tire/wheel. This piece is machined to perfectly mate with the interior shape of the molded wheel so that the adapter and the wheel will turn as one.

The adapter houses a 1/2" ID ball bearing so that the assembly can be slid onto a fixed shaft and rotate freely on that shaft. The 1.5" Actobotics hub pattern is machined into the adapter so that a sprocket or gear can easily be attached to the outside of the adapter to drive the wheel assembly.

A single wheel adapter can be used and the other side of the wheel can be supported by one of the stock bearings included with the pneumatic tire/wheel or an adapter on each side can be utilized for a more symmetrical setup.

ServoCity recommends running a 1/2" shaft spacer next to the bearing installed into the adapter in order to

provide proper clearance between the bearing and the surface that it's running next to. Price is \$15.99.

## TAPPED WHEEL MOUNT

**T**his next wheel adapter from ServoCity is also intended to be used with their five spoke pneumatic tire/wheel. This piece is machined to perfectly mate with the interior shape of the molded wheel so the adapter and the wheel turn as one.

This adapter has the large 1.5" hub pattern with 1/4" thru-holes machined into the face so that ServoCity's 1" HD clamping hub can be attached using 1/4-20 low profile socket head screws.

By installing a clamping hub onto the adapter, the wheel assembly can be secured onto a 1" shaft. This style adapter is commonly used in conjunction with an axle that is mounted in bearings so that the axle and wheel can rotate in unison. Price is \$14.99.

For more information, contact:  
**ServoCity**  
[www.servocity.com](http://www.servocity.com)





## ULTRA-WIDEBAND POWER AMP

The Lemos Patriot PAT-HPA21G is a high performance, low noise, ultra-wideband amplifier. Based on GaN technology, this amplifier achieves high reliability and high efficiency in a very small form-factor. It's also lightweight which makes it suitable for a wide range of applications.

Most power amplifiers are divided into sub bands that cover:

- 0.1-18 GHz
- 0.5-18 GHz
- 2-18 GHz
- 6-18 GHz

The Lemos Patriot amplifier is quite unique since it has good NF and also a very high output power P1db of 24 dbm and high gain of 30 db. This unit can achieve even better specs if the customer doesn't need the entire frequency band of 0.8-21 GHz and only needs (for example) 6-18 GHz.

Features include:

- GaN Based
- Ultra-Wideband Operation
- Compact and Lightweight
- MIL-STD 810F
- Low Current 220 mA Max.

Applications include:

- Driver Amplifier
- Low Noise Amplifier in Receiving Chains
- Radar
- Ultra-Wideband Measurement Instruments
- Lab Use

Physical dimensions are 1.20 x 0.64 x 0.45 inches; weight is 17 grams. Contact Lemos for pricing.

For more information, contact:  
**Lemos International  
Co., Inc.**  
[www.lemosint.com](http://www.lemosint.com)

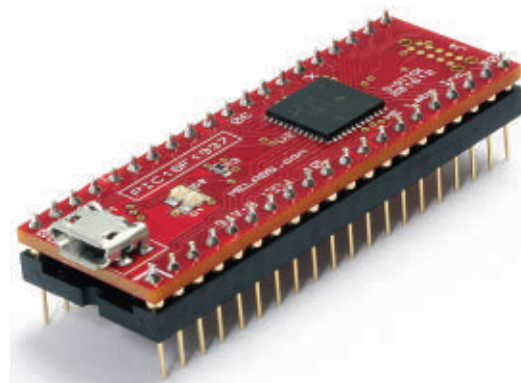


## STANDARD D-STICK

The ME Labs Standard D-Stick provides all the functionality of Microchip's 40-pin PIC16F1937 in a hardware module that includes a USB onboard programmer and virtual COM port. The D-Stick is a compact, simple, and easy to use alternative to connecting a serial port, programmer, power supply, etc., to a solderless breadboard for project development. After development, simply replace the D-Stick with the pinout compatible/production ready PIC16F1937.

When combined with the free PICBASIC PRO™ Compiler Student Edition, the Standard D-Stick becomes a comprehensive development system that includes a code editor, BASIC compiler, in-circuit debugger, and device programmer for under \$30.

- Pinout is identical to Microchip's standard 40-pin DIP.
- Round machined pins are easy on spring contacts, allowing for multiple insertion cycles.
- Built-in micro-USB port supplies power, a programming connection, and a virtual COM port.



- Suitable for serial in-circuit debugging.
- Compatible with all ME Labs Trainer programs.
- Standard version – based on PIC16F1937 compatible with the PBP Student Edition.
- Advanced Version – based on PIC18F compatible with PBP Gold Edition (sold separately and will be available soon).

For more information, contact:  
**ME Labs**  
[www.melabs.com](http://www.melabs.com)



# GUI MADE EASY

The arLCD 3.5" smart TFT touchscreen combines Arduino UNO with the ezLCD smart LCD.

[earthlcd.com/nuts](http://earthlcd.com/nuts)

**SDP** Stock Drive Products  
Setting Ideas Into Motion

One-Stop Shop for  
Mechatronic Components

EXPLORE  
DESIGN  
BUY ONLINE

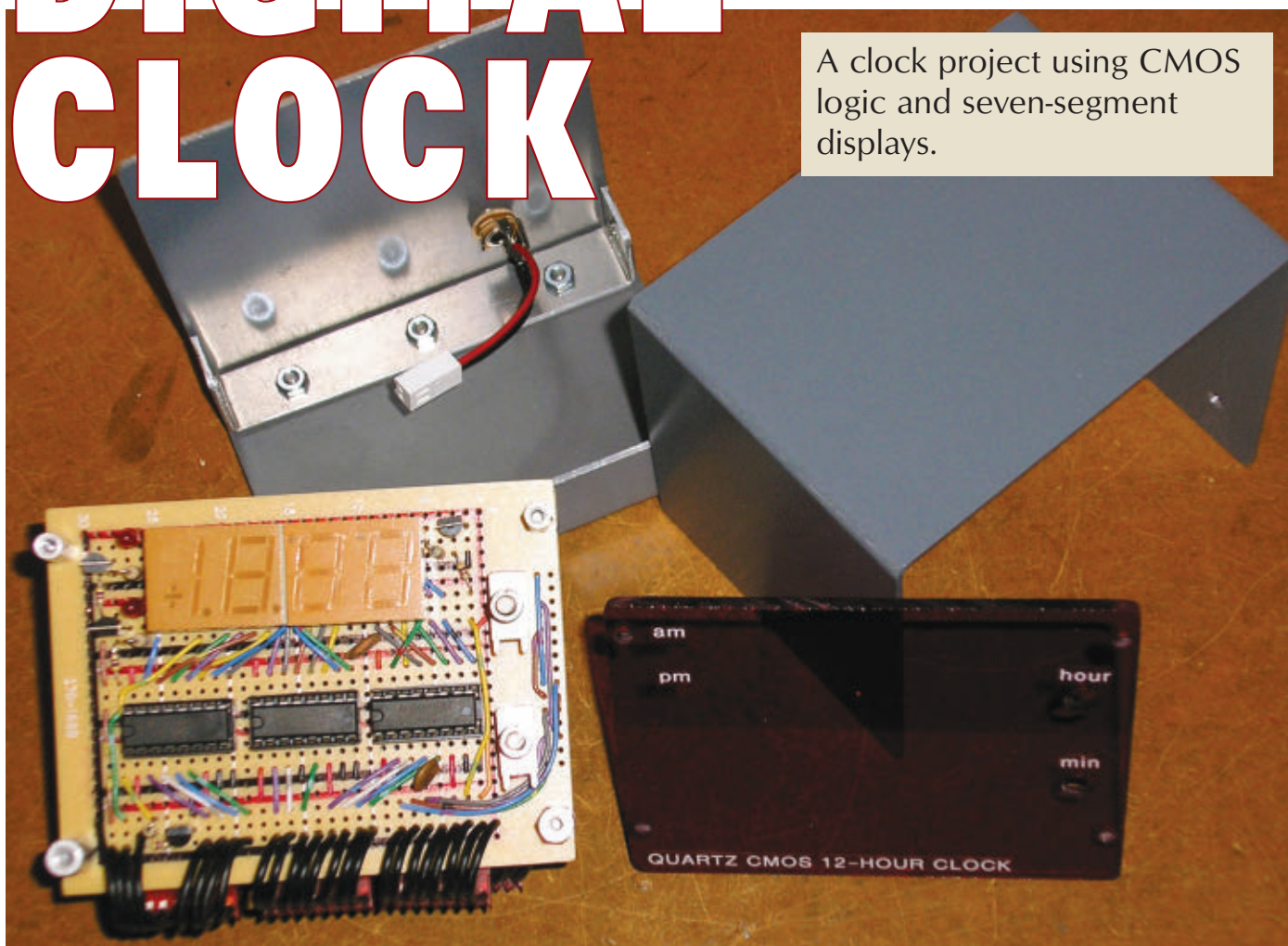
[www.sdp-si.com](http://www.sdp-si.com)  
no minimum requirement

**Designatronics** inc.

By Bryant Julstrom

# An Old-School DIGITAL CLOCK

A clock project using CMOS logic and seven-segment displays.



**T**here are digital clock projects and there are *really digital* clock projects. At one extreme are clocks made entirely with individual transistors, resistors, and other discrete components like the aggressively retro kits produced by KABtronics and described in these pages a few years ago [1]. At the other extreme are microprocessor based clocks with many thousands of circuit elements compressed into one or a few integrated circuits. A novel example of this kind of clock appeared in the March 2014 issue of *Nuts & Volts* [2].

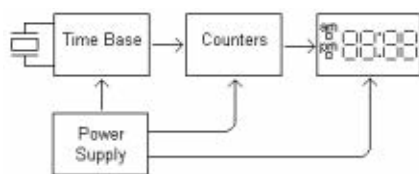
This project lies between these extremes, toward the early end chronologically. It is a straightforward 12 hour clock with CMOS integrated circuits and seven-segment LED displays. The most complicated ICs in it contain several flip-flops and some additional logic. The clock presents the time on a four-digit display — hours and minutes — with the colon between blinking once each second. Two more LEDs indicate AM and PM alternately. Two pushbutton switches allow the minutes and hours to be set, and the whole thing is powered by a small wall supply.



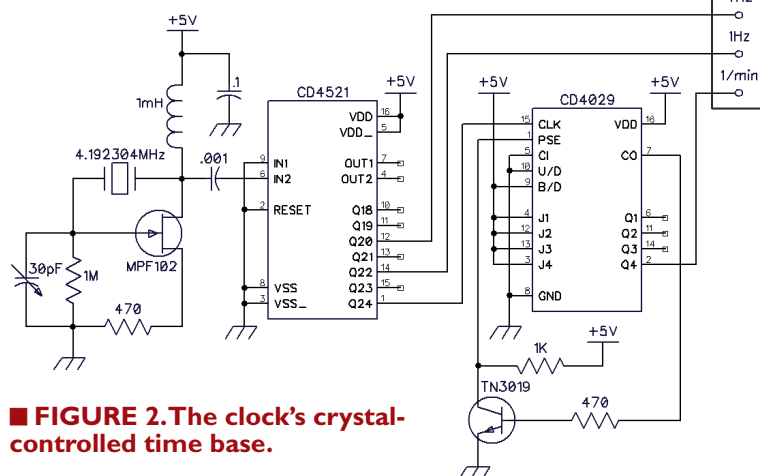
## Structure

A digital clock consists of four parts as **Figure 1** illustrates. A **time base** provides a signal of fixed frequency whose cycles are counted to mark the passage of time. That signal is derived either from the 60 Hz line frequency or — as here — from a crystal oscillator.

**Counters** count the cycles of the time base's signal and generate outputs that represent digits. Driver ICs decode these outputs to drive the **displays**, which show the time. Lastly, a **power supply** provides the power that the other parts require. In a microprocessor based clock, these functions are almost all carried out by the microprocessor; here, they are distributed among nine integrated circuits and a few transistors.



■ **FIGURE 1.** The overall design of the digital clock.



■ **FIGURE 2.** The clock's crystal-controlled time base.

its preset inputs, and it counts down so that it divides its input frequency by 15;  $4 \times 15 = 60$ , so the '4029's highest-order output completes one cycle every minute. This signal drives the clock's counters. **Figure 2** shows the circuit of the time base.

The carry-out output of the CD4029 is used to reload the counter when its count reaches zero. This signal goes high at that time, but the 4029's preset input is active low. A transistor inverter flips the carry-out signal so that it resets the count appropriately.

I could have used another IC, but that seemed inelegant for just one inverter. The transistor — like others in this project — is a TN3019 because I had a bunch of them. Any

general-purpose NPN will do.

## Time Base

The time base consists of one transistor and two integrated circuits. The transistor — an MPF-102 or similar FET — is used in a crystal-controlled oscillator. Many oscillator frequencies can be divided down to control a clock; here, the crystal's frequency is  $4.194304 \text{ MHz} = 2^{22} \text{ Hz}$ . The variable capacitor in the oscillator is a trimmer; it allows slight adjustment of the oscillator's frequency so that the clock will keep accurate time.

The first integrated circuit is a CD4521 (or an MC14521) which contains an inverter and a chain of 24 flip-flops. The output of each flip-flop is connected to the input of the next. The outputs of the last seven flip-flops are available. The first flip-flop receives the oscillator's signal. Each flip-flop divides the frequency of the signal it receives by two, so the output of the last flip-flop is the input frequency divided by 224.

With an input frequency of  $2^{22} \text{ Hz}$ , the output of the 20th flip-flop is 4 Hz; this signal is used to advance the hours and minutes quickly for setting. The output of the 20 second flip-flop is 1 Hz; this signal blinks the colon on the display.

The output of the last flip-flop is  $(1/4) \text{ Hz}$ ; that is, one cycle every four seconds. This signal goes to a CD4029 presettable four-bit binary counter. This counter is preset to 15 by imposing high signals representing binary 1s on

## Counting

Two of the time signals — 4 Hz and 1/minute — go to

### BCD and Counting

In digital logic, signals are binary: high and low voltages represent 1s and 0s, respectively. One such unit of information is a binary digit or bit. The 10 decimal digits can be represented by sets of four bits that indicate each digit's value in binary notation, thus binary-coded decimal (BCD). 0000 represents the digit '0'; 0001 represents '1'; and so on up to 1001, which represents '9'. The four bits are represented by the signals on four lines.

A flip-flop is a digital circuit whose output changes state — from low to high or from high to low — on each complete cycle of its input. Thus, a flip-flop produces an output signal whose frequency is half the frequency of its input signal.

Integrated circuit counters are chains of four flip-flops. The output of each is both an output of the counter and the input to the next flip-flop in the chain. If a signal with frequency  $f$  is presented to the first flip-flop, the four outputs have frequencies of  $f/2$ ,  $f/4$ ,  $f/8$ , and  $f/16$ . In the absence of any additional logic — a binary counter — the four outputs together repeatedly count in binary from 0 through 15.

In decade counters like the CD4518 and CD4510 used here, additional logic resets the flip-flops to 0 when the second and fourth outputs — representing 2 and 8 in the count for a total of 10 — are both high. The four outputs cycle from 0000 through 1001; that is, through the BCD representations of the digits 0 through 9.

the counting circuit. **Figure 3** shows this circuit and the display circuit that receives its outputs. Counting uses four ICs: a CD4518 dual decade counter; a CD4510 presetable decade counter; a CD4027 dual J-K flip-flop; and a CD4081 quad AND gate. The counters and one of the flip-flops generate signals that indicate the minutes and hours digits. Let's follow the 1/minute signal from the time base through these ICs.

The 1/minute signal enters one decade counter of the 4518 which counts the minutes and generates a binary-coded decimal (BCD) representation of the minute's 1s digit at its four outputs. The 8 output of this counter is connected to the input of the second counter in the 4518; when the minute's 1s count rolls over from 9 to 0, this output goes low and the second counter increments the minute's 10s count.

In the minute's 10s count, 0 should follow 5 as 59 minutes rolls over to 00. To accomplish this, the 2 and 4 outputs of the minute's 10s counter connect to an AND gate — one of four in the CD4081. When the minute's 10s count reaches 6 at the end of an hour and the beginning of the next, its 2 and 4 outputs are high so the AND's output goes high. This resets the minute's 10s count to zero and sends a pulse to the next counter — the CD4510 — which counts an hour.

As mentioned, this is a 12 hour clock, so the hour's 10s digit is always either blank or 1, and we don't need a full counter to keep track of it; one of the flip-flops in the CD4027 will do the job. When the hour's 1s count in the

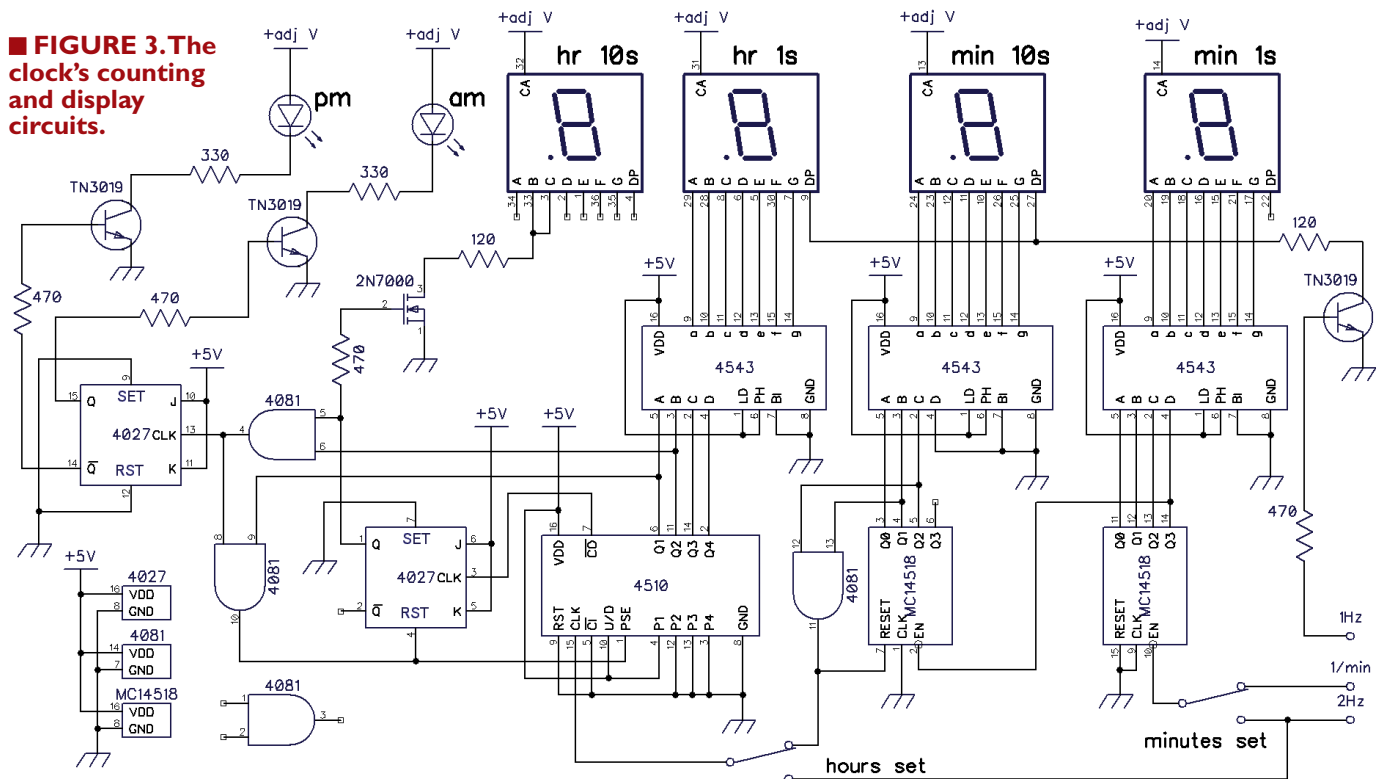
CD4510 rolls over from 9 to 0, the 4510 sends a signal to that flip-flop whose Q output goes high to indicate a 1 in the hour's 10s position.

At this point, things get a little tricky for two reasons. First, the hours count must roll over after 12; that is, the hours count must be reset when it reaches (very briefly) 13. Second, that count must roll over to 1 — not 0 — since 1:00 follows 12:59. Two more AND gates in the 4081 handle the first issue. Together, they notice when the hour's 10s digit is 1 (the Q output of that flip-flop is high) and the 1 and 2 outputs of the 4510 are high; that is, the hours count reaches 13. The output of the second AND gate goes high which resets the flip-flop — its Q output goes low — and raises the LOAD input of the 4510, whose preset inputs specify a value of 1. (It is possible to accomplish this transition with a non-presetable counter, but it requires more logic. That implementation is left as an exercise.)

The three decade counters and one flip-flop are now correctly counting the minutes and hours indicated by the 1/minute signal from the time base. All that remains is to light up the AM and PM indicators correctly.

It makes sense to assign this task to the second flip-flop in the 4027. It has complementary Q and  $\sim Q$  outputs, and exactly one of the indicators will be lit at any time. However, morning becomes afternoon and evening becomes morning again at 12:00, not 1:00, so we cannot use the hours-reset signal to toggle this flip-flop. Fortunately, we already have (as **Figure 3** shows) an

**FIGURE 3.** The clock's counting and display circuits.





appropriate signal.

One of the AND gates goes high when the hour's 10s signal and the hour's 1s 2 signal are both high; that is, at 12:00. This signal goes to the second flip-flop's input, so that each time the hours count reaches 12 it flips (or flops), turning one indicator off and the other on.

Two pushbutton switches apply the 4 Hz signal to the minutes and hours counting inputs to quickly advance those counts and set the minutes and hours.

## Displays

The outputs of the counter stages go to ICs and transistors that drive seven-segment common-anode displays: 3-1/2 digits for the minutes and hours counts, and the colon between the hours and minutes. The minute's 10s display is mounted upside-down so that its decimal point and that of the hour's 1s display form the colon. The 1 Hz signal from the time base controls the colon between the hours and minutes through a transistor switch, so the colon blinks once each second.

The details of the connections to the displays depend on the particular displays chosen. These can be four one-digit displays, two two-digit displays, or one display containing all four digits. The pin numbers in the figure conform to the last of these. In any case, consult the display's datasheet.

CD4543 decoders drive the three lower-order digits.

None of the parts are obscure or hard to find. All should be available through major distributors such as Digi-Key and Allied. All resistors are 1/4W.

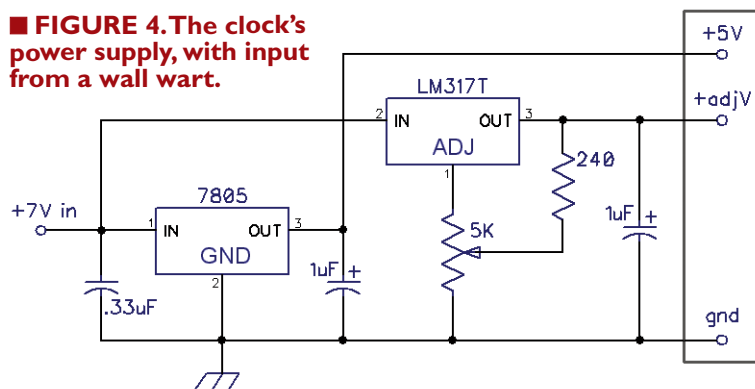
### Time Base:

CD4521 CMOS 24-stage frequency divider  
CD4029 CMOS pre-settable up-down counter  
MPF102 FET  
2N2222 or other general-purpose NPN transistor  
4.194304 MHz crystal  
1 mH choke  
30 pF trimmer capacitor  
.1  $\mu$ F, 16V capacitor  
.001  $\mu$ F, 16V capacitor  
1M resistor  
1K resistor  
2 - 470 $\Omega$  resistor

### Common Anode Counting and Display Circuits:

Seven-segment common-anode LED displays: 4 one-digit; 2 two-digit; or 1 four-digit, not multiplexed.  
2 - LEDs  
CD4518 CMOS dual up counters  
CD4510 CMOS presettable up/down counter  
CD4081 Quad two-input AND gate  
CD4027 Dual J-K flip-flop  
3 - CD4543 CMOS BCD-to-seven-segment latch/decoder/driver

■ **FIGURE 4. The clock's power supply, with input from a wall wart.**



They translate the BCD signals from the counters into signals that turn on the appropriate segments of the displays. The hour's 10s digit, the colon, and the AM and PM indicators are all controlled by transistor switches driven by the signals from the counters and the time base. A 2N7000 was used to drive the hour's 10s digit because a 2N3019 didn't switch in this situation; this is why we breadboard.

## Power Supply

Figure 4 shows the clock's power supply. It begins with at least 7 VDC from a wall supply. A 7805 three-terminal regulator provides regulated +5V for all the integrated circuits. An LM317 provides an adjustable positive voltage for the common-anode displays which

3 - 2N2222 or other general-purpose NPN transistor  
2N7000 Small-signal MOSFET  
2 - 330 $\Omega$  resistor  
4 - 470 $\Omega$  resistor  
2 - 120 $\Omega$  resistor  
2 - SPDT momentary pushbutton switches

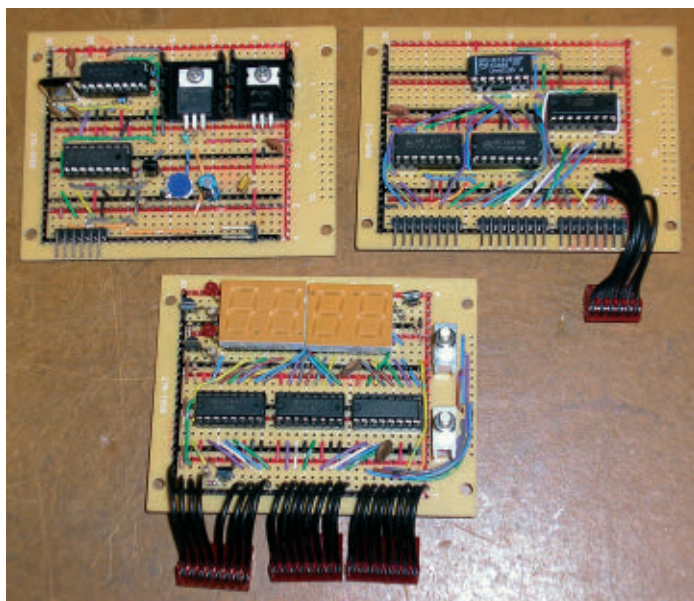
### Power Supply:

LM7805 Three-terminal 5V regulator in TO-220 package  
LM317T Three-terminal adjustable positive regulator in TO-220 package  
2 - 1  $\mu$ F, 16V electrolytic capacitor  
.33  $\mu$ F capacitor  
5K trimpot  
240 $\Omega$  resistor

### Miscellaneous:

Enclosure  
Tinted transparent acrylic to match the displays  
DC wall supply: 7V to 9V  
Headers and connectors  
Connector for the wall supply  
Standoffs  
Four rubber feet  
Hardware

**PARTS  
LIST**



■ **FIGURE 5.** The three circuit boards that make up the clock.

require less than 5V. The higher this voltage, the brighter the displays, so the pot that sets this voltage controls the display's brightness.

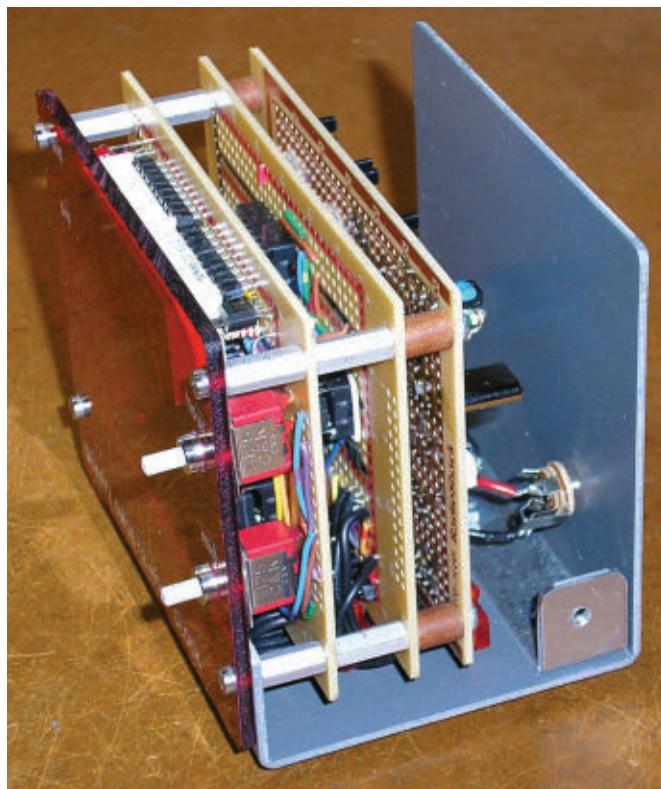
## Construction

I could have designed circuit boards for this project and had them fabricated, but prototyping boards (often called proto boards) are handy and convenient for one-off projects. For IC-based projects like this one, my favorite is the RadioShack 276-168 which has rows of three-hole pads separated by two buses, plus some extra two-hole pads on one end.

The recent downsizing of RadioShack means that these boards are not as easy to come by as they once were, but they remain available.

The clock occupies three of these boards. One holds the power supply and the time base, with small heatsinks on the two regulators. A trimpot sets the output voltage of the LM317. The second board holds the counters and their associated logic, and the third holds the drivers, displays, LEDs, and setting switches. All the ICs are mounted in sockets; the displays are mounted on SIP sockets; and all three boards contain several .1  $\mu$ F bypass capacitors (not shown in the schematics) from +5V to ground.

The board's traces are not indicated on their component sides, so I chose one bus on each for the ground connection and the other for the +5V supply, and traced them in black and red Sharpie™ on the board's component sides. The wiring was done with 24 ga solid insulated wire, with red used consistently for all +5V connections and black for all ground connections; do



■ **FIGURE 6.** The inside of the assembled clock.

these first. Other colors were used for visibility and clarity.

Right-angle headers and connectors attach the boards. With spacers and standoffs between them, the boards form a three-layer sandwich; the connectors allow the sandwich to be easily taken apart for debugging. **Figure 5** shows the three boards.

The clock's enclosure is bent up from two pieces of aluminum: one for the base and one for the top. The front panel is a piece of transparent red 1/4" acrylic (Plexiglas), drilled to accommodate four screws at its corners and the setting switches. (When drilling acrylic, enlarge small holes gradually and go slowly and carefully. Drilling too quickly can cause the bit to catch in the material. Clamp the material rather than holding it with your hand. Drill a few test holes first.)

The sandwich of the three boards and the acrylic panel is mounted to the base with two screws at its bottom, where the spacers are shorter than those at the top by the thickness of the acrylic. A connector for the wall supply is mounted on the back panel. **Figure 6** shows

## References

- [1] Keith Bayern: "Transistor Clock," *Nuts & Volts*, July 2009, pp. 42-46.
- [2] Craig A. Lindley: "A Unique LED Clock," *Nuts & Volts*, March 2014, pp. 33-39.





■ **FIGURE 7.**The completed clock.

the inside of the assembled unit.

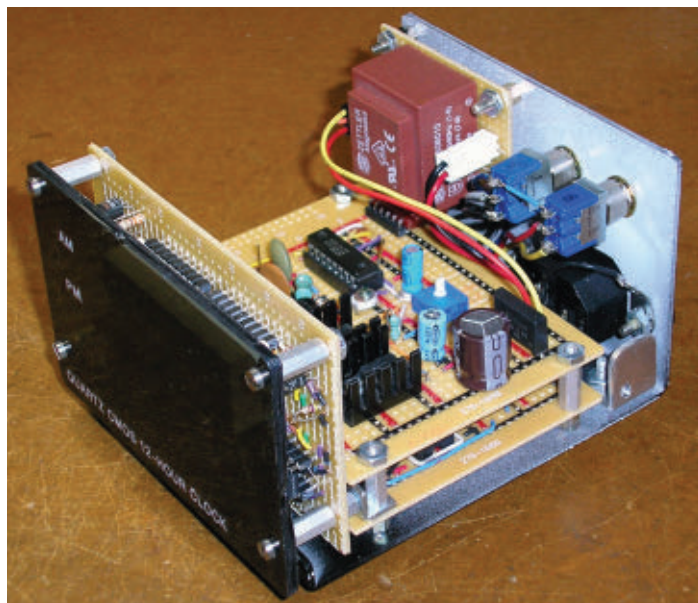
Self-tapping sheet metal screws hold the top to brackets at the rear of the base. The metal parts of the enclosure were cut from a salvaged panel painted with lightly textured gray paint. The paint cracked at the bends, but another light coat of gray fixed that and preserved the texture. The minimal labels were made with a Brother label maker. **Figure 7** shows the finished clock.

## Adjustment

The trimmer capacitor in the crystal oscillator allows slight adjustment of the oscillator's frequency. Set the clock, determine over a day or two if it is running fast or slow, and modify the capacitor's setting accordingly. It helps to sketch pictures of the capacitor's setting, and the process may take a few weeks. Be patient.

## A Second Clock

In designing and prototyping the clock I've just described, I built a second power supply and time base board. This one requires an input voltage of about 8 VAC, and therefore includes a full-wave bridge and an electrolytic filter capacitor. Also — in place of the CD4029 — it uses a 74C193, also a presettable binary counter which has the



■ **FIGURE 8.**The inside of the second clock.

advantage of not requiring an inverter between its carry-out output and its load input. Since this board was already assembled and tested, I decided to build another clock using it.

Again, there were three boards: the power supply and time base; a counter implementing the same circuit as before; and a driver and display board. The physical arrangement, however, was different. This time, the power supply/time base and counter boards formed a horizontal sandwich.

The display board — a section of a Datak 12-600B — was mounted to the counter board with long right-angle headers and two small Keystone right-angle brackets, which are threaded for 4-40 screws. (Jameco stocks these brackets as part number 1581530.)

The set switches and a small 8 VAC transformer were mounted on the unit's back panel, along with a three-wire computer-style line connector. As before, the sub-assemblies connect via headers and connectors, including (in this case) a five-wire connection from the counter



■ **FIGURE 9.**The two 12 hour clocks and a 24 hour clock.



board to the switches. The three boards are mounted to the bottom of their enclosure as **Figure 8** shows. **Figure 9** shows both clocks all put together with a third one of similar (but simpler) design and a 24 hour presentation.

## Cautions

If the setting switches bounce, setting the clock is an exercise in random numbers; quality switches are a good

investment. A project like this contains a very large number of solder joints often close together, so there are many opportunities for cold joints and solder bridges. Build stage by stage and test as you go.

An oscilloscope is very handy for testing and debugging, and it's interesting to see the waveforms and frequencies at various points in the circuit. You can test with signals of higher frequency than 1/minute. Modules are helpful; it's good to be able to take the clock apart for testing and debugging.

## Variations

Many variations of this general clock design are possible. Different oscillator frequencies can be divided down to the frequencies a clock requires, or a 60 Hz signal can be derived from the 120 VAC line.

Clocks of this overall structure can be built with a variety of integrated circuits. Many counter ICs are widely available and inexpensive. CMOS ICs (4000 series) are far more economical of power than are TTL (7400 series).

A 24 hour clock is simpler than one with a 12 hour display. It rolls over from 23:59 to 00:00, and there's no need for the AM/PM LEDs, though another decoder/driver IC is necessary. The minutes counting circuit can be preceded by an identical circuit that — beginning with a 1 Hz signal — counts and displays seconds.

The clocks described here draw no more than 150 mA at 5V, with almost all of that going to the displays and LEDs; TTL versions may draw as much as 500 mA, requiring a larger transformer or wall supply, and serious heatsinking for the regulators; mount the heatsinks to the metal back panel.

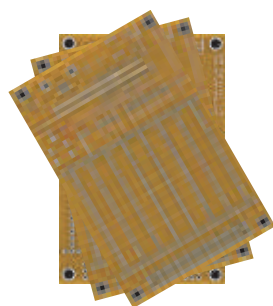
In any case, read the datasheets of the ICs you select, build and debug a breadboard first, and preserve the breadboard while you build the permanent version. **NV**

# FREE

PCB Layout Software &  
PCB Schematic Software



## expresspcb.com

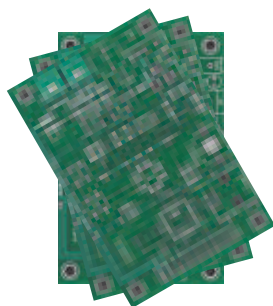


3 PCBs

**\$41** + shipping

2-layer  
**MiniBoard**

(Shipped in 1 business day)

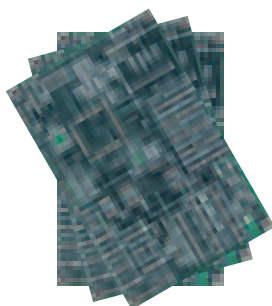


3 PCBs

**\$61** + shipping

2-layer  
**MiniBoardPro**

(Shipped in 2 business days)



3 PCBs

**\$81** + shipping

4-layer  
**MiniBoardPro**

(Shipped in 3 business days)

- 1** **DOWNLOAD** our free CAD software
- 2** **DESIGN** your 2 or 4-layer PCB
- 3** **CHECK** your design with xCHECK DRC
- 4** **SEND** us your design with just a click
- 5** **RECEIVE** top quality boards in just days

Now Offering  
**SMT  
STENCILS**

- Nano Coating available -



MADE IN THE USA | [www.expresspcb.com](http://www.expresspcb.com)

# COMPUTER CONTROL AND INTERFACING WITH THE NI MYRIO

By David Ward

*The four previous articles have already demonstrated most of the features that this final project will bring together in a complete system. Refer to **Photos 1 and 2** and **Figures 1 and 2**. There are a few new items in this VI (virtual instrument) that were not presented previously which will be explained in this final installment.*

**Part 5: A temperature control system and data logger.**

Planning a National Instrument (NI) LabVIEW VI is different than planning other types of computer programs that are sequential in nature, such as assembly language and C. Whereas a flow chart might work well when planning sequential programs, they probably won't be of much help when planning a LabVIEW VI. So, what particular planning approach works well for LabVIEW VIs?

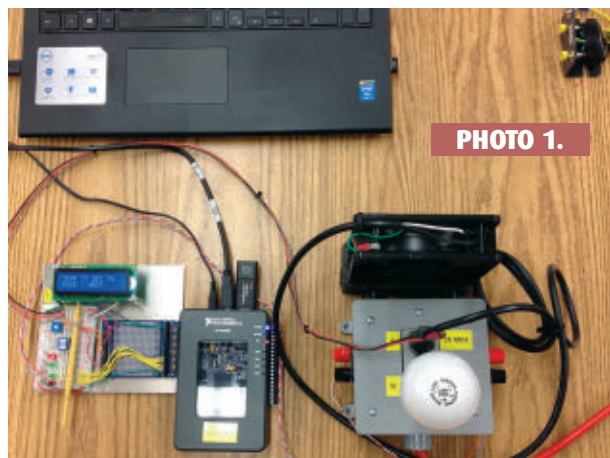
First, develop a written statement or description of what you want your program to accomplish, along with a title and a general program specification statement. Second, sketch out what you would like the front panel or user interface to look like. What basic controls, indicators, charts, etc., will be needed on the front panel to support the general program specifications? These controls and indicators can then be placed on the front panel at this point.

Each control or indicator that is placed on the front panel

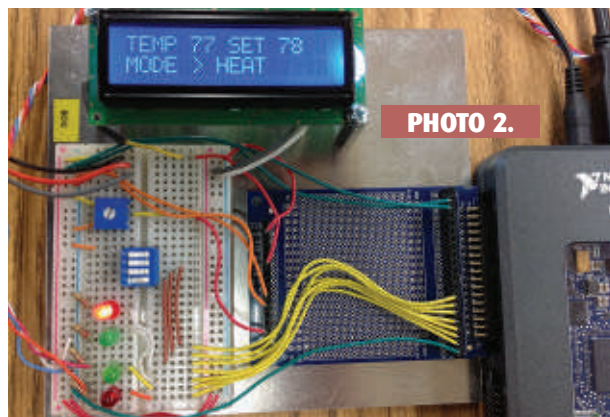
usually has an associated function icon that will appear on the block diagram as well. Some functions — like the MyRIO I/O functions on the block diagram — do not have icons that appear on the front panel. Following these suggestions will help you get your VI started and planned out.

As far as the block diagram is concerned, when the icons are placed down with the front panel controls, they will not be in any useful placement, order, or have any connections between them. The block diagram format that the original NI Main VI (that was first opened in Part 1) uses will work very well for most VIs that you will develop (see **Figure 3**). The front panel is not shown since it will only contain the stop control on it.

This format has a flat sequence structure with three frames placed down. The flat sequence will execute from left to right. The left or first sequence is commented as "Initialize." The center frame has a *while* loop placed in it



**PHOTO 1.**

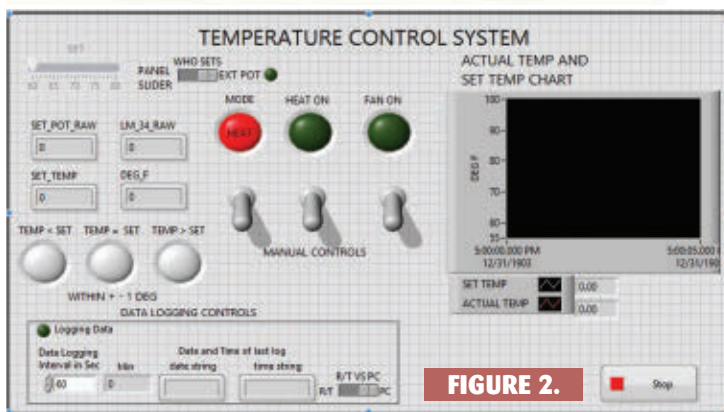


**PHOTO 2.**





**FIGURE 1.**

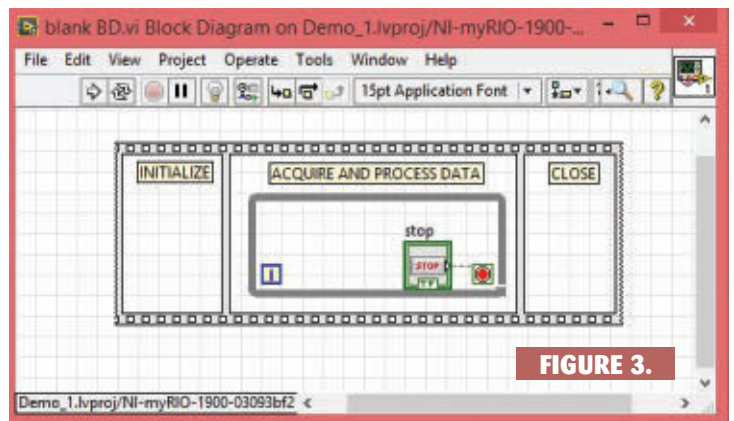


**FIGURE 2.**

and is commented as “Acquire and Process Data.” The right and final frame is commented as “Close.” The Initialize frame will execute first and can be used to make sure things like I/Os are off or on to begin with. It can also be used to initialize numerical values, clear charts, etc., and start your VI to the conditions you would like to see when your VI first opens.

Your VI will spend very little time in this frame. The Acquire and Process Data frame is where your VI will spend the majority of its time. It will complete items in the *while* loop over and over again until the termination “stop sign” icon changes to exit the *while* loop. How fast it can complete the *while* loop depends on many factors. However, if a delay is placed in this loop, it can slow it down as it will delay for its set time after each loop execution.

By the way, the little blue “I” in the lower left box in the *while* loop is the *while* loop iteration count. You can connect a numeric indicator to it and see how many loops the *while* loop has completed when the VI is run. This main *while* loop is where your VI will



**FIGURE 3.**

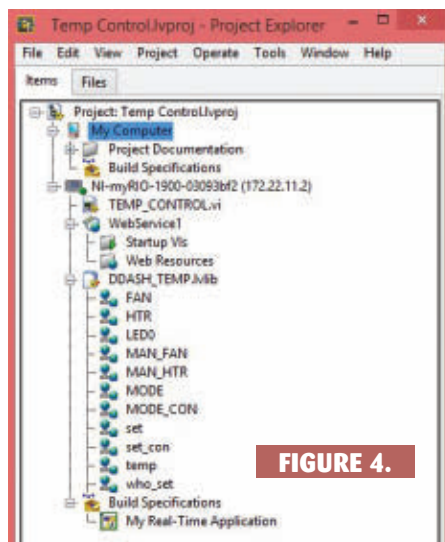
spend most of its time acquiring data from the MyRIO and the front panel controls, making decisions, performing calculations, solving logic conditions, and outputting and displaying the results.

When you decide to exit the VI by clicking on the stop button on the front panel, the *while* loop will terminate and the VI will execute anything in the final or right sequential Close frame. Here, you can again turn I/Os on or off, perhaps write any final data to a file, and perform any other tasks you may find necessary as you test the operation of your VI.

Let’s look at the final project – a temperature control system and data logger. The general program specification or description for this is: This VI will operate similar to a simple household thermostat used to control the temperature in the home’s living areas. It will have a control to set and display the desired temperature. It will display what the actual or current temperature is. It will have a control to select between either a heat mode or a cool mode. In the heat mode, a heater output will be on if the actual or current temperature is lower than the set temperature. In the cool mode, a cooler output will be on if the current temperature is higher than the set temperature. It will also

log essential data to a file at intervals chosen by the user. It will be capable of being controlled from three different sources: the VI’s front panel when connected to a PC; wirelessly from an iPad; or from actual “physical” controls wired to the MyRIO unit.

A word of advice as you begin building this VI. It was developed over a period of time piece by piece, adding new functions as older functions were tested, debugged, improved, and refined. Start simple and don’t try to do everything at one time. If you do not need particular features – the external LCD display or the wireless publishing to an iPad, for example – ignore them and work on



**FIGURE 4.**



the functions you want to incorporate.

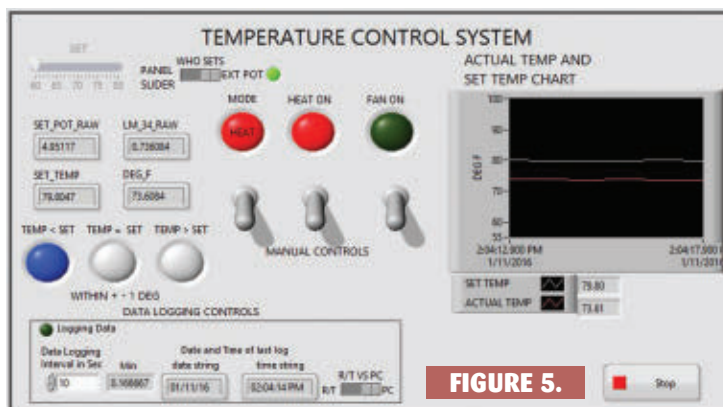
There will be problems when you try to duplicate anyone's complicated LabVIEW VI. Even having an electronic copy of a VI does not guarantee that it will operate on another MyRIO or PC without some troubleshooting taking place.

There are many files associated with any VI, and they can be located in different folders, directories, etc., on different computers. I realize that this system is not a cost-effective replacement for a \$20 household thermostat. This is just a way of providing a universally understood control system, and using LabVIEW and the MyRIO to solve the task while learning how to use them.

The file view, front panel, and block diagram for this VI can be seen in **Figures 4, 5, and 6.** **Figure 6** is an overall view of the complete diagram which will be broken down into individual boxes in the sections that follow. When explaining how the VI operates, most of the time will be spent going over how the block diagram works. It should be fairly straightforward to recreate the front panel and place the same controls and indicators as well as the iPad data dashboard. To make the block diagram explanations easier to follow, decorative frames were placed around most of the areas. Numbers were placed in the upper left corner of each decorative frame. These frames are for documentation purposes only and do not affect how the VI operates.

The Initialize frame on the left simply makes sure that the heat and cool outputs to the MyRIO D I/O lines are off. It also initializes who sets control to the front panel; this might be necessary in the event someone using an iPad had set itself as the temperature controller and did not return this control back over to the front panel. The Close frame on the right makes sure that the heat and cool MyRIO outputs are off. The explanations will be listed in order frame by frame. However, we'll call it box by box here so as not to be confused with the sequential frames.

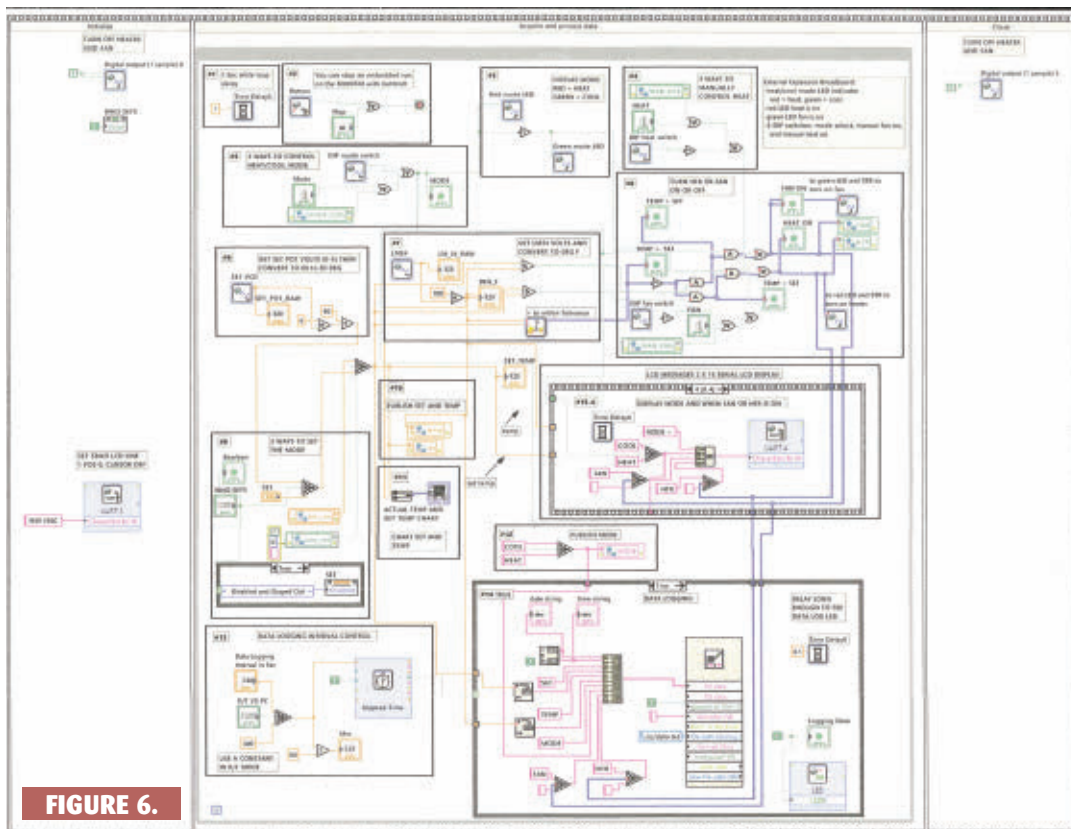
First, a word about the wireless "network published" variables you will find located throughout the block diagram. They are only necessary if you want to



**FIGURE 5.**

include the wireless control and display features through an iPad data dashboard. For example, in box #5 is the green **MODE\_CON** rectangle with an arrow pointing out of its right side. There is a corresponding **MODE\_CON** on the project view window under the **DDASH\_TEMP.lvlib** label. This icon was placed in box #5 by dragging it from the project view window down onto the block diagram.

The arrow on its right side is pointing out; this means that it is in the **READ** mode and will be wirelessly receiving data from the iPad data dashboard control linked to **MODE\_CON**. It will receive this data from the iPad control and pass that data into the VI. The green color signifies that it is a **BOOLEAN** type of data — either a **T** or **F**. Notice on the block diagram in box #10 there are two orange boxes with the words "temp" and "set" in them.

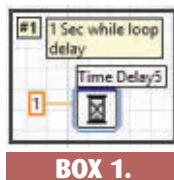


**FIGURE 6.**

Their arrows are on their left sides and are pointing in. This indicates that they are in the WRITE mode and are receiving data from the VI, wirelessly transmitting or “publishing” that data out of the MyRIO unit. The iPad data dashboard will have indicators linked to these and display their values. The orange color indicates that their data type is a DBL (double precision 64-bit number). You will also see pink rectangles for string data types; refer to box #12.

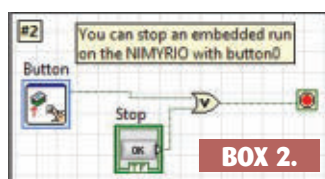
#### Box #1:

A one second time delay. This slows the *while* loop down so that the LCD is not constantly fluctuating. If you are not using an external LCD display, this could probably be eliminated.



#### Box #2:

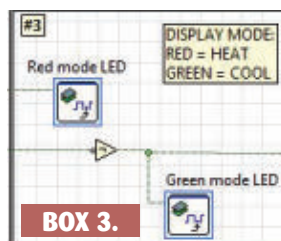
This is the main *while* loop terminator. By clicking on the front panel stop button or on the BUTTON0 on the MyRIO itself, the *while* loop will terminate.



The BUTTON0 feature allows the user to stop the VI when it is running in real time and a front panel is not present.

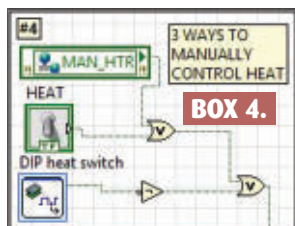
#### Box #3:

This turns on external physical LEDs located on a breadboard connected to the MyRIO unit. The red LED indicates that you are in the heat mode; the green LED indicates the cool mode. These are the two top LEDs in **Photo 2**.



#### Box #4:

This box allows three different ways to manually turn the heat output on, overriding the automatic temperature controlling function of the VI.

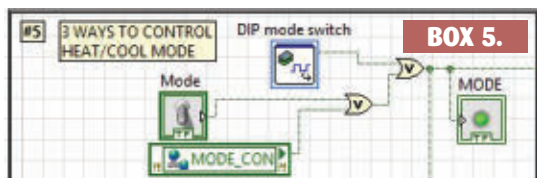


You can either turn the heater on from the iPad, click on the front panel HEAT ON manual control, or from the physical DIP switch #3 located on the breadboard.

You probably do not want to leave any of these controls on all of the time. They are really only there for troubleshooting purposes.

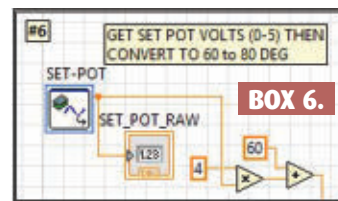
#### Box #5:

This box does the same thing as box #4 except it is for a manual override for the MyRIO cool output, which is DIP switch #2 on the breadboard.



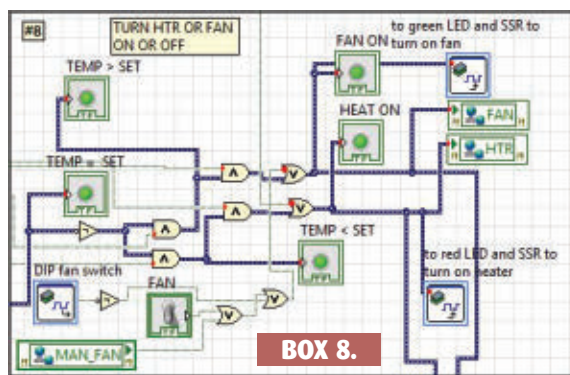
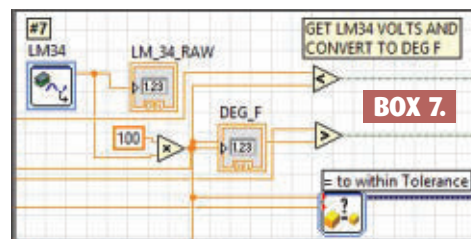
#### Box #6:

This box acquires the analog voltage from a physical 10K ohm potentiometer located on a breadboard. It converts its 0V to 5V range into a 60 to 80 degree temperature range;  $4 \times 5 = 20$ ,  $20 + 60 = 80$  for the maximum temperature that this pot can set.



#### Box #7:

This box acquires the analog voltage coming from the LM34 temperature sensor. It displays its raw or unaltered voltage on the front panel, then multiplies it by 100 to convert it to degrees F and displays the actual temperature. The “= to within tolerance” icon determines within + or - one degree when the heater or cooler should turn on or off. This makes it so that when the actual temperature equals the set temperature, the heat or cool output does not rapidly turn on and off attempting to maintain an exact temperature match but allows the system to be adjusted within a range.



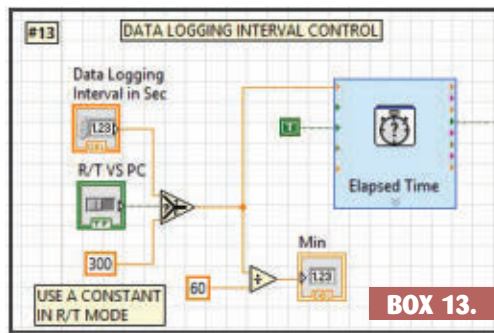
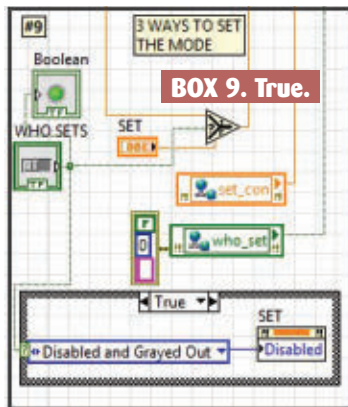
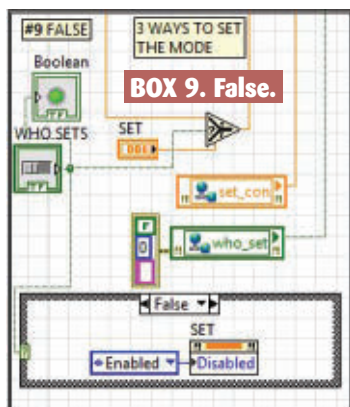
#### Box #8:

This box turns the heater on in the heat mode or the cooler output on in the cool mode. LED indicators were added to the front panel to show when the TEMP < SET, TEMP = SET, or TEMP > SET. There are two physical LEDs on the breadboard as well (see the lower two LEDs in **Photo 2**). The green LED comes on when the cooler output is on and the red LED when the heater output is on. SSR's positive control pins can be connected to the anodes of each of these LEDs to control actual higher voltage and current AC or DC loads.

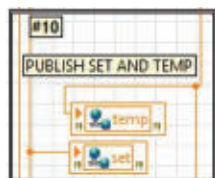
#### Box #9 True and Box #9 False:

This box determines who is actually controlling the set temperature: either the front panel control; the physical 10K pot on the breadboard; or the slider control on the iPad. At the bottom of this box is a true/false case





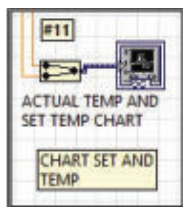
for the elapsed time function; however, they are converted and displayed in minutes too.



structure which will disable and gray-out the front panel slider when it is not selected as the control.

#### Box #10:

These two icons are where the set temperature and the current or actual temperature are published or transmitted wirelessly.

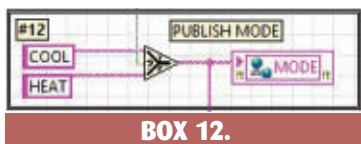


#### Box #11:

This box sends the set temperature and the current or actual temperature to the chart on the front panel.

#### Box #12:

This box is where the currently selected mode is published or transmitted wirelessly: when true, the string "COOL" is transmitted; when false, the string "HEAT" is transmitted. The triangle between them with the "?", "T," and "F" in it is a select: functions > programming > comparisons > select.



#### Box #13:

This box controls how often the data is logged to a file. The R/T VS PC control on the front panel should be set to R/T before building and deploying a real time version of this VI to the MyRIO. When it is run in real time, 300 seconds (five minutes) will be the interval.

When the front panel is in control of the VI, other values can be entered into the data logging interval control. These values are in seconds

#### Box #14 True and Box #14 False:

This is the data logging box and is placed in a true/false case structure. It will execute what is in the true case when the elapsed time has been completed. Several strings are concatenated or put together to be written to the data logging file.

The date, time, set temperature, the actual or current temperature, the mode (heat or cool coming from box #12), and finally if the cooling or heating outputs are on at the time of logging – "FAN" or "HTR."

Also, LED0 on the MyRIO unit and an LED on the



**MAKE UP TO \$100  
AN HOUR AND MORE  
AS AN FCC LICENSED  
TECHNICIAN!**

### Get your "FCC Commercial License" with our proven Home-Study Course!

- No costly school. No classes to attend.
- Learn at home. Low cost!
- No experience needed!
- **MONEY BACK GUARANTEE:** You get your FCC License or money refunded!



There are thousands of openings in Communications, Radio, TV, Satellite, Maritime, Radar, Avionics and other fields. With your FCC License it will be much easier to qualify for these exciting positions.

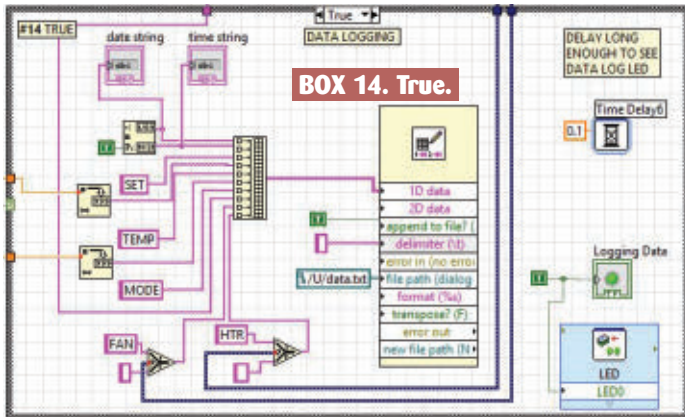
visit: **www.LicenseTraining.com**

or call for **FREE Information Kit: 800-932-4268**

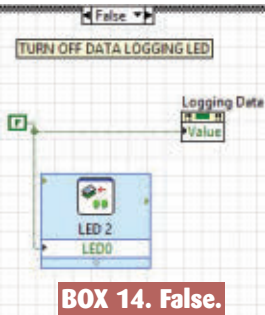
**COMMAND PRODUCTIONS • FCC License Training**  
Industrial Center, 480 Gate Five Rd., POB 3000, Sausalito, CA 94966-3000







**BOX 14. True.**

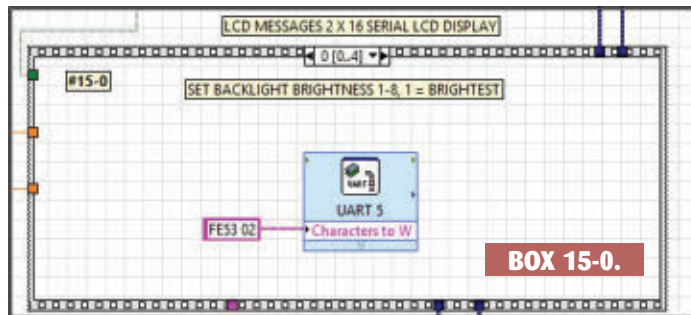


**BOX 14. False.**

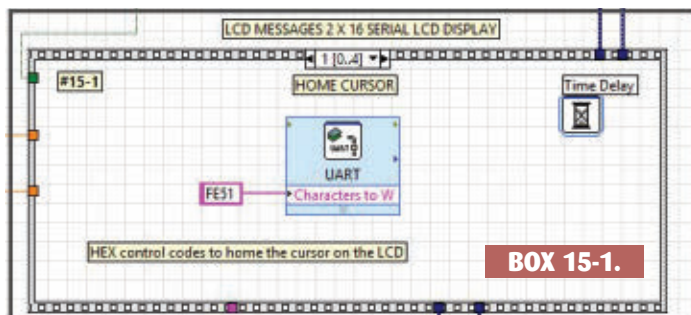
front panel will light up when the data is being logged. A 100 ms delay is placed here so that the user has enough time to observe the LED status. This lets you know that the VI data logging feature is working — this is especially useful when running in the real time mode. The false case simply turns these LEDs off.

#### Box #15-0 through -4:

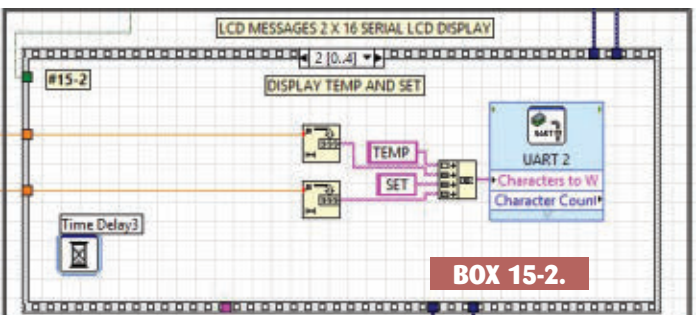
This is a stacked sequence: a flat sequence that is stacked one on top of the other mainly to conserve space. There are five frames in this sequence (0 through 4) which



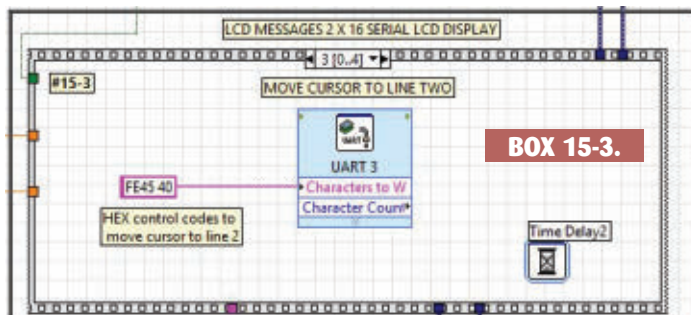
**BOX 15-0.**



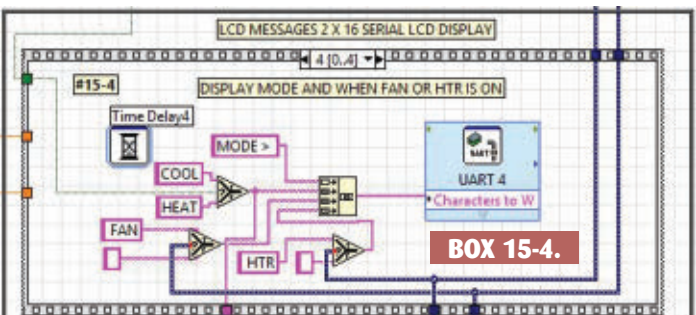
**BOX 15-1.**



**BOX 15-2.**



**BOX 15-3.**



**BOX 15-4.**

will execute in that order.

Most of the frames have a delay placed in them to accommodate the serial LCD that they are controlling, which is a NewHaven 2x16 display (NHD-0216K3Z-FL-GBW-V3). The UART control is transmitting or “writing” serial data at 9600 baud in hexadecimal format to the LCD.

To get hexadecimal control characters out to the LCD, select the string constant and from the pull-down menu change the display from normal to hexadecimal; see box #15-0, 15-1, and 15-3.

Most of the NewHaven control codes begin with FE (hex) and then certain other numbers to clear the LCD, change lines, etc. There is text in each frame briefly explaining what that frame is doing. Of course, these hex codes will probably change depending on the particular LCD manufacturer.

## Conclusion

Good luck and enjoy developing new projects for your MyRIO. Not all functions and capabilities of the MyRIO have been demonstrated in this series of articles, but hopefully enough has been presented to help you get started using the MyRIO and working with LabVIEW. It is a great system to develop with and has many other possible applications.

Again, a great MyRIO resource is at [www.ni.com](http://www.ni.com); search for MYRIO to see some of the things other people have been doing with their units and LabVIEW. **NV**

# Take a CAN Bus for a Spin

CAN controller  
interrupts and flags  
give us an edge.

By Jon Titus KZ1G

Post comments on this article and find any associated files and/or downloads at [www.nutsvolts.com/magazine/article/April2017\\_CAN-Bus\\_Controller-Interrupts-Flags](http://www.nutsvolts.com/magazine/article/April2017_CAN-Bus_Controller-Interrupts-Flags).

## Part 3

**This final article explains how to use the interrupts and flags available in an MCP2515 CAN controller IC, and describes how device-to-device acknowledgments work on the bus. The acknowledgment bit in a CAN frame plays a key part in successful communications and identification of errors.**

**A**lthough you can understand CAN controller interrupts and flags without a second CAN device, you will need one to experiment with interrupt-driven software. I use an inexpensive Microchip CAN BUS analyzer tool (**Figure 1**). This device accepts differential CAN bus or logic-level signals, and its PC software transmits and receives data via a USB connection.

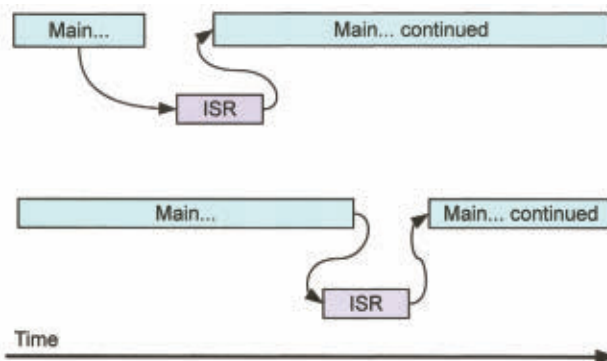
My investigations use a true differential bus such as the one shown in Figure 4 in the first *Nuts & Volts* article in this series (December 2016). For commercial work, better — and expensive — analyzers can purposely create bus errors, record and play back bus frames, and so on.

## Interrupts

Programmers often write code that runs in a loop and regularly tests a flag bit or I/O pin to determine whether a device needs attention and to take action if it does. The period between these software tests might range from a few milliseconds to hundreds of seconds depending on other tasks an MCU must perform. People and machines often require an almost-instant response, so most MCUs incorporate one or more interrupts. When an MCU receives an interrupt signal, it immediately stops any code currently running and transfers control to a separate program called an interrupt service routine (ISR). The ISR code disables the interrupt request input, runs code associated with the interrupting device, re-enables the interrupt input, and returns program control to the main program. **Figure 2** illustrates the program flow along a time axis. Because interrupts may occur at any time, ISRs



**FIGURE 1.** A Microchip CAN bus analyzer comes with PC software that displays bus communications and sends CAN frames. Courtesy of Microchip Technology.



**FIGURE 2.** Timing diagram for an interrupt service routine (ISR). An interrupt can occur at any time, which can lead to debugging problems. Aim to keep ISRs short, and eliminate blocking code and calls to non-reentrant code.

exist outside the main portion of a program. The asynchronous nature of interrupts can complicate testing because programmers cannot predetermine when an interrupt will occur.

The following program illustrates a simple ISR that turns on an LED when the input signal at mbed pin p12 changes from a logic 1 to a logic 0; a falling edge. (You can download all the software for this article and an extra program at the article link.) In the example below, the “infinite” while-loop statements cannot affect the LED called LED\_irq. Only the ISR can do that, and it exists as a section of code separate from the main program:

```
//Interrupt Test INTJT.cpp
#include "mbed.h"
//header file for mbed MCUs

DigitalOut myled(LED1);
//define output for flashing LED
DigitalOut IRQ_led(LED4);
//LED for interrupt indication
InterruptIn IRQ_pin(p12);
//Use mbed pin 12 as interrupt input

void LED_ISR()           //Interrupt service routine
{
    IRQ_led = 1;         //Turn on LED4 on mbed
}
//End of ISR code

main()
{
    IRQ_pin.mode(PullUp);
    //Use internal pull-up resistor
    IRQ_pin.fall(&LED_ISR);
    //IRQ_pin will detect a falling edge of
    //pulse. Specify the location of the ISR code
    IRQ_led = 0;         //Turn off LED4

    while(1)             //Run this loop forever
    {
        myled = 1;
        //Blink LED1 to show this code runs
        wait(0.1);
        myled = 0;
        wait(0.1);
    }
    //End of while loop
}
//End of main
```

In this program, I identify two onboard LEDs and define mbed pin 12 as an interrupt input I call *IRQ\_pin*. You may choose another name if you wish. The ISR – named *LED\_ISR* – contains one instruction that will turn on LED4. In the main portion of the code, the *IRQ\_pin.mode(PullUp);* statement turns on an internal pull-up resistor at the *IRQ\_pin* so the input becomes a logic 1 if left unconnected. (You could use an external pullup resistor instead.) The statement *IRQ\_pin.fall(&LED\_ISR);* sets the mbed pin so it will cause an interrupt only when it detects a logic 1 to logic 0 edge. It includes the address of the *LED\_ISR* code.

If you run this program and then ground mbed pin 12, LED4 will turn on. Nowhere in the main program does the MCU test the state of pin 12. When you use the online mbed tools and libraries, the mbed Cortex-M3 processor automatically disables interrupts when it

starts an ISR and re-enables them when it leaves an ISR. Other MCUs such as those in the PIC families require ISR software to handle these tasks.

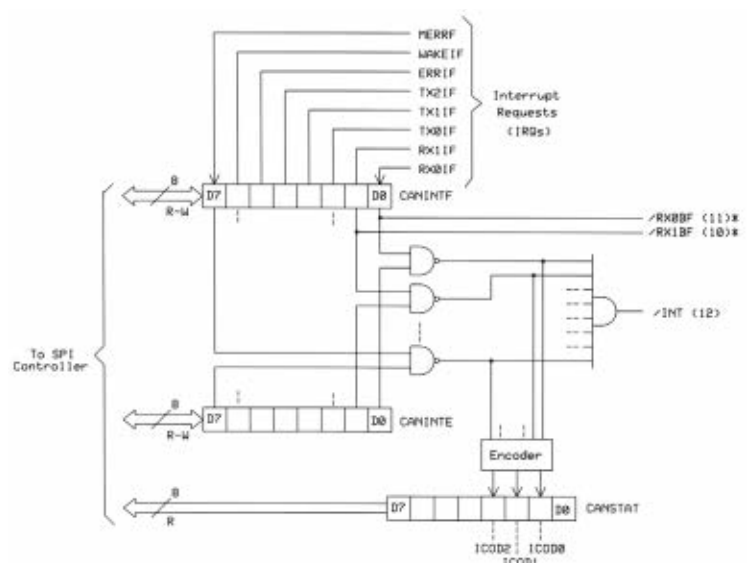
## Interrupts for an MCP2515

Because we don’t know when a CAN controller will receive data, an interrupt can let an MCU immediately attend to the receiver buffers. The MCP2515 CAN controller provides one interrupt-output signal (/INT) at pin 12. The MCP2515 *CANINTF* (CAN Interrupt Flag) register holds flag bits for eight devices or events. A logic 1 bit indicates a request for service:

1. *RX0IF* and *RX1IF* show when the corresponding receiver buffer has new data.
2. *TX0IF*, *TX1IF*, and *TX2IF* mark the corresponding transmitter as having completed a transmission.
3. *ERRIF* lets software know an error of some type occurred.
4. *WAKEIF* lets an MCU know the MCP2515 has detected bus activity while in the low power “sleep” state.
5. *MERRF* indicates an error occurred during transmission or reception of a message.

As explained earlier, software can read the *CANINTF* register value and test flag bits at any time to determine whether a device or event needs attention. (After the software takes action, it must reset the corresponding *INTF*-register flag bit to a logic 0.) The *CANINTF* bits alone will not cause an interrupt.

A program also must set the corresponding bit or bits in the CAN Interrupt Enable (*CANINTE*) register. The contents of this eight-bit register let us enable (logic 1) or disable (logic 0) interrupts as needed. **Figure 3** shows the



**FIGURE 3.** A simplified diagram of the registers, gates, and signals involved in MCP2515 interrupts. Pin numbers refer to the DIP version of an MCP2515 IC. Register read-write operations occur through an SPI connection with an MCU.



registers and pins the MCP2515 CAN controller uses to manage interrupts. (For clarity, I omitted details not pertinent to this article.)

When a program sets the *CANINTE* register to 0x1F (00011111<sub>2</sub>), only the three transmitter (*TXnIF*) and two receiver (*RXnIF*) flags may cause interrupts. For many applications, they suffice. Regardless of bits set or cleared in the *CANINTE* register, a program may always read *all bits* from the *CANINTF* register. (Refer to sections 6 and 7 in the MCP2515 datasheet for more information about the error and wake interrupts.)

When a program depends on the state of a *CANINTF* flag bit to, say, read a receiver buffer, ensure you clear that flag bit with an MCP2515 bit modify command after the program completes the read operation. This rule applies to all eight flag bits. A program cannot force a flag bit to clear if the condition that “raised” the flag still exists. Thus, if *RXB0* receives data and you do nothing with it, the logic 1 remains set in the *INTF* register’s *RX0IF* bit.

The following program demonstrates how to set up code for an MCP2515 interrupt and how the code responds. In this case, the MCP2515 causes an interrupt *only* when it receives a message with ID 0x105. Each interrupt causes an LED to change its state. To test this software, you need a second CAN device to send a message with the ID 0x105. Ensure the MCP2515 */INT* signal at pin 12 connects to the MCU’s interrupt pin (mbed P12 in this example).

It’s important to note the ISR only changes the LED’s state, sets a software flag (*RXB0\_Flag*) to “true,” and clears the *RXB0IF* flag bit. So, the ISR takes only microseconds to service the interrupt. The main program periodically tests the software flag and then runs your application code to handle the *RXB0* data. Of course, you could do the same steps in other ways. I aim to keep ISRs short:

```
/* Program CANJT4.cpp */

#include "mbed.h"
//header file for mbed MCUs
#include "MCP2515JT.h"
//header file for MCP2515 CAN IC
DigitalOut myled(LED1);
//define output for flashing LED
DigitalOut IRQ_led(LED4);
//LED for interrupt indication
InterruptIn IRQ_pin(p12);
//Use p12 on mbed for IRQ
byte RXB0_Flag;
//Flag bit indicates RXB0 data

//Interrupt-service routine for RXB0
receipt of new data
void RXB0ISR()
{
    IRQ_led = !IRQ_led; //Toggle LED4
    bitModify(MCP2515INTF, 0x01, 0x00);
    //Clear RXB0 interrupt
    RXB0_Flag = true;
    //Set this flag to "true"
}
//End of RB0ISR

//Main program starts here
int main()
{
```

```
    reset(); //Reset MCP2515
    setMode(CONFIGURATION);
    //Put MCP2515 in CONFIG mode
    baudConfig(125); //Set CAN bit rate to 125kbps
    setMask_0(0x3FF); //Set RXM0 (Mask-0) to have
    // no effect
    setFilter_0(0x105); //Set RXF0 acceptance filter
    // F0 for ID = 0x105

    setMode(NORMAL); //Use normal MCP2515 mode
    bitModify(RXB0CTRL, 0x60, 0x20);
    //Use only RXF0 filter
    bitModify(MCP2515INTF, 0x03, 0x00);
    //Clear RX0 and RX1 interrupt flag bits
    IRQ_pin.mode(PullUp); //Use the internal pullup

    //resistor on mbed p12
    IRQ_pin.fall(&RXB0ISR);
    //Set the IRQ_pin to detect a falling edge 1->0
    IRQ_led = 0; //Turn off IRQ indicator LED
    RXB0_Flag = false; //Clear this flag when
    // program starts
    bitModify(MCP2515INTE, 0x01, 0x01);
    //Set RX0IE interrupt-enable bit to logic-1

    while(1) //Run this loop forever
    {
        myled = 1; //LED-blink code to show
        // program working
        wait(0.1);
        myled = 0;
        wait(0.1);

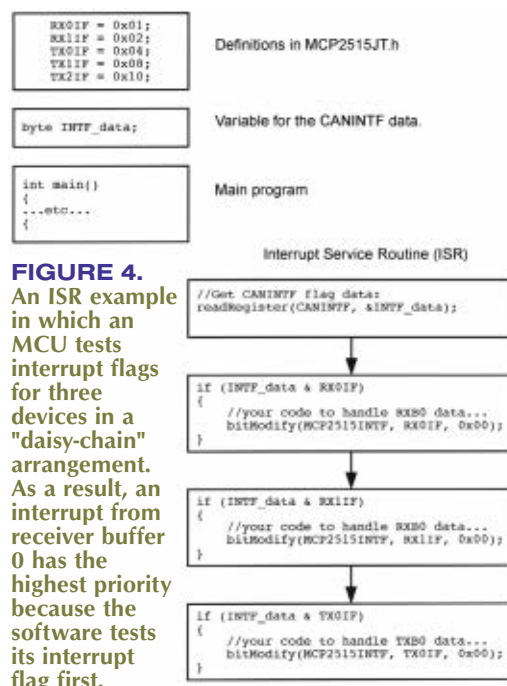
        if (RXB0_Flag == true)
            //Test to see if valid data has arrived
            {
                //Your data-handling code goes here...
            }
        //End of while loop
    }
    //End of main
```

## Test Several Interrupts

If you set the *CANINTE* register so more than one condition will cause an interrupt (refer again to **Figure 2**),

the ISR must test the *CANINTF* flag bits one by one to determine what event or events caused the interrupt. The ISR could use bitwise-AND operations to test the flag bits in any order. The flow chart in **Figure 4** offers an example of this type of ISR which tests for an interrupt from *RXB0*, *RXB1*, and *TXB0*. The MCP2515JT.h file defines a one-bit mask for each of the eight interrupt flags, *RX0IF*, *RX1IF*, and so on.

The mbed’s



Cortex-M3 processor includes a Nested Vectored Interrupt Controller (NVIC) that lets programmers control as many as 240 interrupts with a priority from 0 to 255 for each. It takes quite a bit of study to understand the NVIC, but it's available for complicated programs and real time operating systems (RTOSs). Find information about the Cortex-M3 processors on the ARM website at <http://infocenter.arm.com/help/index.jsp>.

## Interrupt Tips

Keep all ISRs as short as possible because they “steal” time from a main program. In poorly written software when an ISR from a low priority device runs, it might inadvertently prevent a device with a higher priority from executing its own ISR. We call this a *priority inversion*, which you want to avoid. For more information, visit [https://en.wikipedia.org/wiki/Priority\\_inversion](https://en.wikipedia.org/wiki/Priority_inversion).

Also, do not use *non-reentrant functions* such as *printf* or *malloc* in an ISR. Suppose a program starts to run a *printf* operation and gets part way through when the MCU receives an interrupt from device X. The device-X ISR also includes a *printf* statement. After the ISR completes its task, control goes back to the main program. Because your code can not re-enter the interrupted *printf* statement, you have a problem. The partially complete *printf* function in the main program fails. Check your compiler specifications for reentrancy information, and don't write non-reentrant code! For more information, visit [www.embedded.com/electronics-blogs/beginner-s-corner/4023308/Introduction-to-Reentrancy](http://www.embedded.com/electronics-blogs/beginner-s-corner/4023308/Introduction-to-Reentrancy).

In addition, an ISR — really any program — must not include blocking code or code that could block following

operations until another operation completes a task. Suppose you always expect two CAN messages, one right after another. The first message arrives and triggers an interrupt. The ISR gets the first message and then runs a loop to wait for the second message. If the second message never arrives, the program waits and waits, and waits. It never gets out of the ISR (Ref. 1).

Instead, programmers might write an ISR that saves the first message, increments a counter, and immediately returns to the main program. A second message would go through the same ISR actions. The main program could periodically test the counter for the value 2 and then process the two messages. The main program could include, say, a 100 millisecond clock that could cause an error interrupt if a second message hasn't arrived during that period (Ref. 2).

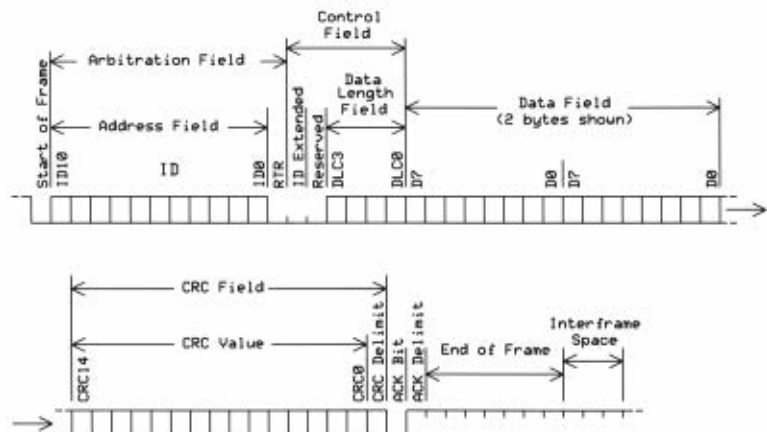
If you don't plan to use interrupts, code can run through a loop, monitor the *CANINTF* bits, take action based on the flag states, and then reset the flags. Just ensure you have set the *CANINTE* (CAN interrupt-enable) register to 0x00. This value prevents any flag bit from causing an interrupt at the */INT* output (pin 12). Of course, if you leave this pin unconnected, there's no concern about interrupts.

The MCP2515 provides two special outputs — */RXBOBF* and */RXB1BF* — you can use as individual interrupt or flag signals that indicate the respective receiver buffer has new information. These flag outputs operate independent of the *CANINTE* register bits. If you use these two flags, ensure your software clears them after you read the respective receiver buffer.

The MCP2515 also provides an interrupt code in the three read-only *CANSTAT.ICOD* bits, but I have never used them. The three-bit code presents interrupt information in priority order, so 001<sub>2</sub> (the highest priority) indicates an error. The code 111<sub>2</sub> identifies an interrupt from the *RXB0* buffer. See the MCP2515 datasheet for details.



**FIGURE 5.** The logic analyzer information for the transmission of data bytes 0x0F and 0xC4 with a standard identifier of 0x0AF. The green circle identifies the acknowledgment bit. The three red arrows point to “extra bits” placed in the data by the sending CAN controller.



**FIGURE 6.** The arrangements of bits in a normal CAN transmission with an 11-bit ID and two bytes of data. Tick marks are included to clarify bit timing.

## Acknowledge Errors

An important characteristic of CAN communications involves the acknowledgment of an error condition. To start this discussion, I'll first give you an overview of a CAN communication. For clarity, I'll discuss only communications that use a standard 11-bit identifier (ID). Let's assume we have four CAN devices that each have one receiver ID filter set for 0x0C1, 0x02B, 0x0AF, and 0x0A0, respectively. For now, we'll treat the values as “addresses.” Device C1 will send two data bytes — 0x0F and 0xC4 — to device AF. **Figure 5** shows the logic signals that appear on the bus.

More generally, all CAN transmission begins with a logic 0 start-of-frame (SOF) bit, followed by the 11-bit ID (**Figure 6**). The remote transmission request (RTR) bit that follows the ID indicates a

standard data message (logic 0) or an RTR (logic 1). Device 0x0C1 sends 0x0AF a standard message. A CAN controller will send an RTR message to request information from one or more devices. That transmission includes an address but no data bytes. You program a register bit to create an RTR message.

The CAN standard specifies a six-bit control field in which the four LSBs hold the data-length-code (DLC) that indicates the number of data bytes: 0 through 8 or 0000<sub>2</sub> to 1000<sub>2</sub>. The original CAN standard reserved the two MSBs in the control field for future use. In the latest standard, one bit — labeled ID Extended — indicates whether a frame uses a standard 11-bit or an extended 29-bit identifier. One bit still remains reserved for future use.

Next, the CAN frame includes a 15-bit cyclic redundancy code (CRC) value calculated by the sending controller, followed by a logic 1 “delimiter” bit that ends the CRC field. After the CRC-delimiter bit, the sender transmits a recessive (logic 1) output and its receiver monitors the bus. Whether or not specifically addressed, *all* devices on the bus receive the frame and *all* calculate a CRC based on the data they received. They all use the same CRC algorithm.

Why do the “unaddressed” devices also need to calculate the CRC? It helps them detect controller problems. After the CRC calculations, each device compares its result with the CRC in the received frame. When the two match, a CAN device transmits a single-bit dominant (logic 0) acknowledgment (ACK) signal on the bus. If the CRCs do not match, a device puts out a one-bit recessive (logic 1) state.

Assume all receiving devices work properly and all put

## Resources

ARM Cortex-M3 Nested Interrupt Vector Controller (NVIC):  
<http://infocenter.arm.com/help/index.jsp>

Reentrant Code Tutorial:  
[www.embedded.com/electronics-blogs/beginner-s-corner/4023308/Introduction-to-Reentrancy](http://www.embedded.com/electronics-blogs/beginner-s-corner/4023308/Introduction-to-Reentrancy)

Priority Inversion:  
[https://en.wikipedia.org/wiki/Priority\\_inversion](https://en.wikipedia.org/wiki/Priority_inversion)

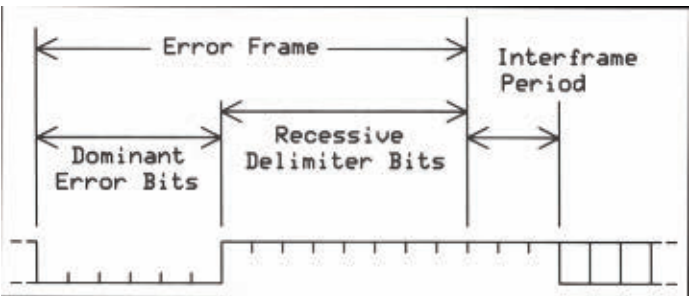
## References

1. “Non-Blocking Code,” Engscope;  
[www.engscope.com/pic-example-codes/non-blocking-code](http://www.engscope.com/pic-example-codes/non-blocking-code).

2. Titus, Jon, “The Hands-On XBee Lab Manual,” Newnes Press, Waltham, MA, USA. 2012, pp. 228-246. ISBN: 978-0-12-391404-0; [www.amazon.com/Hands--XBEE-Lab-Manual-Communications/dp/0123914043](http://www.amazon.com/Hands--XBEE-Lab-Manual-Communications/dp/0123914043).

3. Voss, Wilfried, “A Comprehensive Guide to Controller Area Network,” Copperhill Media, Greenfield, MA 2008. ISBN: 978-0976511601; <http://copperhilltech.com/technical-literature>.

4. Microchip MCP2515 datasheet, DS21801G.  
[ww1.microchip.com/downloads/en/DeviceDoc/21801e.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/21801e.pdf).



**FIGURE 7.** An error frame includes six dominant bits followed by eight recessive bits in an error delimiter. Only an error frame purposely puts this arrangements of bits on a CAN bus.

a logic 0 on the bus during the ACK period. The sender monitors the bus and expects this logic 0 signal. So, no CAN device reported an error and everything seems to look good. (Keep in mind the acknowledgment only indicates a CRC match and not that a message was received successfully.)

## Oops, a CRC Mismatch

What happens in a CAN controller when a received CRC and a calculated CRC don’t match? This situation provides a good example of the clever capabilities of CAN communications you cannot get with simple UART transfers:

1. Say one CAN device (XYZ) calculates a CRC not equal to the CRC in a message. That error will cause it to keep its transmitter output in the recessive state during the ACK period. At the start of the ACK period, device XYZ detects a dominant state on the bus caused by other receivers that have no CRC mismatch. The difference between the dominant “CRC-OK” signal on the bus and recessive “CRC-mismatch” signal from device XYZ causes it to send an error frame (Figure 7) right after the ACK delimiter period.

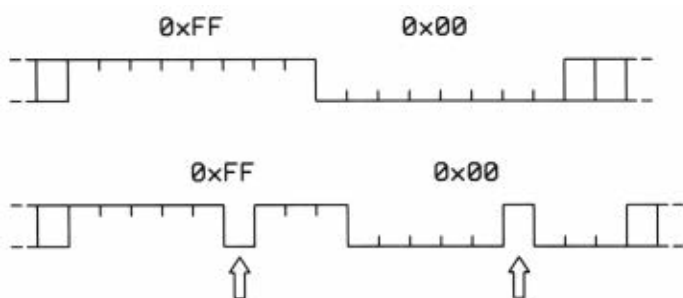
2. Now consider the case when *all* devices on a bus detect a CRC error. They all put a recessive signal on the bus. The sender expects a “CRC-OK” dominant signal instead. The recessive signal indicates the sender either calculated an incorrect CRC or transmitted an incorrect CRC. So, the sending device knows it has a problem and reports an error.

Other types of errors can occur at other times and also cause transmission of an error frame. The receipt of an error frame may trigger a resend of the original message. CAN controllers can count errors and cause an interrupt if they exceed preset limits. However, the CAN bus has proven very reliable and data errors rarely occur, although you could experience signaling problems in a poorly designed bus topology.

## What’s an Error Frame?

As shown in Figure 7, an error frame comprises a





**FIGURE 8.** To send a message with 0xFF and 0x00, a CAN controller inserts stuff bits (arrows) so a sequence of bits includes no more than five bits with the same state. This example does not take into account any bits prior to the first 0xFF byte. Can you describe an algorithm to remove stuff bits at a receiver?

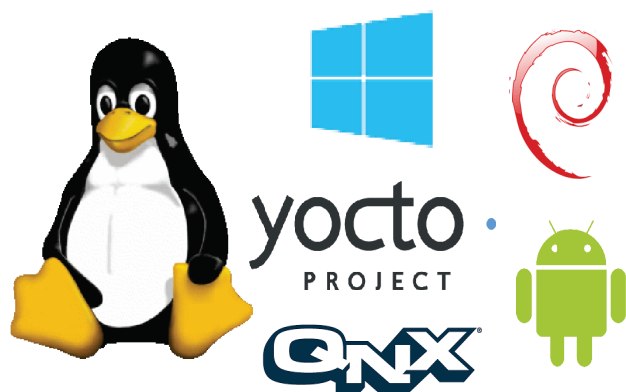
dominant state (logic 0) for six clock periods followed by a recessive state (logic 1) for eight clock periods. A three-bit recessive interframe space separates the error frame bits from any other information that might next appear on the bus. Some people say the six dominant states “violate” the CAN specification, but they’re simply a *different part* of the standard. The so-called violation occurs because CAN controllers allow only five consecutive bits at the same logic level – recessive or dominant. So, how will a CAN message handle a transmission of 0xFF followed by 0xFF, or 16 logic 1 bits in a frame?

A CAN bus requires a way to have devices stay “locked” on the clock frequency that governs the bit rate. So, a transmitted message provides synchronization information during logic-level transitions. To ensure often-enough transitions occur, a CAN controller automatically inserts or “stuffs” an extra bit in the communication as needed. Look at the transmission of 0xFF and 0x00 (**Figure 8**). The controller allows only five bits with the same logic state. If the sixth bit has the same logic level, the controller inserts a bit with the complementary logic level and then the sixth bit follows. Thus, you will see a maximum of five same-level bits in a CAN message (see **Figure 5** also). The six-bit recessive error frame includes no stuff bits and thus provides a way to signal an error condition to other bus devices.

You now have a good introduction to how the CAN bus and CAN controllers operate, and how communications occur. You might find a use for this bus when you need reliable high speed communications without a lot of complicated hardware or large protocol stacks.

I find the MCP2515 CAN controller easier to use than many CAN controllers built into MCUs and recommend it as a good IC to use as you investigate CAN capabilities. I welcome comments and questions at [tituskz1@gmail.com](mailto:tituskz1@gmail.com). **NV**

## Choose your own adventure.



### TS-4900 Computer on Module

WiFi and Bluetooth Enabled  
1 GHz i.MX6 Computer Module  
Now Supporting Ubuntu Core

- 1GHz Single or Quad Core Cortex A9 ARM CPU
- Up to 2GB DDR2 RAM
- 4GB MLC eMMC flash storage
- WiFi and Bluetooth radios on-board
- High speed industry standard connections
- Rugged, Industrial with Fanless -40°C to +85°C Range

**Starting at**  
**\$109**  
Qty 100  
**\$144**  
Qty 1

Now Supporting:



ubuntu core

For more details go to: **TS.TO/UC**

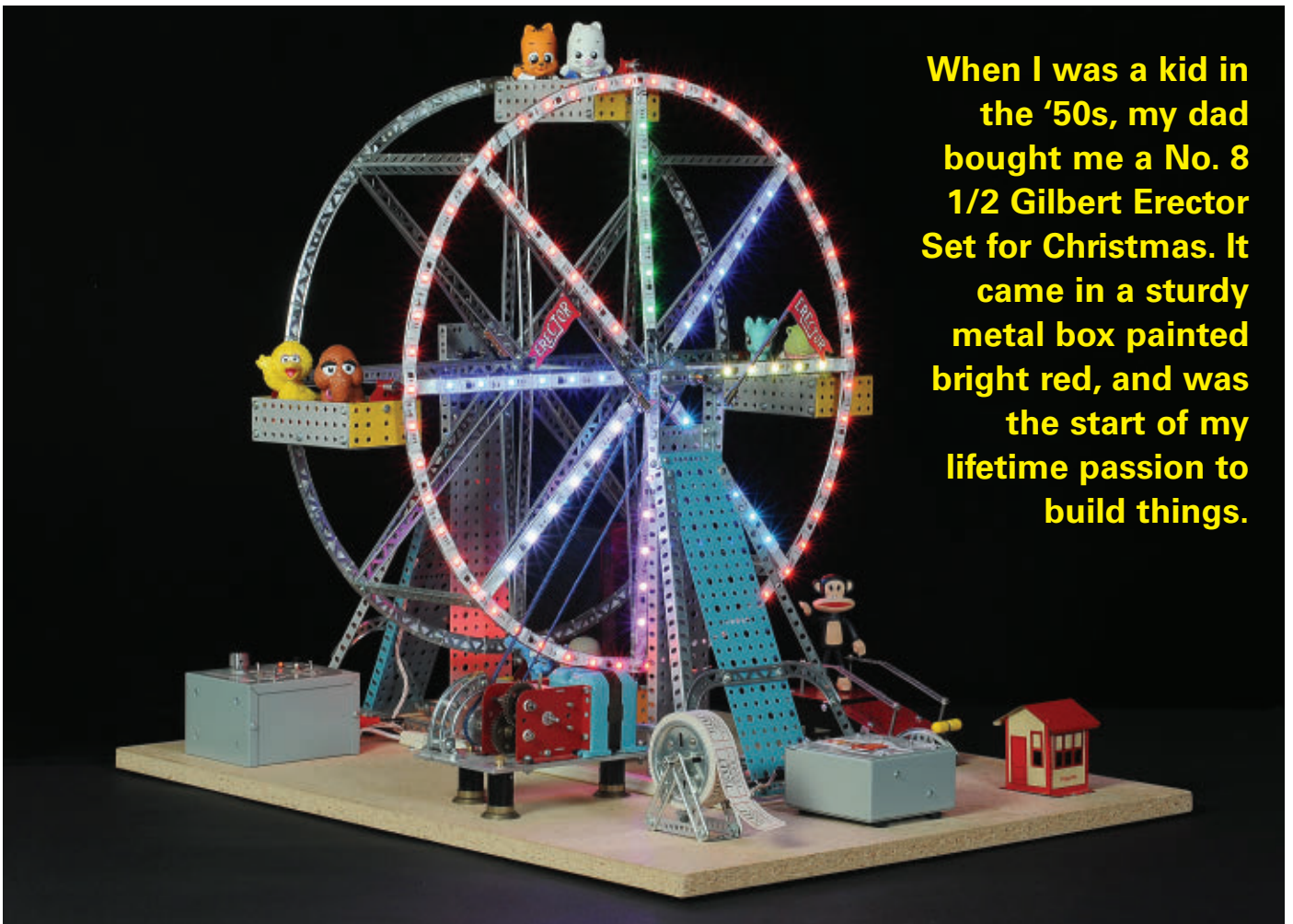
[www.embeddedARM.com](http://www.embeddedARM.com)

**Technologic**  
Systems

By David Goodsell

# The Magic of the Gilbert Erector Set ...

*or How I Built a Ferris Wheel for Big Bird*



When I was a kid in the '50s, my dad bought me a No. 8 1/2 Gilbert Erector Set for Christmas. It came in a sturdy metal box painted bright red, and was the start of my lifetime passion to build things.

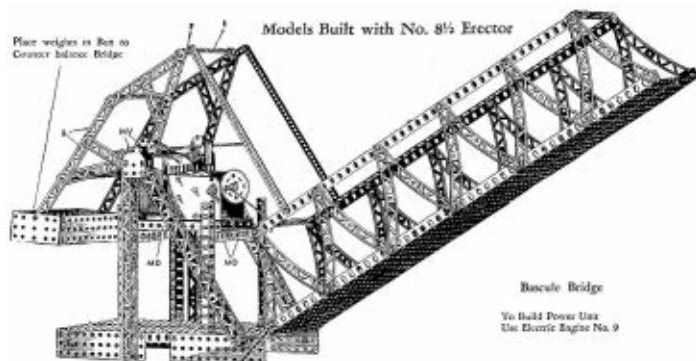
## The Big Picture

Back then, there were many different sizes of Erector sets, from the simple No. 1 1/2 to the complex size of 12 1/2 and beyond. Each version had more and different parts, and included the most fantastic manuals with dozens of beautiful illustrations of the exciting things to build like lift bridges, beam engines, drilling rigs, and the ultimate Giant Ferris Wheel. **Figure 1** shows a typical illustration.

**Figure 2** is just like the set I had as a kid. It contained no less than 253 parts plus 357 nuts and bolts to hook everything together. However, the real magic of the set was the lack of step-by-step assembly instructions. There were none. The manual had intricate drawings of the things to build, but I had to figure out the exact assembly sequence using my imagination. It was a great learning experience.

Unfortunately, Gilbert Erector Sets are long gone from the marketplace due to many factors. Luckily, many

Post comments on this article and find any associated files and/or downloads at [www.nutsvolts.com/magazine/article/April2017\\_Gilbert-Erector-Ferris-Wheel](http://www.nutsvolts.com/magazine/article/April2017_Gilbert-Erector-Ferris-Wheel).



**FIGURE 1.** The Erector Set manual had dozens of illustrations of exciting things to build like a beam engine or this Bascule bridge.



**FIGURE 3.** I recently built this motorized Ferris Wheel from a vintage set and added flashing LEDs and a carnival sound track.

vintage sets are still available online and in antique stores for a reasonable price. Some are in poor condition with rusty parts that need some TLC, but occasionally you find primo sets that are ready to use or that you just have to get. Recently, I saw a beautiful No. 12 1/2 on eBay that I was sorely tempted to buy. We'll see.

## Building the Ferris Wheel

Okay, enough of the reminiscing. Let's get on with how I built the Giant Ferris Wheel and added some modern components to enhance it, like an Arduino microcontroller. **Figure 3** shows the end result.



**FIGURE 2.** This set was one of their best sellers: the No. 8 1/2, which I had when I was a kid and is still available on eBay.



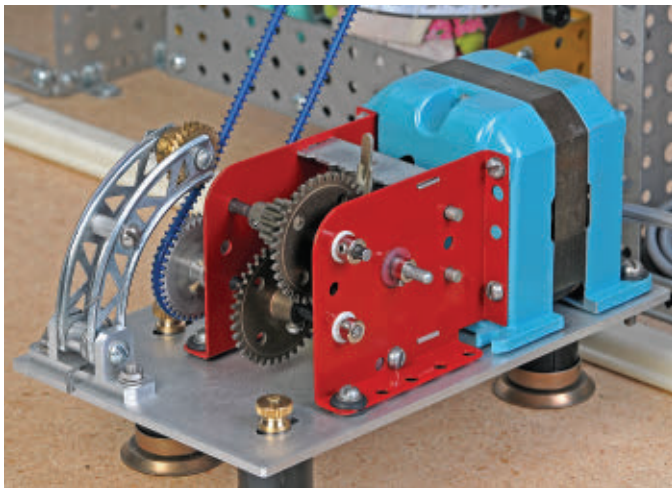
**FIGURE 4.** Here are several typical parts including a 5" girder, 5" curved girder, 3" disc wheel, and a cool electromagnet.

After several months of searching online for a suitable No. 8 1/2 set, I found a decent looking one for \$150 on eBay. When it arrived, I was a little disappointed because the steel girders were dulled by oxidation; definitely not as bright as they looked in the listing. However, after spending the day with a package of #0000 steel wool and a lot of elbow grease, they polished up quite nicely.

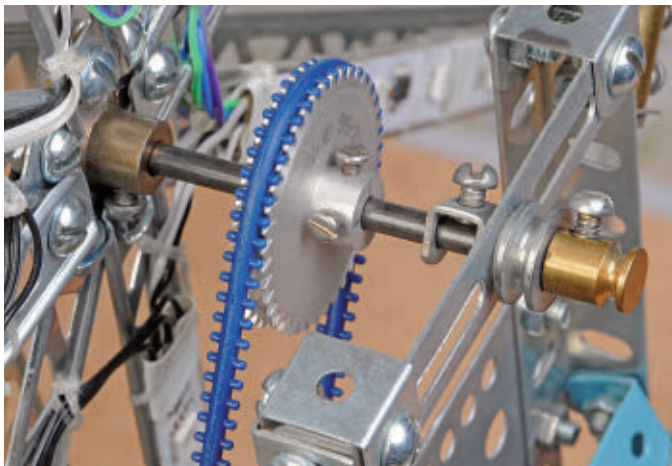
**Figure 4** is a sampling of the restored parts. Notice the cool lifting magnet in the upper center. Its resistance is 12 ohms and it operates on three batteries: 4.5V/12 ohms = 375 mA.

Another favorite component in the larger sets was the





**FIGURE 5.** The Erector all-metal 110 VAC type A49 motor has extra holes for various gear ratios and reversible operation.



**FIGURE 7.** The manual showed a drive belt made of string but it slipped, so I substituted a cogged belt. Sweet!

“Motor.” Refer to **Figure 5**. It was not a wimpy plastic DC motor powered by two AA cells, but an all-metal, multi-speed, reversible, 110 VAC power mover. Just add a pulley and you had a winch that could do some real work.

The motor that came with the set I bought had some chipped paint, was a little grimy, and needed a new power cord, so I took the opportunity to completely refurbish it to its former glory. Listening to the sound of its newly oiled gears going round and round was music to my ears. Plus, it was reversible with the movement of a small handle that engaged a sequence of gears. I remembered back to when I first learned how to set up gear ratios in order to slowly raise a lift bridge or rotate a Ferris wheel.

As some of you might recall, the 8-32 nuts that came with the sets were square instead of hexagonal and sometimes were painful to hold and tighten on the bolts. The square edges would jab into your fingertips (**Figure 6**). Therefore, my first step was to set aside the original nuts



**FIGURE 6.** The set's 8-32 square nuts fit nicely into tiny spaces, but are painful to tighten with your fingers compared to hex nuts.

## PARTS LIST

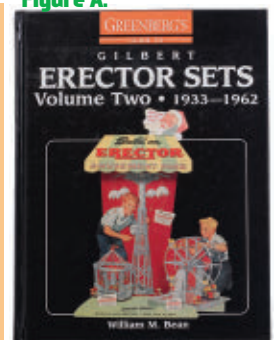
DESIG	COMPONENT	SUPPLIER
BELT	Cog Drive Belt, 33"	PIC Design, FA-336
BOX1	Chassis, 3x4x5, Aluminum	Digi-Key, L104
BRD1	MP3 Player Shield	SparkFun, DEV-12660
C1	Capacitor, 100 mfd 25V	Digi-Key, P10269
CHAR1-9	Characters, Cartoon	Toys-R-Us
D1	Diode, 1N4001	Digi-Key, 1N4001DICT
LED1-6	Tricolor LED Strips	RadioShack, Closeout
NUTS	3/8-16 Steel Nuts	Home Depot
PS1	Power Supply, 12V 1A	RadioShack, Vintage
PULLEY1,2	No-Slip Pulley, 40 teeth	PIC Design, MGP2-40
Q1	NPN transistor, 2N3904	Digi-Key, 2N3904FS
R1	Resistor, 10K	Dig-Key, 10KQBBK
RLY1	Relay, 12V, DPDT	RadioShack, 2750043
S1	Rotary Switch, Four-position	Junk Box
S2	Toggle Switch, SPDT	Digi-Key, CKN1023
S3,4	Microswitch, Roller, SPDT	RadioShack, 2750017
S5	Pushbutton, N.O.	RadioShack, 2751556
SET1	Erector Set, #8 1/2, c1950	eBay
SPKR	Powered Speaker/Amp	RadioShack, Vintage
SR1	Slip Ring, Six-conductor	eBay, China
U1	Arduino Micro w/Headers	Adafruit

and buy several gross of modern small pattern hex nuts. No more sore fingers!

It took me a couple of days to build the basic Ferris wheel structure and get it spinning freely on the axle. The manual showed a pulley mounted on the motor and another larger one on the wheel axle, connected together by a long loop of string that was supposed to act as a belt. No joy! The string slipped on the pulleys no matter how much tension was applied. Then, I recalled a toothed belt and pulley arrangement that I had used during my

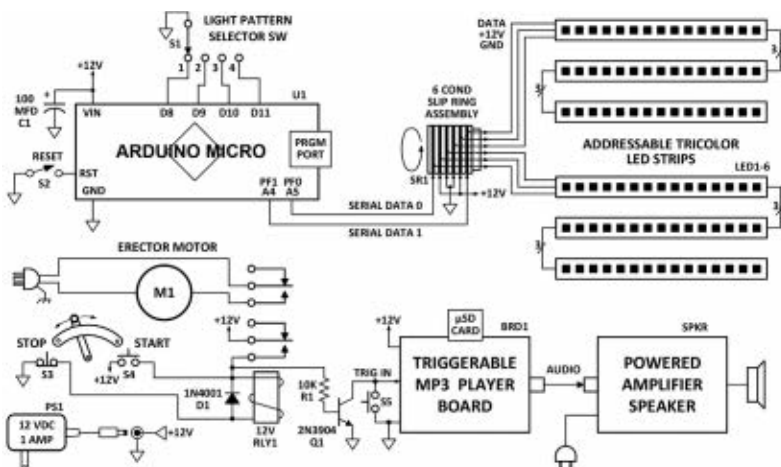
A handy reference book is *Greenberg's Guide to Gilbert Erector Sets* shown in **Figure A**. It has full color pictures of every set and even has a listing of each and every part in all the different sets. For example, in 1950, a No. 10 1/2 set contained (52) 10" girders, (26) 5" curved girders, (4) 7" axles, (4) 3" disc wheels, (271) 8-32 square nuts, and hundreds of other parts. If you're interested, Wikipedia has a brief history of Erector Sets on their website.

**Figure A.**





**FIGURE 8.** My start/stop lever switch emulates the manual clutch lever used on old-fashioned carnival rides.



**FIGURE 9.** An Arduino microcontroller sends 24-bit serial data to the tri-color LED strips to produce various flashing patterns.

aerospace days. So, I looked them up in the *PIC Design* catalog.

Yikes. The price was \$29 for the belt and \$54 for two mating toothed pulleys. I debated whether I should simply use a belt made of o-ring type rubber, which probably would have worked fine. **Figure 7** clearly illustrates that I caved in and spent the money for the cool looking blue belt. The final rotation rate was about 5 RPM.

## Adding Bells and Whistles

**Figure 8** shows how I used several 2-1/2" curved girders to build a start/stop lever on the control box. The action sort of duplicates the actuation of the clutch mechanism in a real Ferris wheel, instead of using a modern toggle switch.

**Figure 9** is the schematic of the whole system including the latching relay circuit for the motor. The start/stop lever also triggers the newly added MP3 player board to start the carnival music.

A while back, someone suggested that a carnival-like sound track would be a good addition, so I did it. The music file came from the Internet and was transferred to a SparkFun triggerable MP3 player board using Audacity. The MP3 board was not made to directly drive a speaker, so I dug out an old RadioShack powered speaker that I've had for

years and mounted it on the rear corner of the baseboard. I never throw anything away because you never know when you'll need it! Although it took quite an effort to add the sound track to the project, I'll have to admit that the carnival music made a big difference. It was just like you were walking down the midway.

One day when I was in RadioShack, I stumbled across a bunch of addressable tri-color LED light strips for a vastly reduced price on the closeout rack. I figured the lights would look great on the Ferris wheel, so I attached them around the rim and down the spokes with lacing cord. The

strips require a 24-bit serial data stream to control the brightness and color of the tri-color LEDs. I chose an Arduino to do the job because it was easy to program, had enough memory, and was pretty small.

**Figure 10** is an interior photo of the newly added chassis box containing the Arduino and MP3 player. As you can tell, the project was rapidly spiraling out of control. The bells and whistles were taking over.

Now the big question ... how to get power and data to the rotating LEDs? A slip ring rotary joint was the answer. When I was working in aerospace, mil-spec slip ring assemblies were very expensive. However, a quick look on eBay saved the day. For under \$10, I found a commercial six-conductor plastic unit from China, and it got here in less than a week.



**FIGURE 10.** The red SparkFun triggerable MP3 board plays carnival music to give the project the feel of the midway.



I can't help but wonder how they can make a profit.

**Figure 11** shows the slip ring assembly installed on the main axle of the wheel. Refer to the schematic for the wiring. Two of the conductors were used for the serial digital signals from the Arduino to the LED controller chips.

Actually, I think I spent more time playing around with different lighting patterns than anything else in this project. I had fun generating a number of sketches that produced hypnotizing patterns of colored lights. It was hard to stop playing with it. (BTW, the Arduino code is available at the article link.)

I won't go into any detail about the code here because it's pretty simple and most of it was "example code" for the discontinued LED strips from the Shack. SparkFun and Adafruit have much better addressable LED strips now, plus example code to run them.

## Bells and Whistles, Phase 2

I thought that adding the lights was the end of the project until one evening when I was proudly showing it to my wife, she commented, "That's real nice but what it needs are some cute little characters riding in the cars." Garf! Unfortunately, she was right. So, I headed to the toy store and found eight properly-sized candidates, including Big Bird.

I glued them down in the front of the cars so you could see their faces as they went round and round. The only problem now was that the cars were tilted forward, not level. I hate it when things are not level. So, I added a number of hefty steel nuts in the rear of each car until they were all perfectly flat. Success!

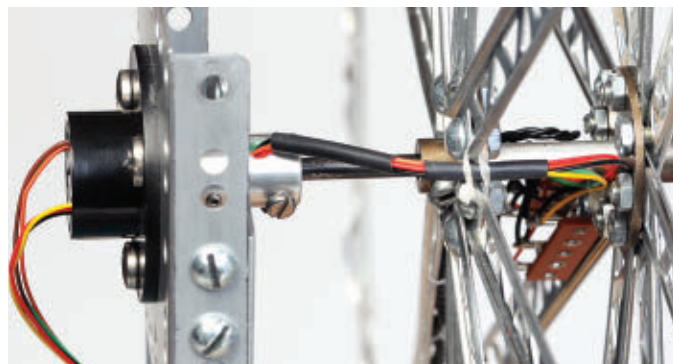
I started the motor and the "cute" little characters went half way around and promptly came to a dead stop. What the heck? After an intensive 10 minute failure mode analysis, I discovered the core problem. Each car now weighed a different amount because of the different sizes of the characters and quantity of steel nuts. This situation unbalanced the big wheel and overwhelmed the torque of the Erector motor. The answer: more steel nuts. Finally, it was perfectly balanced and rotated very smoothly. Done, done, and done!

A short low resolution video of Big Bird taking a ride can be seen at [http://youtu.be/K5JuUVva\\_QQ](http://youtu.be/K5JuUVva_QQ).

All my friends were impressed with the operation and the kids were mesmerized by the flashing lights and carnival music. It was well worth the effort. However, my wife still had one more thing to say about the project. "I love how it turned out," she said, "but where are you going to find the room to store it?"

## Final Thoughts

It was fun working again with the Erector Set metal



**FIGURE 11.** A six-conductor slip ring assembly from China transmits power and data to the LEDs on the rotating wheel.

girders, axles, and especially the motor. It's sad that sets like these are not economically feasible anymore. I think that kids are missing out with the snap together, no-thinking-required plastic kits of today. Fortunately, vintage Erector sets are still available.

I think that buying one and building a project like the Ferris wheel or a Bascule bridge with your kids or grandkids would be a kick. I hope you will truly consider it. **NV**

Best Value  
prototype PCBs online  
Turnkey Assembly available!

Serving over 10,000 PCB  
Designers and Buyers  
since 2003

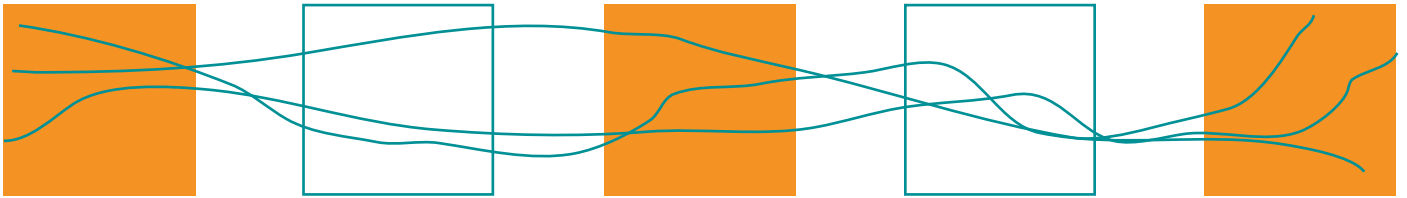
Made in USA  
[www.PCBFabExpress.com](http://www.PCBFabExpress.com)

☎ 408-522-1500

[Support3@pcbfabexpress.com](mailto:Support3@pcbfabexpress.com)



# Synchronizing a 60 Hz Crystal Controlled Oscillator to the Line



**Customers sometimes require that our crystal controlled digitally synthesized 60 Hz oscillators be synchronized to the 60 Hz line. Synchronization is essential when you need to switch a load between two independent sources, such as the 60 Hz line and a 60 Hz oscillator (with a power amplifier). A simple relay can be used to switch between them. However, the sources must be synchronized. Otherwise, large potentially damaging transients will occur.**

**F**or example, suppose the oscillator and the line were the same voltage, but 180 degrees out of phase. The line would be at voltage +V, while the oscillator would be at -V. The load would see a transient voltage of 2V as the relay switches between sources. (For a line voltage of 115 Vrms, V can be anywhere from 0 to 160 volts, depending on when in the cycle the switching occurs. A 320 volt transient is possible). This stresses both the relay and the load, possibly to the point of doing damage.

If the 60 Hz line and oscillator output were synchronized and of the same magnitude, the switching would occur between two identical voltages with no transient at all. The conclusion is that if you want to switch a load between sources, they should be synchronized and of equal magnitude. This article shows how to achieve synchronization (identical frequency and phase) of the waveforms.

The oscillator frequency is determined by a quartz crystal, and the line frequency is controlled by the utility company. In some locations, this may vary by as much as  $\pm 0.5$  Hz over the course of a day. The question is: How can we synchronize the crystal oscillator to the line? Almost by definition we cannot change the frequency of

the crystal, yet it must be adjusted to the line frequency which may be anywhere from 59.5 Hz to 60.5 Hz.

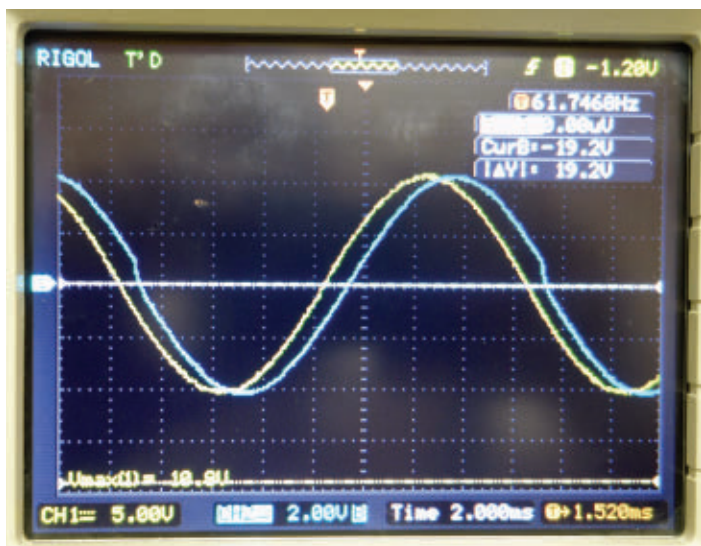
It is possible to “pull” a crystal off of its nominal frequency by adding reactive components into the resonant circuit. This technique might be able to pull the crystal a few kHz off nominal.

Assuming the 15.36 MHz crystal can be pulled  $\pm 5$  kHz, that represents about  $\pm 325$  PPM (or  $\pm 0.02$  Hz) which is not enough to cover a potential discrepancy of  $\pm 0.5$  Hz.

An effective technique used here is to distort the waveform so it fits in the allotted cycle time of the 60 Hz line. For example, consider the waveforms in **Figure 1**. The yellow trace is the (simulated) line signal running at 61.74 Hz, simulating a very high line frequency. The blue trace is the 60.00 Hz output of the crystal controlled oscillator, synchronized to the fast running line.

The synchronization circuit restarts the 60 Hz (blue) waveform at (nominally) the zero crossover of the 61.74 Hz waveform and effectively chops off a small segment of the oscillator’s 60 Hz waveform. This forces the 60 Hz oscillator waveform into the same time period as the 61.74 Hz line signal, at the expense of some distortion.

**Figure 1** shows the restart of the 60 Hz (blue)



**FIGURE 1. High line frequency.** Line frequency (yellow) is 61.746 Hz. Oscillator frequency (blue) is 60.007 Hz. Oscillator frequency (blue) restarts at about 190 degrees of the (yellow) line waveform. A segment of the oscillator output has been chopped off to make the oscillator period equal to the period of the line.

waveform occurs at about 190 degrees of the (yellow) line waveform.

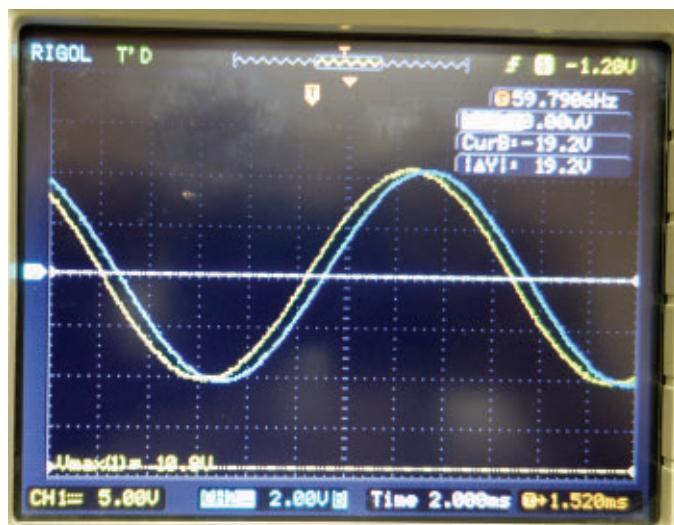
**Figure 2** shows a more normal situation where the line frequency is 59.79 Hz, the oscillator frequency is 60.007 Hz, and the difference is 0.217 Hz. There is no discernible distortion, and the signals are synchronized but slightly out of phase. The phase difference is allowed here to more clearly distinguish the waveforms, and will be corrected later.

**Figure 3** shows a low line frequency of 57.835 Hz. The oscillator frequency is 60.007 Hz. The oscillator must run more than 360 degrees to fill the time period of the line. It also restarts at about 190 degrees.

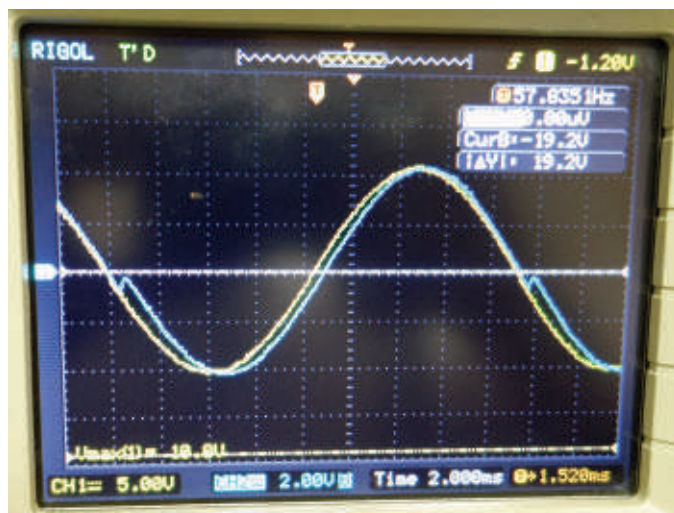
These photos illustrate that synchronization using this method creates distortion that becomes negligible for small differences in frequency. Furthermore, the distortion occurs when the voltage is near zero volts, suggesting that the energy of the distortion is very small.

Most practical cases resemble **Figure 2** where the distortion is essentially absent. The more extreme cases of **Figures 1** and **3** are shown to aid understanding of this technique.

A synthesized function generator was used to simulate the line voltage for these photos. This allowed us to easily change the line frequency for our test purposes. The photos show how the oscillator is synchronized to any line frequency by fitting the oscillator signal into the period of the line, either by chopping off a slight bit of the waveform or letting it run for a little longer than one cycle to fill up the period of the line. This may cause some



**FIGURE 2. Normal line frequency.** Line frequency (yellow) is 59.79 Hz. Oscillator frequency is 60.007 Hz. Difference is 0.217 Hz. Signals are synchronized with no discernible distortion.



**FIGURE 3. Low line frequency.** Line (yellow) is 57.835 Hz. Oscillator (blue) is 60.007 Hz. The oscillator runs for slightly more than a full cycle, thereby synchronizing to the slightly slower 57.835 Hz line.

distortion of the oscillator waveform, but under normal conditions the distortion is negligible.

The small phase difference between the traces is partly due to the zero crossing detector (look ahead to **Figure 6**) which does not switch at precisely the zero crossing. However, the phase difference can be removed in software as will be described shortly.

## Oscillator Operation

Before presenting the synchronizing software (in flowchart form), it is necessary to understand how this



Memory Location	Sine Wave Amplitude
\$f300	Amplitude for N=0
\$f301	Amplitude for N=1
\$f302	Amplitude for N=2
\$f303...\$f3fc	N=3 to N=252
\$f3fd	Amplitude for N=253
\$f3fe	Amplitude for N=254
\$f3ff	Amplitude for N=255

**FIGURE 4A. Memory contents per memory location.**

oscillator works. The synthesized oscillator uses NXP Semiconductors' (formerly Freescale and before that Motorola) eight-bit MC9S08SE4 microcontroller.

This processor contains an index register called 'h:x' where h and x are independent eight-bit registers. (The semicolon between h and x indicates that these registers are concatenated. They are read as one 16-bit register, even though they are two independent eight-bit registers.) They are used to point to locations in memory that contain a sine wave lookup table.

In our case, h:x has been chosen to start at \$f300 ('\$' means a hex number) and end at \$f3ff. The h register remains fixed at \$f3, and the x register cycles through 00, 01, 02,...FE, FF, 00, 01, etc. Each memory location contains a point on the sine wave. There are 256 points (0-255), and each step is  $360/256 = 1.40625$  degrees. A table of memory locations is shown in **Figure 4A**.

The amplitudes in the table are calculated from the equation:

Sine Wave Amplitude =  $128 + 127\sin(1.40625 \times N)$  with  $0 \leq N \leq 255$

For example, for N=0, Amplitude = 128 (angle = 0 degrees)

N=64, Amplitude = 255 (angle = 90 degrees)

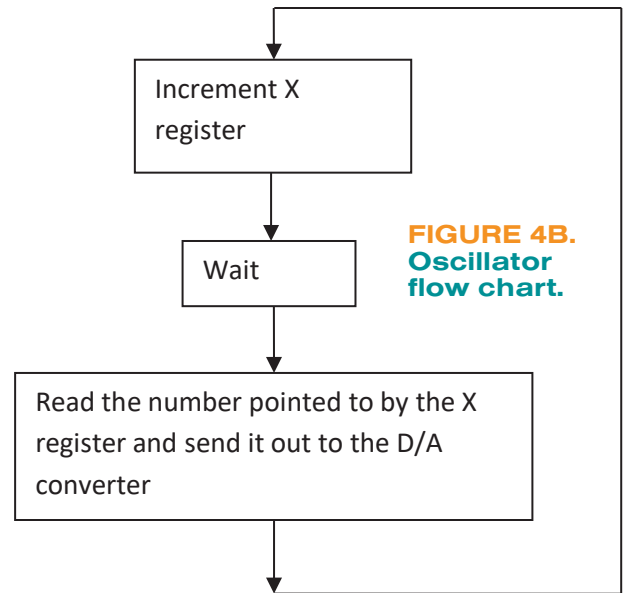
N=128, Amplitude = 128 (angle = 180 degrees)

N=192, Amplitude = 1 (angle = 270 degrees)

This represents a sinusoid with a zero line of 128, a positive peak of 255, and a negative peak of 1.

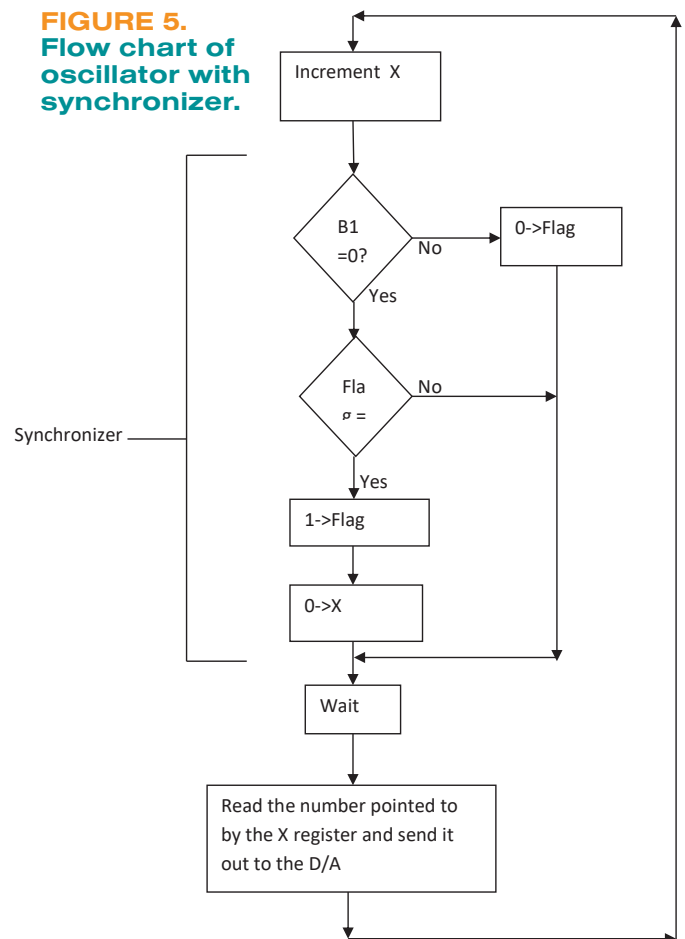
For purposes of this discussion on the basic operation of the oscillator, it is important mainly to understand that cycling through the x register points to a complete sine wave cycle. We are now ready to understand the oscillator as represented in the flow chart of **Figure 4B**.

Assume for the moment that the Wait time is zero (no wait). The fastest possible sine wave (for this processor and this software) will occur. A point on the sine wave will be sent to the D/A (digital-to-analog) converter. The X register will be incremented to the next point in memory. It will be sent to the D/A converter and so on. The X register will race around from \$00 to \$FF repeatedly, thereby repeatedly describing complete cycles of sine



**FIGURE 4B. Oscillator flow chart.**

**FIGURE 5. Flow chart of oscillator with synchronizer.**



waves at the output.

Now, suppose we introduce a Wait time of exactly 65.10416667 microseconds (how to get this exact delay will be told momentarily). Each new point on the sine



wave will be sent to the D/A converter every 65.10416667 microseconds. Since there are 256 points on the sine wave, 256 points will take  $256 \times 65.10416667 = 0.0166667$  seconds, which is the exact period of 60 Hz! In other words, this Wait time produces a sine wave with a frequency of exactly 60.000 Hz.

The Wait box in the flow chart creates the delay by waiting for an interrupt from the timer system of the processor. Using a 15.36 MHz crystal, this processor will generate a bus frequency of  $15.36 \text{ MHz} / 2 = 7.68 \text{ MHz}$ .

The timer system is thereby driven from a clock with a period of  $1 / 7.68 \text{ MHz} = 130.2083333 \text{ nanoseconds}$ . The internal timer counter is programmed to be a modulo 500 counter so that it overflows and jumps back to 0 at a count of 500. Note that  $500 \times 130.2083333 \text{ ns} = 65.10416667 \mu\text{s}$ .

The processor is programmed to interrupt at every overflow of the modulo 500 counter. The resulting interrupt service routine lets the processor pass the Wait state and go on to create a point on the sine wave every 65.19416667  $\mu\text{s}$ .

If a modulo 600 counter were used instead of the modulo 500 counter, interrupts would occur at  $600 \times 130.2083333 \text{ ns} = 78.125 \mu\text{s}$ , and the output period is  $78.125 \mu\text{s} \times 256 = 0.02$  seconds, or 50 Hz exactly.

Using a modulo 75 counter produces exactly 400 Hz. Thus, the three main power frequencies can be precisely synthesized with a 15.36 MHz crystal, which is a stock part with many distributors.

It is important to recognize that for 60 Hz, the interrupts are spaced 65.18416667  $\mu\text{s}$  apart. The time necessary for reading the table, sending the data out to the D/A converter, and incrementing the index register all take place during the time between interrupts and subtract from the time in the Wait state.

For example, suppose it takes a total of 30  $\mu\text{s}$  to read the table, send the data out, and increment X. Then, 30  $\mu\text{s}$  will already have passed and the system will stay in the

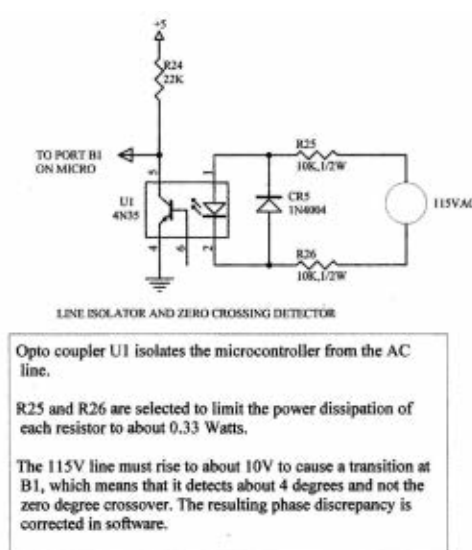


FIGURE 6. Optocoupler circuit.

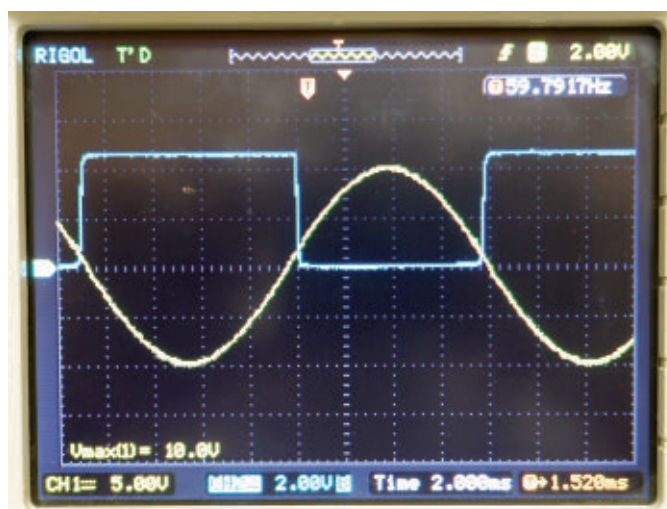


FIGURE 7. Optocoupler output (blue) vs. simulated line voltage (yellow).

Wait state for only another  $65.18416667 - 30 = 35.18416667 \mu\text{s}$  until the next interrupt occurs. In other words, part of the interrupt time period is used to do necessary tasks and not just waiting.

This system produces exactly 60 Hz, subject only to the accuracy of the 15.36 MHz clock. It remains now to adjust this frequency to the line frequency, which may deviate from exactly 60 Hz.

## Synchronizer Operation

The synchronizer adds some extra steps into the flow chart in

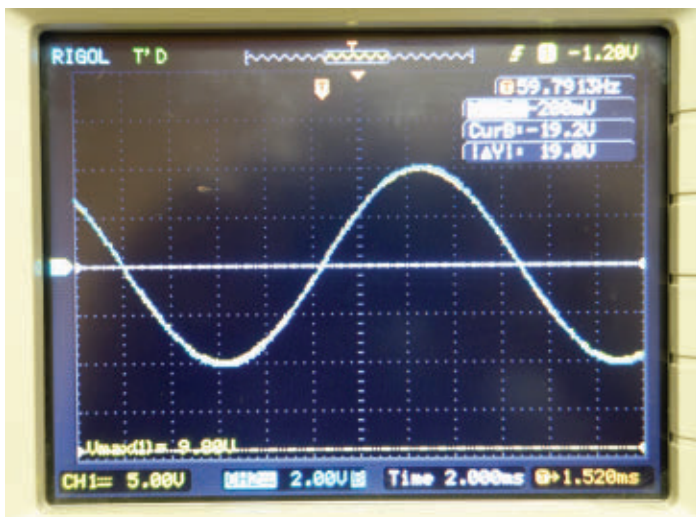
**Figure 5.** Its function is to restart the waveform on the positive going zero crossing of the line voltage. This either chops off a small segment of the 60 Hz waveform or extends the running time of the 60 Hz waveform; in both cases, to make it have the exact period of the line.

The line voltage cannot be connected directly to the microcontroller. It is connected through an optocoupler such as a 4N35. The circuit is shown in **Figure 6.**

The optocoupler — in addition to isolating the line — creates an inverted square wave replica of the line at port B1 of the micro. When the line goes positive, the square wave is logical zero; when the line goes negative, the square wave is logical 1. A photo showing the simulated line voltage and the signal at the B1 port on the microcontroller is shown in **Figure 7.**

Consider the case where no line voltage is applied. The signal at B1 will be continuously High. That is  $B1=1$ . The flow chart shows that under these circumstances, the main part of the synchronization logic will be bypassed, and only the flag will be repeatedly reset (with no effect). The oscillator will work just as previously described. In other words, the synchronization system has no effect if the synchronization signal (line voltage) is absent.

Consider next that the line voltage is connected. When the line voltage is negative,  $B1=1$  and the system behaves as described above.



**FIGURE 8. Signals are synchronized and in phase. Line (yellow) is at 59.79 Hz. Oscillator is at 60.007 Hz. Synchronizer uses 10 instead of 0 as the restarting memory location.**

The instant that the line voltage crosses the zero axis in a positive direction, the signal at B1 goes from 1 to 0. When the oscillator reaches the decision point for B1, it will select 'yes' for B1=0 and proceed downward in the flow chart.

In the next step (Flag=0?), the answer will again be yes because the flag had been reset while B1 was high.

In the next steps, the flag will be set and the X register loaded with 0 (which is the start of the sine wave table in memory). This effectively restarts the oscillator at the top of memory and at zero degrees of angle.

The oscillator will proceed to load that memory value

to the D/A converter, and then return to the top of the flow chart to increment X and again go through the synchronizer.

At that time, B1 will still be zero, so the process will proceed downward through the flow chart. Now when it reaches the decision point "Flag=0?," the answer will be 'no,' the flow chart will bypass reloading the X register, and no further action will be taken. This shows how the X register is reloaded only once on the falling edge of B1 (equivalent to the positive slope zero crossing of the AC waveform). When the AC line eventually goes negative, B1 will return to 1 and the flag will be reset, thereby preparing for another cycle.

To summarize, the synchronizer reloads the X register with its zero phase setting on every falling edge of the square wave at B1, thereby forcing the waveform to restart in synchronism with the line.

## Adjusting the Phase of the Synchronized Signal

Any phase relationship between the oscillator and line can be created by loading a memory location between 0 and 255 when the line crosses zero. The number selects at which point in memory and thereby what phase of the oscillator sinusoid shall coincide with the zero crossing of the line voltage. The photo in **Figure 2** shows a small phase difference between the line and the oscillator. This can be corrected by loading a number other than 0 at the falling edge of B1. Loading 10 instead of 0 corrects the phase discrepancy. The signals are now synchronized exactly in phase and with no visible distortion as shown in **Figure 8**.



**FIGURE 9. The '9S08-50/60/400' oscillator board.**

## Hardware and Test Setup

**Figure 9** shows the complete oscillator board. The microcontroller (MC9S08SE4) is the 28-pin DIP IC on the right side of the board. Further to the right is the six-pin optocoupler (4N35). I like using DIP packages wherever possible because they're accessible to humans. The lead spacing is 0.1 inch which is fine for hand soldering.

SMD packages start at 0.050 and go down to 0.01 lead spacing which is impossible to solder by hand. If there's any doubt about an IC, the DIP sockets allow quick and easy replacement without damage to the board.

The TO-220 packages on the left side of the board are voltage regulators. On a new board, I first remove all ICs and check the  $\pm 5V$  supplies. This assures that the ICs will get correct power and I don't wind up with a board full of dead chips due to power supply problems.

The six-pin programming header is located slightly





to the left of the MC9S08SE4. It is used to program the micro. The SMD IC in the center of the board is a four-channel D/A converter. It uses an SMD to DIP adapter board because a DIP version of this IC was not available. (It is a four-channel D/A converter which is used for three-phase operation of this oscillator).

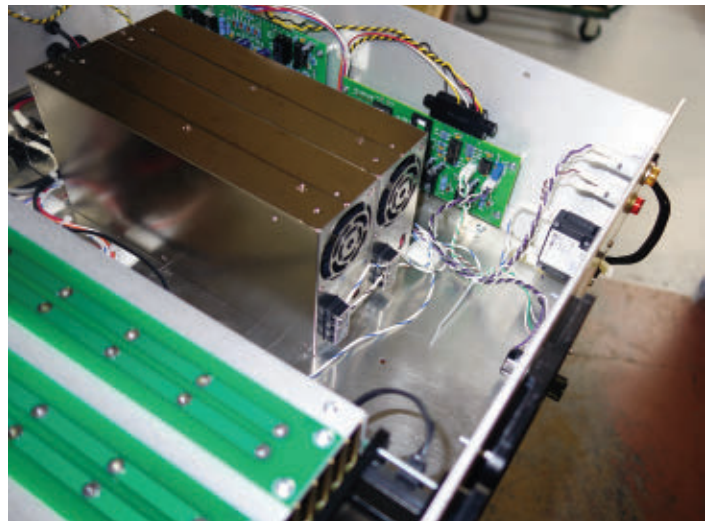
**Figure 10** shows the oscillator installed in a unit. The oscillator is mounted on the side wall and has been trimmed down in size to fit this mounting requirement. The driver PCB (printed circuit board) is also mounted on the side wall (to the left of the oscillator).

Two switching power supplies are in the middle of the photo. They provide the DC power for the unit. The FET (field effect transistor) bank is in the left foreground. It provides the amplification of the oscillator signal.

## Conclusion

Though the science of synchronizing one analog signal to other analog signal is somewhat well-trod ground, you can see how the introduction of digital systems has created new and surprisingly interesting challenges for synchronization.

I think it's important to note that the approach used in



**FIGURE 10.** Oscillator as installed in a Model 500S-CRH (500W AC current source). The oscillator PCB has been trimmed to fit on the side wall of the unit.

this article is just one of many possible ways to solve this particular problem, and I hope you have enjoyed this in-depth look at how we tackled it. **NV**



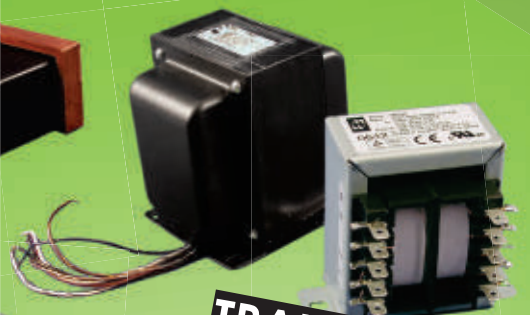
HAMMOND  
MANUFACTURING®

hammondmfg.com  
(716) 630-7030

CHASSIS



TRANSFORMERS



ENCLOSURES



Thousands of  
**IN-STOCK**  
enclosures and transformers to  
help make your ideas reality.



# Translational Reality Part 2: Control the World by Playing Pong

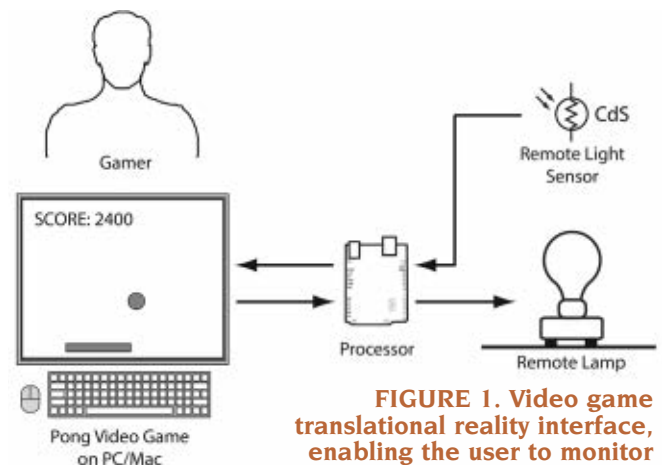
This article builds on “Translational Reality Interfaces” (*N&V*, March 2017) in which a toaster equipped with a translational reality interface enables a user to control an irrigation pump. In this article, I’ll show you how a classic video game can be adopted for use as a translational reality interface to enable the player to monitor and operate a remote device by simply playing the game.



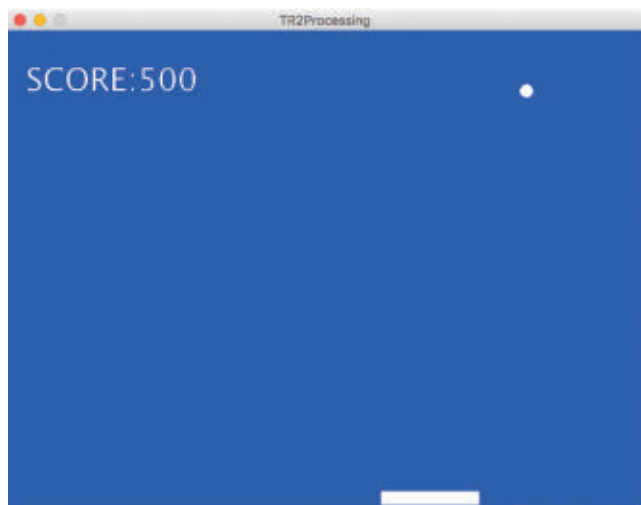
## Life is Just a Game

In Part 1, I mused that one day some of us will play our favorite video games and, in so doing, operate a remote device that could be hovering over a battlefield or in an operating room. Assuming the games are fun, that’s not a bad way to earn a living or at least some extra income. The aim of this article is to illustrate that the technology necessary for this work-play scenario is here today, in the form of translational reality interfaces.

Recall from Part 1, that translational reality is a closed loop, real time interface. In the example shown in **Figure 1**, the gamer or user plays a game of Pong on a PC/Mac and in doing so, controls a remote lamp that operates on a particular logic. The signal from a remote light sensor



**FIGURE 1.** Video game translational reality interface, enabling the user to monitor and control a remote lamp.



**FIGURE 2. Pong game interface, coded in Processing.**

causes changes in the video game, requiring the user to take action to avoid losing points and/or to gain points.

The two major elements in this example — the video game and the lamp — were chosen because of their simplicity. It's hard to find an example of a game simpler than Pong, and a lamp or LED is often the indicator of choice when debugging a microcontroller. Still, what the game and lamp represent is important. Substitute a more substantial game and more complex remote device, and you could have a system that's actually practical. For now, let's get the simple framework working so that you have an experimental platform.

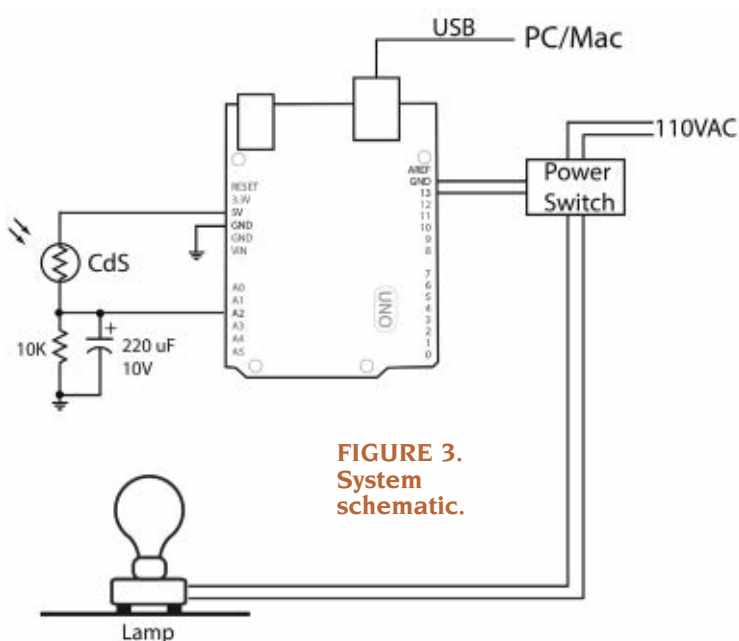
## Game Play

Pong was understandably the game that put Atari on the map. You simply slide the paddle horizontally to intercept the bouncing ball to keep it in play. Miss the ball and you lose points or the game.

I've added a few twists to the basic game to make it part of the translational reality interface. There are two states, identified by the color of the paddle and ball. When they're white (as in **Figure 2**), game play is normal. Points accumulate for successful intercepts, at the rate of 100 points/intercept.

However, when the ball and paddle turn red (signifying an alarm/alert condition at the remote site), the user must hold down the space bar while intercepting the bouncing ball with the mouse-directed paddle. Fail to hold down the space bar during the moment of impact with the paddle and the score is returned to zero. However, if the user manages to hold down the space bar while moving the paddle (with the mouse) under the bouncing ball, then the score increases by 1,000.

As currently configured, the gamer is allowed three



**FIGURE 3. System schematic.**

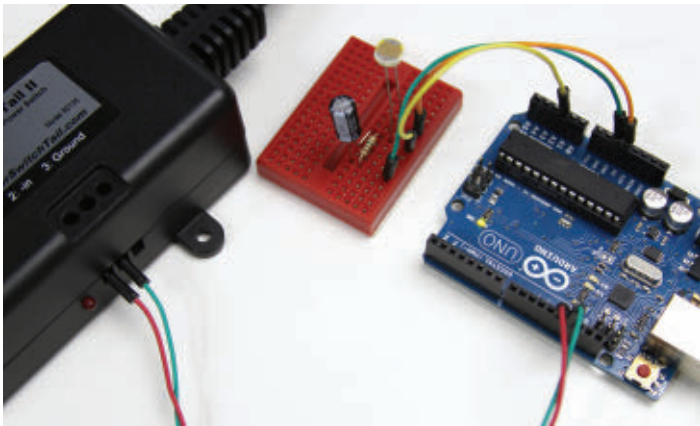
misses per session. However, as with the background color, size of the ball, and other aspects of the game, everything is customizable by changing a few lines of Processing code. Also, in case you're wondering, the lead photo shows the Pong game with the screen refresh disabled. If you haven't played Pong before, perhaps you can appreciate the simple geometry involved in game play. With a little practice, you should be able to move the paddle to where the ball is going before it gets there.

## Hardware

The hardware requirements for this project are minimal, assuming you have a desktop or laptop computer and an Arduino-compatible microcontroller. I use the popular Uno in this example. As shown in **Figure 3**, the Uno is connected to a CdS (Cadmium Sulfide) photoresistor, a 10K resistor, and a 220  $\mu$ F 10V electrolytic capacitor. In the dark, the resistance of the photoresistor is several million ohms. Resistance drops to 1K or lower with illumination.

When the photoresistor is illuminated, the 220  $\mu$ F capacitor is charged by the Arduino's 5 VDC source. When illumination is withdrawn, the photoresistor is essentially an open circuit, and the voltage across the capacitor and resistor combination follows the expected 1/RC time constant. The decaying voltage level is read by analog pin A2 on the Arduino. On the digital side of the

Translational reality is a real time, closed loop system in which a user operates a familiar first device (e.g., a video game) and, in so doing, both monitors and controls a second device. User feedback is appropriate to the first device — game play and scoring, for example — but depends on the operation of the second device.



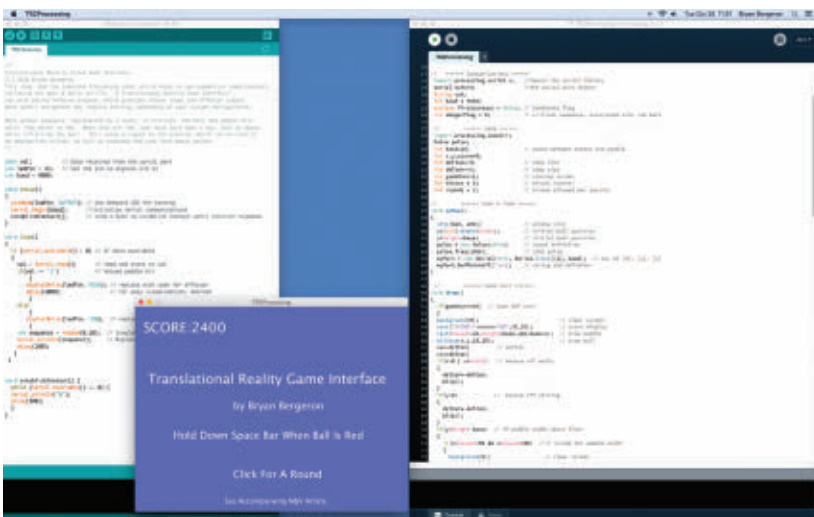
**FIGURE 4. Peripheral hardware components: PowerSwitch II, CdS sensor, RC circuit, and Uno.**

Arduino, the voltage across pin 13 to ground is used to signal a PowerSwitch II (\$29, [Sparkfun.com](http://Sparkfun.com)) which switches on 110 VAC whenever pin 13 is HIGH. **Figure 4** shows the three peripheral hardware components.

The use of the PowerSwitch II is simply for flexibility. If you read Part I of this series, you know that there are a variety of second devices that can be controlled by the interface. With the PowerSwitch II, you could plug in and control just about any home appliance or light fixture, from a table lamp to a blender or a toaster. Assuming we keep it simple and use a lamp, the photoresistor will work fine as a monitoring sensor. If you opt for a different output device, then you'll have to modify the sensor electronics accordingly. Given the simplicity of the design, that shouldn't be an issue.

## Software

Working with both Processing and an Arduino



**FIGURE 5. Software development environments, with Arduino IDE (left) and Processing IDE (right). The purple startup screen is running in the lower left.**

```
char val;
int ledPin = 13;
int sensorPin = A2;
int sensorValue = 0;
int baud = 9600;

void setup(){
  pinMode(ledPin, OUTPUT);
  Serial.begin(baud);
  establishContact();
}

void loop(){
  if (Serial.available() > 0)
  {
    val = Serial.read();
    if(val == '1'){
      digitalWrite(ledPin, HIGH);
      delay(1000);
    }
    else
    { digitalWrite(ledPin, LOW);
    }
    sensorValue = analogRead(sensorPin)/50;
    Serial.println(sensorValue);
    delay(100);
  }
}

void establishContact() {
  while (Serial.available() <= 0){
    Serial.println("A");
    delay(300);
  }
}
```

**LISTING 1. Arduino source code.**

microcontroller lends itself to parallel software development as shown in **Figure 5**. Assuming you have the screen real estate, having both IDEs (integrated development environments) and the executing game visible simultaneously supports rapid development and debugging. What's more, because the languages are essentially identical, you don't need to rethink logic or control construction from one IDE to the other. That said, the Processing IDE is more advanced, with little niceties such as line numbering.

The program for the Arduino is shown in **Listing 1**. For clarity, the code is uncommented here. However, the source file at the article link is heavily commented.

As you can see, the code is straightforward, right out of the example files from the Arduino IDE. After defining the variables, the setup routine is run, which begins the serial communications and calls the subroutine to establish contact with the serial port on the Mac/PC.

The main loop continually checks the serial port. If it reads a "1" from the computer, then pin 13 — which is connected to the onboard LED — is set HIGH. The LED and relay within the PowerSwitch II are energized. The lamp connected to the PowerSwitch II is energized.



Conversely, if something other than “1” is received, then pin 13 is set LOW, and the LED and PowerSwitch II are de-energized.

Next, the external sensor attached to the sensor pin — analog pin A2 — is read, and the result scaled down by a factor of 50. The resulting value is sent over the serial port to the computer hosting the Pong game. This is the one point in the Arduino code that you might want to modify; either to change the divisor to reflect your environment and/or component specifications, or to increase the complexity of the pattern you want to monitor. For example, instead of a single analog value, you might want to monitor a string of different sensors read in a particular sequence.

Note that there is nothing special about digital output pin 13, other than it’s useful during debugging because of the onboard LED. If you’re concerned about excessive load on pin 13, then move the external device to a different output pin and change the code accordingly. The PowerSwitch II draws only a few mA, so the load on the pin isn’t an issue.

The Processing code (the highlights of which are shown in **Listing 2**) is necessarily more involved than the Arduino code. Because of space limitations, only the highlights of the code are shown, and there are no

Processing is a free open source interactive programming language with an integrated development environment (IDE). Because it’s designed to teach the fundamentals of computer programming in a visual context, it’s extremely easy to learn and use. As a bonus for Arduino users, if you know how to program the Arduino, you know how to program in Processing. Furthermore, there are libraries for communicating between a computer running MacOS/Windows/Linux and an Arduino microcontroller. To download and learn more about Processing, go to [processing.org](http://processing.org).

comments. As with the code for the Arduino, a fully documented version is posted at the article link.

Starting at the top, two libraries are loaded: the serial library and sound library. The serial library is required for communications with the Arduino, and the sound library supports the “blip” that occurs whenever the ball impacts a wall or the paddle.

Key elements of the setup routine are setting up the serial communications port on your desktop computer and establishing “\n” as the string terminator. Depending on your computer port configuration, you may have to change the serial list assignment from 1 to 0 or 2 or 3.

The draw routine is concerned with both game graphics and communications with the Arduino. It defines

```
import processing.serial.*;
import processing.sound.*;

void setup(){
  myPort = new Serial(this, Serial.list()[1],
    baud);
  myPort.bufferUntil('\n');
}

void draw(){
  if(gameOver==0){
    ellipse(x,y,15,15);
    x=x+deltaX;
    y=y+deltaY;
    if(x<0 || x>width){
      deltaX=-deltaX;
      blip();
    }
    if(y<0){
      deltaY=-deltaY;
      blip();
    }
    if(y>height-base){
      if(x>mouseX-50 && x<mouseX+50){
        background(0);
        deltaY=-deltaY;
        blip();
        score++;
      }
      if ((dangerFlag ==1)&&(keyPressed==true)){
        fillWhite();
        score = score + (dangerFlag * 10);
        myPort.write('1');
      }
    }
    else
    { if(dangerFlag==1){score = 0;}
      myPort.write('0');
    }
    dangerFlag = 0;
  }
  else
  { misses++;
    deltaY=-deltaY;
    blip();
    if (misses > rounds){
      gameOver=1;
      misses = 0;
    }
  }
}

void serialEvent( Serial myPort){
  val = myPort.readStringUntil('\n');
  if (val != null){
    val = trim(val);
    if (val.equals("2")){
      dangerFlag = 1;
      fillRed();
    }
    if (firstContact == false){
      if (val.equals("A")){
        myPort.clear();
        firstContact = true;
        myPort.write("A");
      }
    }
    else
    {myPort.write("A");}
  }
}

void gameOverSplash(){}
void mouseClicked(){}
void blip(){}
void screenBlue(){}
void fillRed(){}
void fillWhite(){}
}
```

**LISTING 2.**  
Highlights of the  
Processing  
source code.

Atari introduced Pong in 1972 as their first arcade game. It was one of those accidental successes in that it was the result of a training exercise assigned to Allan Alcorn by the co-founder of Atari, Nolan Bushnell.

the size of the ball (15x15 pixels), increments the X and Y position of the ball with time, causes the ball to reverse direction on impact with a wall or the paddle, and communicates with the Arduino. It writes a "1" to the

Arduino if the user is successful in rebounding a red ball, and changes the color of the ball back to white.

The *serialEvent* loop looks for a danger or alert condition from the Arduino, signified by a "2." In the current configuration, when the voltage across the 220  $\mu$ F capacitor drops significantly, the value read by A2 is eventually "2." The color of the ball and paddle are changed to red, and the player must hold down the space bar – or other key – when moving the paddle with the mouse to intercept the ball.

The subroutines called by the main program are listed by title only. The full definitions appear in the actual source code. There are subroutines for changing the color of the ball and paddle, for putting up the splash screen at first launch and after each session, and for playing sound effects.

## Taking It from Here

An obvious next step is to experiment with controlling and monitoring different devices through the Pong game interface. A heater element/thermistor pair would work, for example. Then, there's the extension of the monitoring code to include sequences and various scaling functions so that it's not simply the decayed signal from an LED or lamp. The point is to have a sequence or single value that requires human attention/decision making and reflect that in real time game play.

A next step is to increase the complexity of the game design. Add levels. Increase the level of difficulty. Add more interactive elements – and tie these to actions on the controlled device.

Are you going to be able to control a factory assembly line by playing Pong? Perhaps not. However, with more advanced game play and decision logic, just about any level of control is possible. Furthermore, even if you aren't interested in the various aspects of translational reality, learning how to use Processing to move data from the environment into a computer and back again is a skill worth learning. **NV**

# ALL ELECTRONICS

[www.allelectronics.com](http://www.allelectronics.com) Order Toll Free 1-800-826-5432

## 24VDC 10A POWER SUPPLY

Mean Well# SP-240-24.  
Fan-cooled power supply.  
Protection for short circuit,  
overload, over voltage,  
over temperature.  
199mm x 93mm x  
50mm. CE, TUV, cULus.

CAT# PS-2401

10 for \$35.00 each

**\$37<sup>50</sup>**  
each



## BATTERY TESTER

Test AAA, AA, C, D, 9V  
and button cell batteries.  
Easy-to-read, color-coded  
reading meter. Separate  
scale for button cells.

CAT# BC-22



**\$5<sup>95</sup>**  
each

## 12 VDC LEDs

Bright 5mm round LEDs with built-in  
resistors for 12Vdc, No external resistor  
required. Works well on 4-12Vdc.  
Dimmer at lower voltages.

COLOR	LENS COLOR	CATALOG#
RED	Red	LED-12R
BLUE	Milky-white	LED-12BW
WHITE	Milky-white	LED-12W
GREEN	Green	LED-12G
WHITE	Water-clear	LED-12WP
BLUE	Water-clear	LED-12BP

**50¢**  
each

10 for 45¢ each  
100 for 40¢ each



## 2-COND. WEATHER PACK CONNECTOR KIT, 12-10 GA

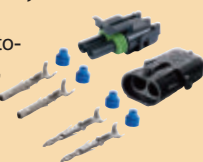
Affordable, waterproof  
connector, ideal for auto-  
motive, marine, racing,  
industrial applications.  
Includes all the parts  
necessary for 1 pair of  
2-conductor connectors.

Requires crimping tool for assembly.

CAT# WP2-10

**\$2<sup>89</sup>**  
each

10 for \$2.60 each  
100 for \$2.17 each



## LED MOTION SENSOR LIGHT

Turns on when it senses movement  
and turns off after 90 seconds.  
3.25" x 2.03" diameter. Uses three  
AAA batteries (not included).

CAT# MSL-04

**\$5<sup>95</sup>**  
each



## ANALOG MULTIMETER

3.5" wide. Scales for AC,  
DC Volts, DC mA & resistance.  
Includes test probes, one 9V  
and two AA batteries. Fuse and  
diode protection.

CAT# AMT-360

**\$10<sup>95</sup>**  
each



## 12 VDC 1 AMP POWER SUPPLY

4' long cord w/ 2.1mm coax  
power plug, center positive.  
cULus, FCC.

CAT# PS-121

**\$7<sup>20</sup>**  
each

10 for  
\$6.85 ea.



## SPST N.O. PUSHBUTTON

Attractive and functional dome-  
shaped momentary pushbutton.  
0.70" dia. black plastic bezel.  
0.45" red plunger. Extends 0.3"  
above mounting plane. Threaded  
bushing mounts in 0.53" diameter hole.

Solder terminals.

CAT# PB-138

**\$1<sup>50</sup>**  
each

100 for  
\$1.20 ea.



# Personal Radio Technologies

## Here are some communications options you may not know you had.

**C**ell phones have become our primary radio communication device. We have become addicted to our phones, as today, they are the do-all device for voice, text, emails, and Internet access. However, did you know that you actually have more radio communications alternatives? These options could come in handy when your cell phone is not as convenient, out of range, or not working during an emergency. Let me introduce you to the General Mobile Radio Service (GMRS) and the Family Radio Service (FRS).

### Remember CB Radio?

First, most of you have heard of the Citizens Band Radio Service. Back in the day in the 1970s before cell phones, CB was hugely popular, and many cars and trucks sported those ungainly eight foot whip antennas or the shorter magnetic mount coil-loaded antennas.

It was great for short range communications on the highway as it allowed anyone to report traffic jams, radar traps, and bad weather. It was a good way for truckers to pass the time by gossiping.

CB Radio is license-free, meaning there is no need to get a blessing from the Federal Communications Commission (FCC). Just buy the radios and talk. CB uses 40 channels in the 27 MHz spectrum, amplitude modulation (AM), four watts of power on AM, and 12 watts if you use the single sideband (SSB) version.

Range is generally restricted to several miles, but it is still very useful. With cell phones being ubiquitous today, almost no one (except possibly truckers) uses CB. However, it is still an option available to you.

Recently, I tried to see what activity there was on the CB channels. I located my old RadioShack handheld CB radio. I monitored the popular channels 9 and 19 and a few others, as well as initiated a continuous scan, but found no activity. I live in the suburbs away from a major highway so that may be the reason.

I made a couple of transmissions to generate some activity. Dead

silence. After several days of trying, I gave up. CB is dead at least in my area. Do any of you out there still use CB?

### GMRS is the Way to Go

The General Mobile Radio Service is a technology like CB in that it allows individuals to operate radios for personal or business use. Unlike CB, it does require a license. However, the license does not involve an exam and it is relatively easy to get. (More about that shortly.)

GMRS operation takes place in multiple channels in the 462 and 467 MHz UHF range. **Table 1** shows the main operating frequencies. Note that some of the channels are shared with the Family Radio Service (FRS) that I will cover later.

GMRS radios use frequency modulation (FM) with  $\pm 5$  kHz deviation and maximum power levels of five watts for handhelds and small base stations and repeaters, and 50 watts for larger base stations and repeaters. Those are effective radiated power (ERP) output levels.

ERP is the actual transmitter output power plus the effective power boost from any antenna gain. Antenna height is restricted to 20 feet above the ground or above the roof or structure to which it is attached.

As for transmission range — like all radio — it varies. At ultra high frequencies (UHF), transmission is line-of-sight (LOS) meaning direct from antenna to antenna. If the antennas do not “see” one another, transmission is blocked or at least is



Post comments on this article and find any associated files and/or downloads at [www.nutsvolts.com/magazine/article/April2017\\_OpenCommunication](http://www.nutsvolts.com/magazine/article/April2017_OpenCommunication).

greatly attenuated. Buildings and trees usually obstruct, scatter, or reflect signals. In a straight LOS path with minimal obstructions, the range for a five watt handheld transceiver is up to 10 miles or so. With obstacles at ground level, the range is only a mile or thereabouts. It depends so much on the environment.

The best thing you can do for maximum range is go to a higher point with a handheld or put up an antenna as high as allowed.

One solution available to GMRS licensees is a repeater. A repeater is a radio system that receives signals on one frequency and retransmits them on another frequency at a higher power level. Repeaters are usually located at high elevation points to reach out as far as possible.

With a 50 watt repeater, the range of a handheld now becomes more like 10 to 15 miles. Again, it is highly dependent on the territory. Some repeaters already exist in many locales that you may be able to use. Go to [www.mygmrs.com](http://www.mygmrs.com) to check what is available. Or, you can set up your own repeater if you have the need or inclination.

Getting an FCC GMRS license is easier than you think. No exam is required, but you must be 18 years of age or older and not be a foreign country representative. You can do it online at [www.fcc.gov](http://www.fcc.gov). Go to the Licensing & Database tab and then select GMRS. Follow the directions from there.

The first step is to get an FCC Registration Number (FRN). Once you get that, you can apply for the license with Form 605. The fee is \$70 and you submit it along with payment Form 159. Credit cards are accepted.

I know that is an over simplification, but you need to go through the process yourself. The term of the license is five years and it is renewable. The license permits you and your family members to use the radios.

## The Family Radio Service as an Option

If you don't want to fuss with

a license, you can still do something similar with Family Radio Service (FRS). FRS is a more limited form of GMRS. It uses fewer channels (see **Table 1** for the details). FRS is a personal communication capability and is license-free. Like CB, just buy the radios and talk. It uses FM with a more restricted deviation of  $\pm 2.5$  kHz. Power is limited to 0.5 watts and to handhelds only. No separate detached antennas, base stations, or repeaters.

Range is more limited but still useful if you are in a clear area. You should be able to get up to a mile or so under good conditions. On a recent walk in our neighborhood, my wife and I were easily able to talk clearly at just over a mile.

I have been using FRS for several years. The family has

Channel No.	Frequency (MHz)	Service	Power GMRS/FRS (watts)
1	462.5625	GMRS/FRS	5/0.5
2	462.5875	GMRS/FRS	5/0.5
3	462.6125	GMRS/FRS	5/0.5
4	462.6375	GMRS/FRS	5/0.5
5	462.6625	GMRS/FRS	5/0.5
6	462.6875	GMRS/FRS	5/0.5
7	462.7125	GMRS/FRS	5/0.5
8	467.5625	FRS only	0.5
9	467.5875	FRS only	0.5
10	467.6125	FRS only	0.5
11	467.6375	FRS only	0.5
12	467.6625	FRS only	0.5
13	467.6875	FRS only	0.5
14	467.7125	FRS only	0.5
15	462.5500	GMRS only	50
16	462.5750	GMRS only	50
17	462.6000	GMRS only	50
18	462.6250	GMRS only	50
19	462.6500	GMRS only	50
20	462.6750	GMRS only	50
21	462.7000	GMRS only	50
22	462.7250	GMRS only	50

**TABLE 1.** GMRS and FRS frequencies and power levels. Some additional frequencies are reserved for repeater operations. Some manufacturers use different channel numbers.



**FIGURE 1.** The Cobra CTX385 GMRS/FRS handhelds. These are typical of the types available.

adopted the radios to stay in touch in car caravans, on the beach, at the zoo, camping, and other places. They work great. They are good to have around in an emergency as many local emergency response units monitor FRS and GMRS channels.

If you want maximum range and flexibility of communications, definitely go for the GMRS license. Otherwise, for just occasional use over short distances, FRS is fine.

If you are serious about GMRS and FRS, I suggest that you get a copy of the FCC rules and regulations. You can find these in a document called Part 95 of the Code of Federal Regulations (CFR) 47. You can get a copy online at [www.gpo.gov](http://www.gpo.gov). You can print it out in pdf form or order a printed copy. GMRS is in Subpart A; FRS is in Subpart C; and CB is in Subpart D.

### GMRS and FRS Radios

GMRS and FRS radios are cheap and readily available at stores or online. Some radios are GMRS or FRS only, or could cover both. Some of the popular handheld manufacturers are Cobra, Garmin, Midland, Motorola, Olympia, and Uniden. Handhelds are usually sold in pairs. Prices vary from about \$30 to \$80. Makers of GMRS base stations and repeaters are BlackBox, Hytera, Icom, Kenwood, Motorola, Olympia, and Vertex. Prices range up to \$1,000. Antennas are extra.

Typical handheld retail vendors are RadioShack, Best Buy, Sam's, Walmart, and some office supply stores. There

are others, and lots of online sources.

I recently bought a new pair of Cobra handhelds (see **Figure 1**). They cover both FRS and GMRS channels, but FRS users can only legally use channels 1 through 14. This unit also has a weather radio inside that receives 10 of the NOAA weather stations in the 162 MHz spectrum. This is great for keeping track of bad weather. These Cobra units use NiMH rechargeable cells but three standard alkaline AA batteries can also be used.

Most of these radios have a feature called the continuous tone coded squelch system (CTCSS) and/or digital coded squelch (DCS). This capability lets you set up a private link where the audio is muted (that's squelch) until you get a unique tone or digital code first from some other radio.

CTCSS uses one of 38 inaudible tones below 300 Hz to wake up the squelch if you get a call. Other communications on the same channel will not be heard. It's the same with DCS where one of 83 digital codes is used instead of a tone.

Interestingly, the promotional material for these units says that the range is up to 23 miles. Don't believe it unless you are standing on a bare mountain top with a clear path to the other unit 20+ miles away. Believe a mile or so depending on the terrain.

### Other Personal Communications Options

But that's not all! You can still go with CB since radios are still available. Check for activity in your area. If you don't mind fooling with the required larger antennas, range can be much better than FRS.

Another possibility is the Radio Control (RC) service. If you want to operate remotely controlled items like planes, boats, cars, drones, or whatever, this is the service for you. It uses digital coding that is not permitted with GMRS and FRS. These devices operate in the 72-76 MHz range. No license is needed.

The high class way to go is to get a ham license. Amateur radio has been a personal communications service and hobby for over 100 years. You can use a huge range of frequencies and equipment technology including digital, TV, satellite, and microwave.

A license exam consists of questions on rules and regulations, operating procedures, electronic fundamentals, radio communications theory, circuits, and equipment. There are three classes: Technician, General, and Extra. This is the license to aspire to.

If you are new to communications, I urge you to try out one or more of the other services first as you learn more. **NV**

# Get “Patriot”ic with Lemos’ LoRa Radio Module

Sometimes (as a Design Cycle reader), you only need to know how something works to apply it to your own application. So, this month, I won’t force a predetermined project on you. Instead, we are going to dig through the datasheet, push the buttons, and turn the knobs. I have a box full of PW1-928 RF Data Transceiver Modules, a PW1-928 datasheet, and a PW1-928 evaluation kit. Let’s figure out how a PW1-928 works.

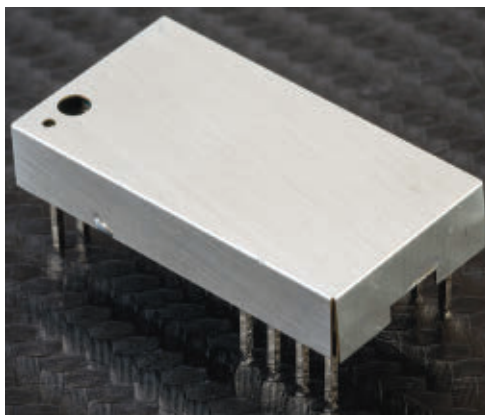
## The PW1-928 RF Data Transceiver Module

The Patriot PW1-928 is a Lemos International custom design. It can perform bi-directional digital data transfer at distances of over 300 feet. Transmitting in the 902 to 928 MHz frequency band, the PW1-928 generates +7 dBm into a 50Ω load. The PW1-928’s RX sensitivity is -112 dBm. Its output power is user adjustable.

The PW1-928 RF module is not a run-of-the-mill data radio. It uses the industry standard LoRa direct sequence chirp spreading protocol. In that the PW1-928 is a custom design, it makes sense that its packet transmission data structure is deemed proprietary. Normally, when something is labeled “proprietary,” it has unique features that the manufacturer/designer doesn’t want to fall into the competition’s hands. In the case of the PW1-928, the proprietary label is there to protect a simplified data structure implementation that helps to significantly improve the data transceiver module’s packet transmission reliability.

The PW1-928 is equipped with a microcontroller logic level Universal Asynchronous Receiver Transmitter (UART) serial interface. The PW1-928’s UART requires the appropriate RS-232 converter hardware to attach to true RS-232 ports that adhere to the RS-232 positive and negative voltage levels. Data transfer and configuration setup functions are performed using the PW1-928 module’s UART interface.

The PW1-928 is delivered with its clothes on. The clothed and shielded version of it is captured in **Photo 1**.



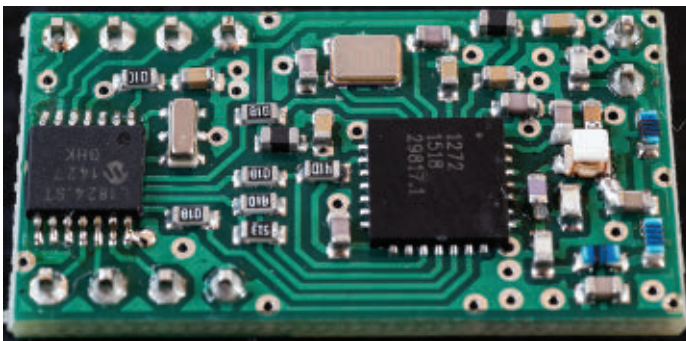
■ **PHOTO 1. The PW1-928 RF Data Transceiver Module comes in a shielded package that interfaces using standard 0.1 inch headers.**

Under normal circumstances, you won’t need to access the PW1-928 internal electronics. However, we are turning all of the knobs and pushing all of the buttons. So, just in case, **Photo 2** is provided for your viewing pleasure.

## PW1-928 Fundamentals

The PW1-928 uses a wideband “chirp” modulation that complies with the FCC part 15-247 regulatory requirements in the 902 to 928 MHz ISM band. An embedded PIC18LF1824 microcontroller running at 16 MHz controls the

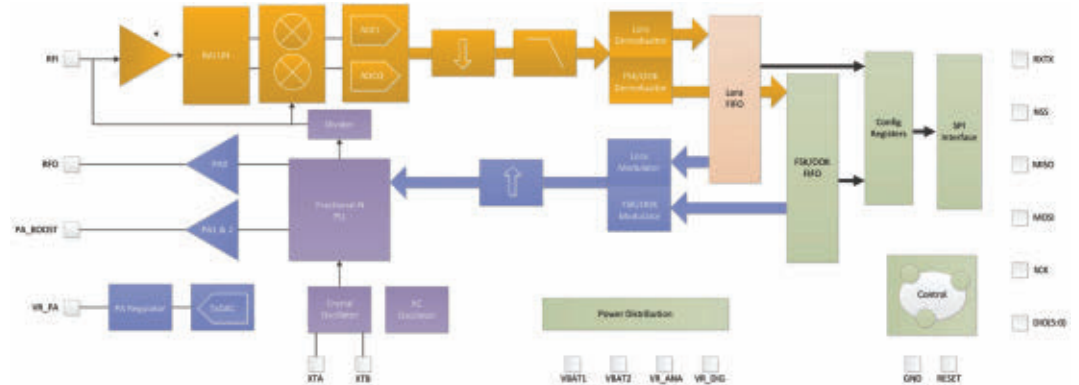
show. The PIC interfaces directly to the PW1-928’s Semtech SX1272 RF transceiver chip. The SX1272 transceivers feature a LoRa long range modem that provides ultra-long range spread spectrum communication and high interference immunity. Using Semtech’s patented LoRa modulation technique, the PW1-928’s SX1272 can



■ **PHOTO 2. Microchip’s PIC microcontrollers are in charge of thousands of devices. Here’s yet another one.**



achieve high sensitivity levels using a low cost crystal and associated low cost glue components. LoRa provides significant advantages in both blocking and selectivity over conventional modulation techniques. A block diagram of the SX1272's internal subsystems is represented in **Figure 1**.



■ **FIGURE 1.** The SX1272 is capable of speaking SPI but not RS-232 serial. The SX1272's RXTX pin is used to trigger an external RX/TX switch.

In addition to servicing the PW1-928's SX1272 RF hardware, the PIC burns cycles that support the high level protocol functions. Commands and data flow between the SX1272 and PIC via an SPI portal. Although the SX1272 speaks SPI, it is not equipped with a native RS-232 compatible serial interface. So, the PW1-928's PIC is used by the SX1272 as an RS-232 to SPI translator. The PW1-928 talks to the outside world on the RF side via the SX1272. The PIC's UART is used to transfer data and commands in the digital hardware domain.

**Figure 2** is a graphical depiction of the PW1-928 RF data transceiver module's I/O pin layout. The physical dimensions come in handy when you are laying out your printed circuit board (PCB) design that will incorporate the PW1-928. You will also need the pinout information contained within **Table 1**.

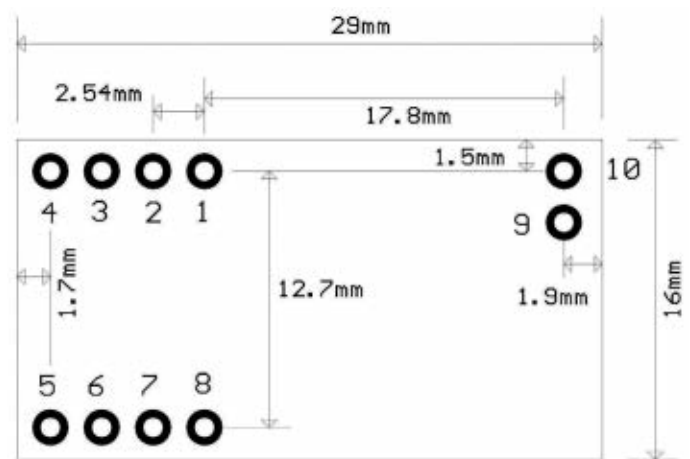
The contents of **Table 1** tell us a bunch. The PW1-928 is a 3.3 volt device. This power supply information helps with the choice of host microcontrollers. In this case, we know that if we plan to incorporate a five volt microcontroller, we may need to add 3.3V to 5.0V I/O logic level converters to the design.

In addition to the power supply information, **Table 1** says we will need a 3.3 volt compatible serial interface on our host device to drive the PW1-928's SI and SO pins. The PW1-928 SO pin is the module's serial output pin. In the case of the PW1-928, the SO pin drives received data (RXD in RS-232 lingo) from the radio subsystem. We should connect the PW1-928's SO pin to the host microcontroller's RS-232 serial port RXD pin.

The PW1-928's SI pin is equivalent to a microcontroller's RS-232 TXD pin. Data from the host microcontroller's TXD pin drives the PW1-928's SI pin. Two independent 64-byte FIFO buffers are placed in the PW1-928's serial interface TX and RX signal paths.

If necessary, we can also support a couple of transmit and receive visual indicators using the PW1-928's RXS and TXS I/O pins. The PW1-928's visual indication pins can also be used as hardware handshake signals.

From experience, we will certainly provide a means of



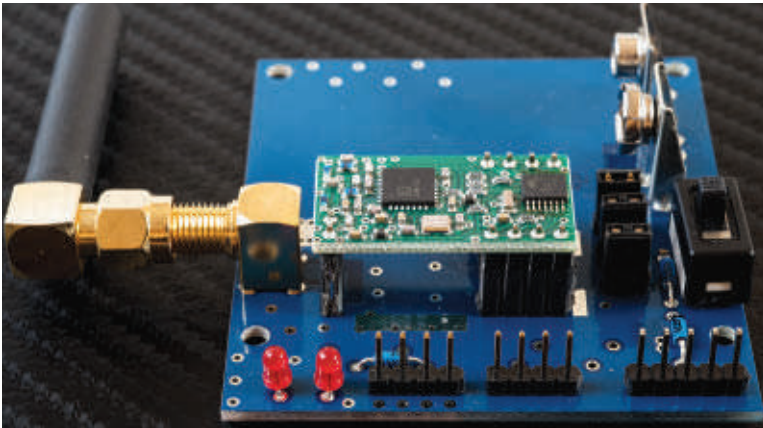
■ **FIGURE 2.** This is the template you will need to integrate the PW1-928 RF data transceiver module into your design.

Pin #	Name	I/O	Description
1	VCC	—	Supply Voltage (regulated 3.3v nominal)
2	SI	I	UART Receive Data Input. Input line for serial data (TXD)
3	SO	O	UART Transmit Data Output. Output for serial data (RXD)
4	n_RST	I	Reset line. This line acts as an active low hardware reset
5	RXS	I/O	RX Status. This line goes high when the receiver buffer contains data
6	TXS	O	Transmit status. This line is high when the TX buffer contains data or unit is transmitting. When low it indicates the TX buffer is empty
7	STBY	I	Standby. Pull low to enable the module. When high (floating) the unit enters a very low current standby state
8,10	GND	—	Ground
9	ANT	—	50-ohm RF Antenna Port

■ **TABLE 1.** Once your PW1-928 PCB pads are positioned, you can use this table to string in the traces.

externally resetting the PW1-928 via its n\_RST signal. Powering the PW1-928 and host microcontroller with a battery will most likely mean that we will at some point want to put the PW1-928 into low current mode. So, we'll plan for that in our design as well.

The PW1-928 can be forced into a low power standby mode using the PW1-928's STBY pin. In standby mode,

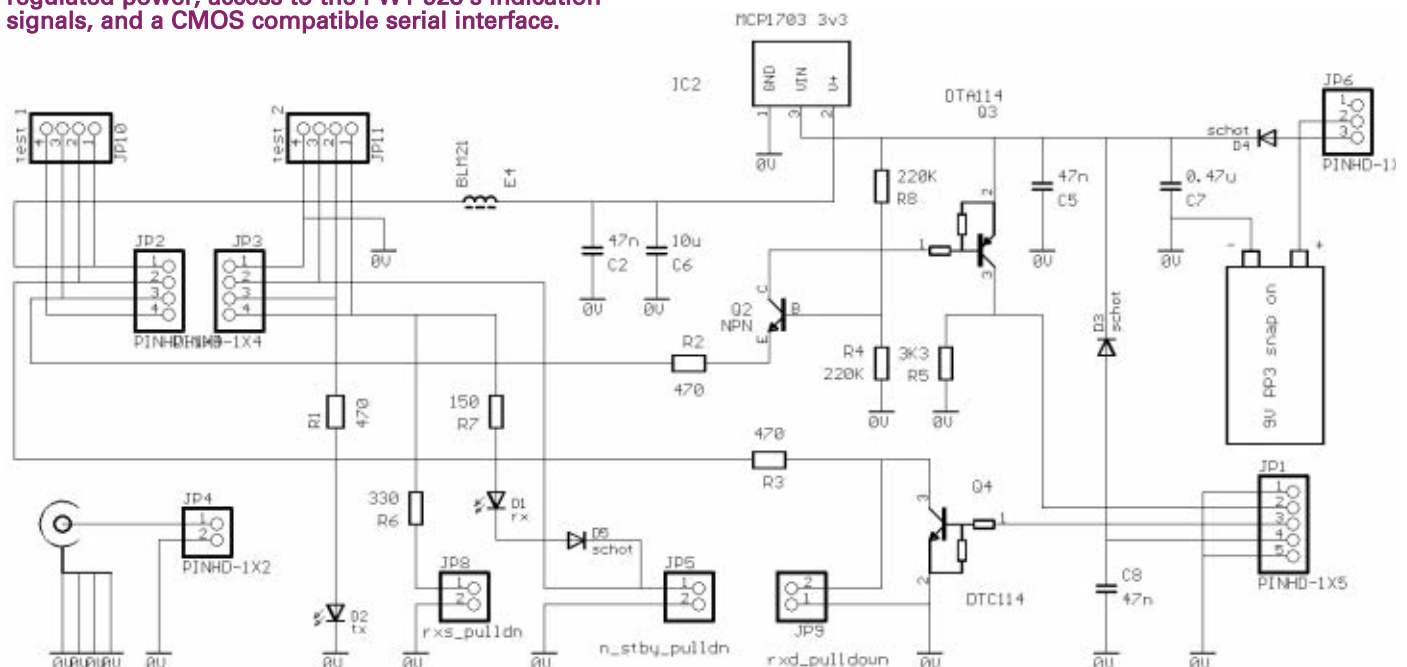


■ **PHOTO 3.** The PW1-928 dev kit provides a regulated 3.3 volt power rail and allows the user to access all of the PW1-928's RF and digital hardware facilities.

the PW1-928's RF section is completely shut down and the processor enters an idle state. In standby state, the receiver is always inactive. However, a transmit burst can be initiated in standby state by driving a specifically formatted data stream on the PW1-928's SI pin.

If the PW1-928 is in standby (STBY pin floating/high) and activity is detected on the PW1-928's SI serial input pin, the PW1-928 will temporarily come out of low power mode, transmit a single burst, and then revert back to standby. The format for a "standby transmit" is specific and consists of a 1 mS low-going pulse applied to the PW1-928 serial input followed by a 2 mS delay. Following the 2 mS delay, the data packet can be transferred and transmitted.

■ **SCHEMATIC 1.** The PW1-928 dev kit provides clean regulated power, access to the PW1-928's indication signals, and a CMOS compatible serial interface.



PW1-928 RF Data  
Transceiver Module  
PW1-928  
Development Kit  
Lemos International  
[www.lemosint.com](http://www.lemosint.com)

CCS C Compiler  
Serial Input/  
Output Monitor  
CCS  
[ccsinfo.com](http://ccsinfo.com)

By default, the PW1-928's UART packet format is fixed at one start bit, eight data bits, no parity, and one stop bit. The PW1-928's default baud rate is 9600 bps. Higher baud rates of 38.4 kbps and 57.6 kbps are also supported. In command/programming mode, the PW1-928 requires an RS-232 serial port baud rate of 2400 bps.

The PW1-928's embedded protocol automatically transmits the data that is driven on the SI pin. All encoding, transmitting, receiving, and decoding functions are handled by the PW1-928's internal processor. This automatic transmission capability allows the PW1-928 to stream data continuously at 9600 bps with no buffer overflows. The PW1-928 supports point-to-point communication links and broadcast topologies.

While in receive mode, the PW1-928's PIC constantly monitors the SX1272 looking for a valid packet receive flag. When the PW1-928's PIC detects a receive flag, the PIC downloads all the data for the packet into a buffer, checks for valid framing, address, and checksum and — assuming a success on all those tests — transfers the payload to a 64-byte FIFO.

A PW1-928 transmit sequence is initiated when the PIC's UART is found to contain a received byte. The PIC enters transmit mode and the SX1272 is put into the "prepare to transmit" mode. Received data that is driven through the PIC's UART is loaded into a different 64-byte FIFO for the 10 mS it takes the SX1272 to enter a stable ready state. When the SX1272 is deemed ready, up to 15 bytes of data are transferred from the transmit FIFO, formatted, and written to the SX1272.

Following the transfer of data from the transmit FIFO to the SX1272, the transmit command is issued and the SX1272 responds by transmitting a packet. The transmission process takes approximately 15 mS. While the transmit process is in progress, the PIC continues to read its UART and transfer any new data to the transmit FIFO. Data in the receive FIFO will be fed out through the UART during transmit, but new incoming packets cannot be processed until the transmit process has completed.

At the end of a transmission, if there is still any data in the transmit FIFO, another read FIFO/format/transfer/start transmit sequence is triggered. If the transmit FIFO is empty, the PW1-928 reverts to the receive mode.

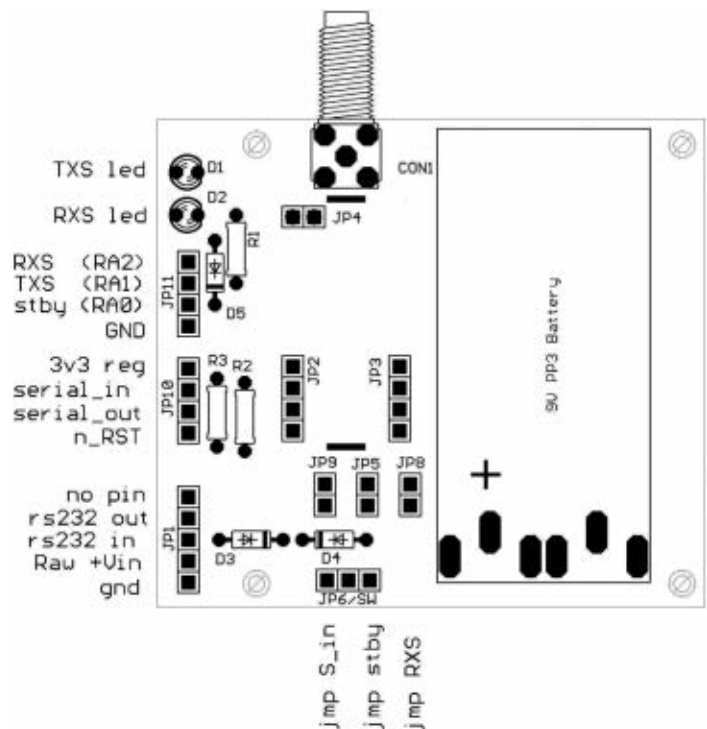
Each PW1-928 is capable of being addressed using a four-bit (1 of 16) group address and a four-bit channel-group identifier. The group address and channel-group identifier addressing elements are combined into a single programmable ID byte.

Each PW1-928 also contains a unique 32-bit serial number which is factory programmed into the PW1-928's internal EEPROM area. The PW1-928 serial number can be accessed by the host microcontroller via its RS-232 serial interface. This access to the 32-bit serial number allows the PW1-928 factory serial number to be used by the host application as a unique identifier.

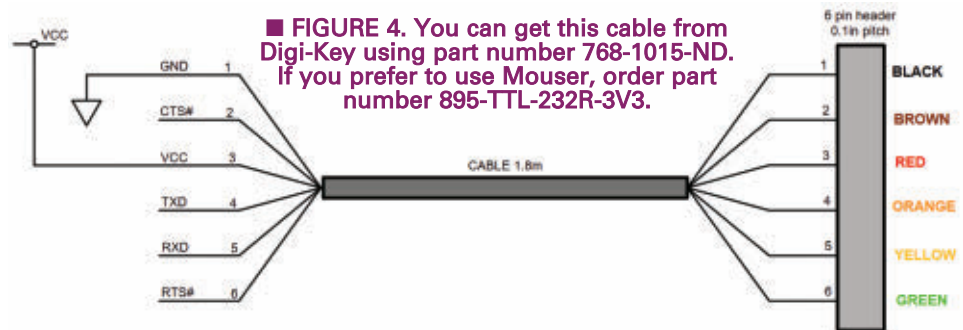
## The PW1-928 RF Data Transceiver Module Dev Kit

The PW1-928 module attaches easily to any microcontroller that is equipped with an embedded UART. However, if scratch building a PW1-928 host is not in the design cycle, bringing up a PW1-928 with its development kit is the easiest alternative option. The official Lemos International PW1-928 dev kit is shown in **Photo 3**.

The PW1-928's component details are graphically illustrated in **Schematic 1**. Basically, the PW1-928 dev kit main board provides a place to mount a PW1-928 transceiver module plus its antenna, a regulated 3.3 volt power supply, jumper access to all of the PW1-928's indication signals, and a 3.3 volt compatible serial



■ **FIGURE 3.** This layout complements the PW1-928 dev kit schematic.



■ **FIGURE 4.** You can get this cable from Digi-Key using part number 768-1015-ND. If you prefer to use Mouser, order part number 895-TTL-232R-3V3.

interface. Power for the PW1-928 dev kit can be sourced by the kit's onboard nine volt battery receptacle or via header pins on the dev kit main board. Note that the dev kit's UART interface is composed of discrete transistors. There are also LEDs attached to the PW1-928 indication I/O pins.

## Turning the PW1-928 Knobs

**Figure 3** augments the information provided by **Schematic 1**. If we wanted to connect our PW1-928 dev kit directly to a host PC's native RS-232 port (good luck finding one), we would use the PW1-928 RS-232 signals and ground associated with the dev kit's JP1 interface. In this case, we are going to interface our PW1-928 to a terminal emulator running on a PC. Since my brand new Dell does not present nine discrete serial port pins in the



## Command strings

RST	hard reset
DFLT	return to addr/chan=0, mode=0, baud=9600
EDUMP	pull the whole EEPROM (80 bytes) in human readable form
EDUMP 0	(as above but only locations 00-0F)
EDUMP 1	(as above but only locations 10-1F)
EDUMP 2	(as above but only locations 20-3F)
EDUMP 3	(as above but only locations 30-3F)
EDUMP 4	(as above but only locations 40-4F)
EDUMP #	(read only serial number, at locations 01,02,03,04)
EDUMP @	(read only channel/address ID, at location 08)
TEST CW	Goes into unmodulated CW on 915MHz
BAUD F	57600 kbps interface baud rate
BAUD M	38400 kbps interface baud rate
BAUD S	9600 kbps interface baud rate
MODE mm	Set mode byte

Mode byte	bit 0	tx inhibit
	1	rx inhibit
	2	diagnostic prefix (bytes 0,1,2)
	3	prefix RSSI byte
	4-6	(reserved)
	7	Responds to any valid packet (ignoring address and channel)

If any of bits 2-6 are set then burst will start with an = character (ASCII 0x3D)

ID SET	ca	program address and channel (stored in EEPROM)
ID TMP	ca	volatile address/channel (used, but not written to EEPROM)
USR WR	a dd	write to user EEPROM space (a is a 4-bit address, 0 through F)
USR RD	a	read from user EEPROM space (a is a 4-bit address)

■ **FIGURE 5.** These are the PW1-928 commands. Be sure to enter the EDUMP numeric in ASCII format. For instance, the EDUMP0 command equates to 21214544554D50300D.

old-fashioned way, we will have to interface our PW1-928 via one of the Dell's many USB ports.

To do this, we will deploy an FTDI TTL to RS-232 converter cable between the PW1-928 and the computer. The FTDI cable electrical pinout scheme is displayed in **Figure 4**. This particular cable is terminated on one end with a USB connector. The opposite end is exposed as a six-pin 0.1 inch pitch female connector. You can obtain this cable from Digi-Key or Mouser.

The PW1-928's serial\_in signal is connected to the FTDI cable's TXD pin. The FTDI cable's RXD signal is connected to the PW1-928's serial\_out pin. We must also place a jumper on the dev kit's JP5, which will take the PW1-928 out of standby mode. The PW1-928 dev kit will be powered by a standard nine volt battery.

According to the PW1-928 datasheet, if we send the PW1-928 a pair of ASCII exclamation points (!!) at 2400 bps, it will return an ASCII greater-than symbol (>).

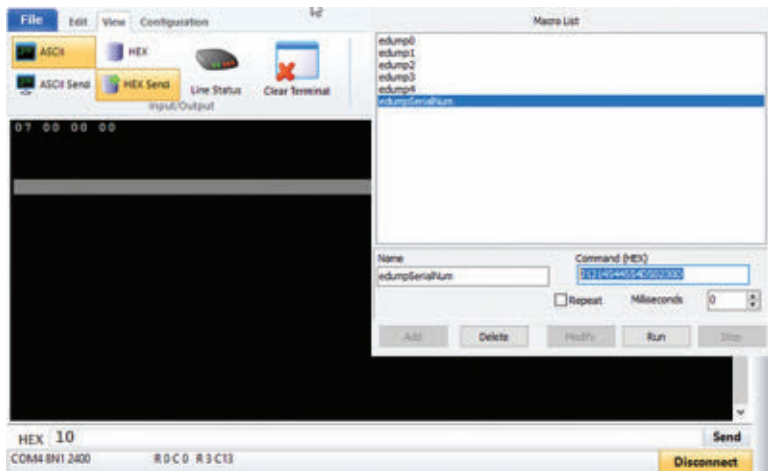
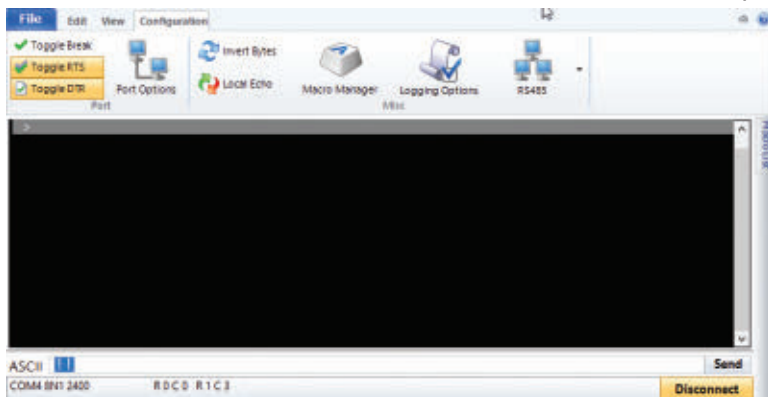
All of the PW1-928 programming command strings must be prefixed with two exclamation mark characters (ASCII 0x21) and are terminated with a carriage return (0x0D). All command string alpha characters must be in upper case, and command string numeric data must be in hexadecimal format. Spaces are ignored and can be added for enhanced human readability. The ESC character terminates and cancels any command entry.

The complete list of PW1-928 commands can be found in **Figure 5**. The terminal emulator I am using is called the Serial Input/Output Monitor and comes as part of the CCS PIC C compiler package. As you can see in **Screenshot 1**, I've used the terminal emulator's macro feature to code up some of the PW1-928 command structures. What you see in **Screenshot 2** are the results of the execution of the `edumpSerialNum` macro.

## Pushing the PW1-928's Buttons

The dev kit comes with a pair of PW1-928 carrier boards. We can easily put the PW1-928 radio modules to work without writing a single line of code or issuing a single command. The PW1-928 can be forced into test mode by simply placing

■ **SCREENSHOT 1.** Yep. Two ASCII exclamation points (!!) will spawn a greater-than symbol (>) answer from the PW1-928 RF data transceiver module.



■ **SCREENSHOT 2.** I used the macro capability of the CCS C Compiler's serial I/O app that comes packaged with their compiler to issue the read serial number command.

Conversely, when the RXS jumper pins are clear and the STBY jumper pins are grounded, the PW1-928 node assumes the receive test mode. The transmitting node sends an ASCII message every 330 ms. I set up one of the PW1-928s as a transmitter and the other as the receiver. The results can be seen in **Screenshot 3**.

Now that you have an idea of what the PW1-928 RF data transceiver module can do, get a dev kit and put it through its paces.

your Design Cycle. **NV**



**PANA VISE®**  
Innovative Holding Solutions

**RF Design Services**  
*Prepared to work with your in-house engineers,  
 or support your RF project from initial design to implementation.*  
 Industrial • Military • Space • Medical • Smart Grid Metering • SCADA • Lighting Control



For complete product details, visit us online!!

# The Nuts & Volts **WEBSTORE**

## GREAT FOR DIYers!

### The Big Book of Maker Space Projects

by Colleen Graves, Aaron Graves

**Start-to-finish  
fun projects for  
makers of all  
types, ages, and  
skill levels!**

This easy-to-follow guide features dozens of DIY low cost projects that will arm you with the skills necessary to dream up and build your own creations. Each project features non-technical step-by-step instructions with photos and illustrations to ensure success and expand your imagination. Learn recyclables hacks, smartphone tweaks, paper circuits, musical instruments, 3-D printing, and much more!

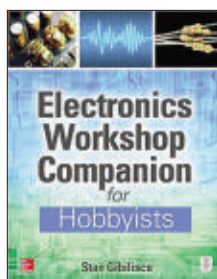


**\$20.00**

### Electronics Workshop Companion for Hobbyists

by Stan Gibilisco

In this practical guide, electronics expert Stan Gibilisco shows you, step by step, how to set up a home workshop so you can invent, design, build, test, and repair electronic circuits and gadgets. *Electronics Workshop Companion for Hobbyists* provides tips for constructing your workbench and stocking it with the tools, components, and test equipment you'll need. Clear illustrations and interesting do-it-yourself experiments are included throughout this hands-on resource.



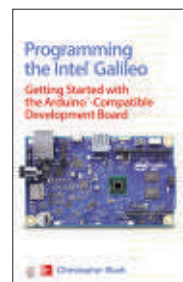
**\$25.00**

### Programming the Intel Galileo: Getting Started with the Arduino -Compatible Development Board

by

Christopher Rush

This hands-on guide offers a step-by-step intro to programming the Intel® Galileo using Arduino™ software. Written by an experienced electronics hobbyist, this book shows how to set up your board, configure the software, and quickly start writing sketches. You will discover how to work with the Galileo's inputs and outputs, use libraries, interface with the Web, and control external hardware. From there, you will learn to engineer and program your own useful and fun Galileo gadgets.

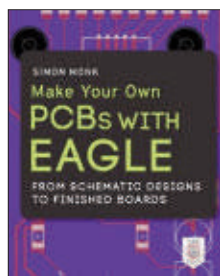


**\$18.00**

### Make Your Own PCBs with EAGLE

by Eric Kleinert

Featuring detailed illustrations and step-by-step instructions, *Make Your Own PCBs with EAGLE* leads you through the process of designing a schematic and transforming it into a PCB layout. You'll then move on to fabrication via the generation of standard Gerber files for submission to a PCB manufacturing service. This practical guide offers an accessible, logical way to learn EAGLE and start producing PCBs as quickly as possible.



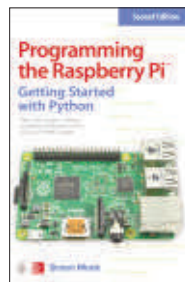
**\$30.00**

### Programming the Raspberry Pi, Second Edition: Getting Started with Python

by Simon Monk

**An updated guide  
to programming  
your own RaspPi  
projects**

Learn to create inventive programs and fun games on your powerful Raspberry Pi with no programming experience required. This practical book has been revised to fully cover the new Raspberry Pi 2, including upgrades to the Raspbian operating system. DIY projects include a hangman game, RGB LED controller, digital clock, and RasPiRobot with an ultrasonic rangefinder.



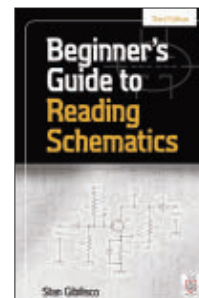
**Reg \$15.00**

**Sale Price \$ 9.95**

### Beginner's Guide to Reading Schematics, 3E

by Stan Gibilisco

Navigate the roadmaps of simple electronic circuits and complex systems with help from an experienced engineer. With all-new art and demo circuits you can build, this hands-on, illustrated guide explains how to understand and create high-precision electronics diagrams. Find out how to identify parts and connections, decipher element ratings, and apply diagram-based information in your own projects.



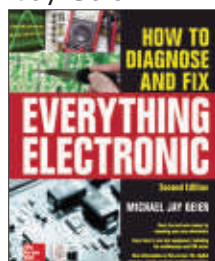
**\$25.00**

### How to Diagnose and Fix Everything Electronic

by Michael Jay Geier

**A Fully Revised  
Guide to  
Electronics**

Repair all kinds of electrical products, from modern digital gadgets to analog antiques, with help from this updated book. The Second Edition offers expert insights, case studies, and step-by-step instruction from a lifelong electronics guru. Discover how to assemble your workbench, use the latest test equipment, zero in on and replace dead components, and handle reassembly.



**\$24.95**

### Programming PICs in Basic

by Chuck Hellebuyck

If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller.



**\$14.95**

### Programming Arduino Next Steps: Going Further with Sketches

by Simon Monk

In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Also shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download.



**\$20.00**



Order online @ [store.nutsvolts.com](http://store.nutsvolts.com)  
Or CALL 1-800-783-4624 today!

## EDUCATIONAL

**Beginner's Guide Educational Combo!**



Price Reduced  
Now Only \$219.95

**ARDUINO**  
A Beginner's Guide To Programming Combo



A low cost introductory guide with experiments in programming electronics using the Arduino.

Only \$59.95

We now have the fully updated Learning Labs 1, 2, & 3 in stock!

Now \$45.95

Now \$47.50

Now \$41.95

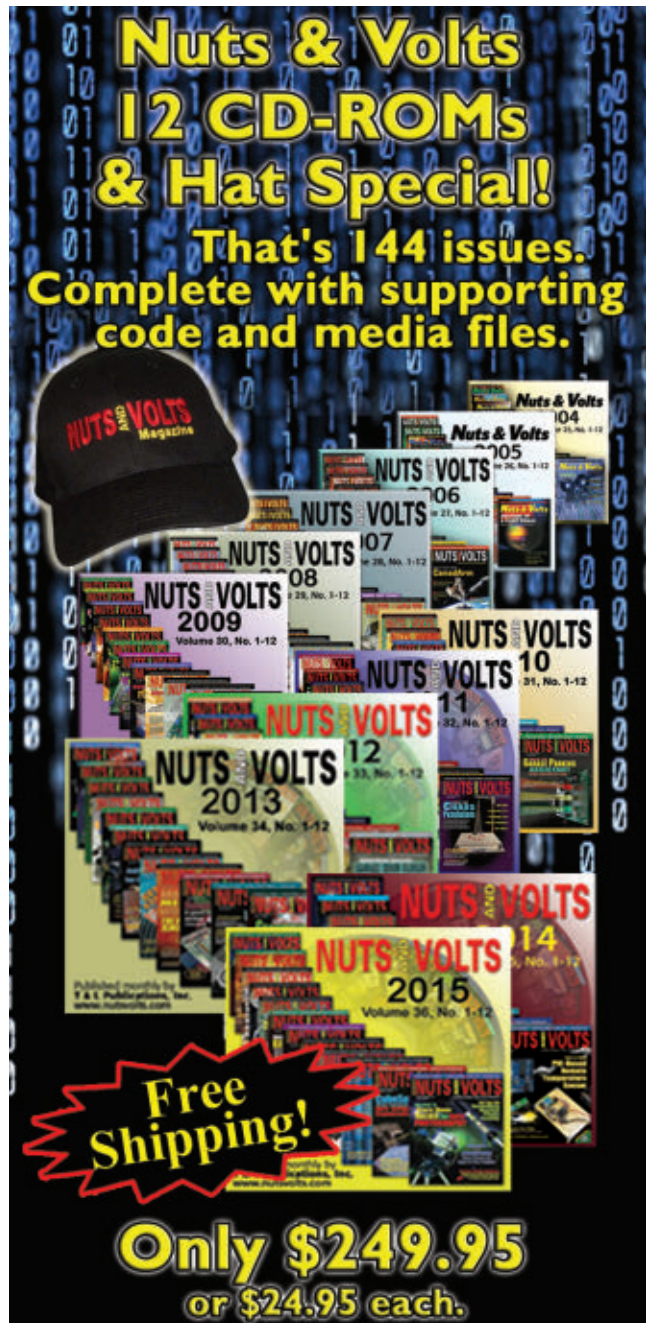
Are you (or someone in your family) looking to get started in engineering, electronics, computers, or robotics? If so, this hands-on comprehensive Learning Lab series is for you! Each Lab includes hands-on exercises and training on practical electronics, components, and circuits. Complete details on all the Learning Labs are available in our webstore. Simply type "lab" into the search field at <http://store.nutsvolts.com!>

Educator Discounts Available!

## CD-ROM SPECIAL

**Nuts & Volts**  
**12 CD-ROMs**  
**& Hat Special!**

That's 144 issues.  
Complete with supporting code and media files.



Free Shipping!

Only \$249.95  
or \$24.95 each.

**POCKET REF**  
The Nuts & Volts Pocket Ref

All the info you need at your fingertips!

This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants & conversions.

Only \$12.95

AND it still fits in your shirt pocket!

Visit [store.nutsvolts.com](http://store.nutsvolts.com) or call (800) 783-4624



## >>> QUESTIONS

### Ni-Metal Hydride Battery Chargers

Now that Ni-Cad batteries are becoming an endangered species, is there any way to convert their chargers to work on the newer and ecologically sounder Ni-Metal Hydride batteries? I read that the Ni-Metal batteries have very different charging requirements rather than the constant current chargers for Ni-Cads. I have several bench chargers, not to mention a multitude of "wall warts" that I would like to modify.

**#4171** **Michael Herman**  
La Quinta, CA

### POTS Ring Generator

I need a circuit to generate 20 Hz 90 VAC to ring a POTS telephone on a theater stage. That's Plain Old Telephone System vs. Pretty Advanced New Stuff (PANS) for the younger readers. If I could locate an old crank ring generator, it would work but would not be as convenient.

**#4172** **Dale Carlsen**  
Woodbridge, VA

### Warm Cows Are Happy Cows

My daughter is in 4H and has a calf we are raising in a shed behind the house. Recently, the bulb burned out on the heat lamp we leave on to keep the cow warm. Is there a simple circuit I could use to remotely monitor the temperature and alert me if it drops below a set point? I know there are remote temperature gauges but I need one with an alert or a way to hack one to add an alert. Any ideas welcome.

**#4173** **Robert Lowery**  
Florence, SD

## >>> ANSWERS

### [#11165 - November 2016] Tube Tech

*Is there any \*technical\* difference between tube amp distortion and solid-state amp distortion? I have heard tube amps described as "warm" sounding but I can't find any info as to why. Isn't "clipping" just "clipping" no matter the device that is performing that function?*

**#1** It's not about clipping. Tubes and FETs have greater inherent 2nd harmonic distortion which gives them the warm sound. Look up the books and articles by Douglas Self to truly understand why modern bipolar output amplifiers are very hard to beat for sonic clarity. Some of the new Class D amplifiers (TI, others) are quite amazingly clear, too. I still use FET input op-amps (LF412) when I want some of that tube warmth, but when I need absolute clarity, modern bipolar devices (LM833 and newer) for crossovers and bipolar output stages are (in my opinion) best. Read what Self has to say.

**Jim Lacenski**  
Bellevue, WA

**#2** In this case, all clipping isn't the same. A transistor circuit is fine up to the power supply voltage, where it mows the peaks off square and flat. This produces a harsh distortion similar to the fuzz pedal for a guitar. A tube circuit starts to round off the peaks before they actually run into the power supply voltage. The rounder peaks account for the "warmer" sound.

**Chip Veres**  
Miami, FL

**#3** Yes, there are technical differences. Both tubes and transistors

are nonlinear devices, and the transfer curve for each is unique. The transfer curve defines how the output should respond to the input. Within a narrowly defined range of input values, the output values change in "mostly" linear fashion — in math terms, we would say the function is monotonic. When your input values start to go beyond the linear operating region, the output is no longer a simple (linear or monotonic) function of the input.

Every device, triode, pentode, JFET, MOSFET, BJT has a unique transfer curve. Imagine that the transfer curve for a triode is not a straight line, but more of a "lazy-S" shape — the middle section is pretty close to straight, but the top and bottom of the curve rolls over. As mentioned above, this curve "maps" your input signal to the output signal; any given value of input is a point on the curve that defines the output. When the signal is near the very top of the curve, however, the output signal change for a given change of input is diminished (like a demon turning down your volume knob). This results in a soft clipping effect if the top of the transfer curve is relatively smooth. Gentle excursions into nonlinear behavior in a triode tend to produce a nice mix of even and odd harmonics; and, if I recall correctly, even harmonics lend warmth to the sound.

In the case of a BJT (bipolar junction transistor), the lazy-S curve looks more like a "Z" drawn backwards, where the extremes of the transfer curve don't bend gently. Instead, they have sharp "corners" and tend toward a "flatter" transfer function at the extremes. When the input signal gets into this nonlinear region, a large change of input signal results in almost no change of output signal, but can produce lots

See more at  
[www.nutsvolts.com/tech-forum](http://www.nutsvolts.com/tech-forum)

Send all questions and answers by email to [forum@nutsvolts.com](mailto:forum@nutsvolts.com)  
or via the online form at [www.nutsvolts.com/tech-forum](http://www.nutsvolts.com/tech-forum)

of harmonics — predominantly odd-harmonics.

The *clipping* is more aggressive at the extremes of signal input; almost an “all or nothing” affair. Contrast that to the tube clipping, which is more like “diminishing returns.” I have heard of an amplifier circuit that adds even-harmonic components of the signal. I haven’t built it, but the idea is that it would create a warmer sound.

Having said all that, clipping and harmonic content (warm versus cool) should not be thought of as synonymous. Clipping occurs at the extreme limits of signal input. Nonlinear transfer curves can create harmonics at any value of input signal. When you look in the mirror in the morning, you are seeing a “linear” reflection of yourself. At the carnival

or fairground when you stand in front of curved mirrors that distort your reflection, you are seeing an exaggerated “nonlinear” reflection of yourself. While seeing such exaggerated nonlinearity is humorous, in the audio realm it would be intolerable to listen to ... maybe.

Guitar effects pedals intentionally distort the signal, sometimes to an extreme that is almost unrecognizable — but, that’s an article for another day.

**Doug Manchester**  
Rocklin, CA

**[#12162 - December 2016]  
Speaker Sharing**

*Does anyone have a simple circuit that will allow me to mix my iPod audio out with my computer’s audio*

*to play through the same speakers? I want to avoid un-plugging/re-plugging just to hear some music.*

I use a simple DPDT toggle switch for something like this. I switch my computer output between the speaker and headphones without wearing out the headphone jack.

In your case, the audio to the speaker would be switched between the iPod or the computer. The speakers must be amplified, and the computer’s output volume adjusted down to be close to the level coming out of the iPod headphone jack. You can mount the jack and switch on the speaker.

**Bert**  
Toronto

**pico**<sup>®</sup>  
Technology

## THE PICOSCOPE 4444

### SEE THE DIFFERENCE



#### A NEW STANDARD IN DIFFERENTIAL MEASUREMENT

- 20 MHz bandwidth, 12 to 14 bit resolution
- 4 differential inputs
- 1000 V CAT III probes
- Low-voltage probes and current clamps

**NEW**

For more information please visit [www.picotech.com/US102](http://www.picotech.com/US102)

Prices are correct at the time of publication. Sales taxes not included. Please contact Pico Technology for the latest prices before ordering. Email: [sales@picotech.com](mailto:sales@picotech.com). Errors and omissions excepted.



**USB** Add USB to your next project--  
It's easier than you might think!   
USB-FIFO • USB-UART • USB/Microcontroller Boards  
RFID Readers • Design/Manufacturing Services Available  
*Absolutely NO driver software development required!*  
[www.dlpdesign.com](http://www.dlpdesign.com)

**PCB Fab & Assembly Services**

- Start from 1pc
- Worldwide shipping
- Save 15% your first PCB order



**NEW! Omni Wheels**

- Actobotics® Compatible!
- Rolls Forward & Side to Side!
- Aggressively Priced!

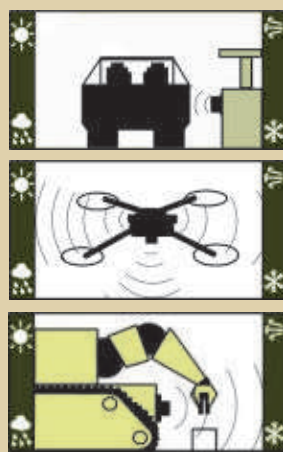
available @ **SERVOCITY**



## CLASSIFIEDS

### ROBOTICS

#### Need sensors?



[www.maxbotix.com](http://www.maxbotix.com)

### HARDWARE WANTED

#### DEC and VME Equipment WANTED!

Digital Equipment Corp  
and  
Motorola VME Systems  
and Parts  
Buy - Sell - Trade

CALL KEYWAYS **937-847-2300**  
or email [buyer@keyways.com](mailto:buyer@keyways.com)

### LIGHTING



[www.ReactiveLighting.com](http://www.ReactiveLighting.com)

# ADvertiser INDEX

■ LOOK FOR ■ SEARCH FOR ■ FIND

Find your favorite advertisers here!

### BATTERIES/CHARGERS

Hitec .....2

### BUYING ELECTRONIC SURPLUS

All Electronics Corp. ....50

### CIRCUIT BOARDS

Accutrace .....67

ExpressPCB .....22

PCB Fab Express .....39

Saelig Co., Inc. ....9

### COMPONENTS

All Electronics Corp. ....50

Capacitors Indirect .....3

Saelig Co., Inc. ....9

SDP/SI .....15

### DATA LOGGING

Measurement Computing .....13

### DESIGN/ENGINEERING /REPAIR SERVICES

Accutrace .....67

ExpressPCB .....22

### DEVELOPMENT PLATFORMS/TOOLS

Anaren .....Back Cover

Technologic Systems .....34

### EDUCATION

Command Productions .....27

PoLabs .....13

### EMBEDDED SYSTEMS

Measurement Computing .....13

Saelig Co. Inc. ....9

Technologic Systems .....34

### ENCLOSURES

Hammond Manufacturing .....45

### EVENTS

Maker Faire .....65

### LCDs/DISPLAYS

EarthLCD .....15

Saelig Co., Inc. ....9

### MICROCONTROLLERS / I/O BOARDS

Technologic Systems .....34

### MISC./SURPLUS

All Electronics Corp. ....50

### MOTORS / MOTOR CONTROL

Hitec .....2

ServoCity .....66

### ROBOTICS

Hitec .....2

SDP/SI .....15

ServoCity .....66

### TEST EQUIPMENT

Pico Technology .....63

PoLabs .....13

Saelig Co., Inc. ....9

### TOOLS

PanaVise .....59

PoLabs .....13

### TRANSFORMERS

Hammond Manufacturing .....45

### WIRE, CABLE AND CONNECTORS

All Electronics Corp. ....50

### WIRELESS PRODUCTS

Anaren .....Back Cover

Lemos International .....59

Technologic Systems .....34

Accutrace .....67

All Electronics Corp. ....50

Anaren .....Back Cover

Capacitors Indirect .....3

Command Productions .....27

EarthLCD .....15

ExpressPCB .....22

Hammond Manufacturing .....45

Hitec .....2

Lemos International .....59

Maker Faire .....65

Measurement Computing .....13

PanaVise .....59

PCB Fab Express .....39

Pico Technology .....63

PoLabs .....13

Saelig Co., Inc. ....9

SDP/SI .....15

ServoCity .....66

Technologic Systems .....34

PRESENTED BY 

# Maker Faire<sup>®</sup>

12TH ANNUAL

BAY AREA

MAY 19-21

SAN MATEO EVENT CENTER

Greatest  
SHOW & TELL  
on Earth

PRESENTED BY



GOLDSMITH SPONSORS

Google



Microsoft

SILVERSMITH SPONSORS

BOSEbuild



Cognizant



GET YOUR  
**TICKETS**  
TODAY!

 [makerfaire.com](http://makerfaire.com)

BROUGHT  
TO YOU BY **Make:**

You can miter the edges

545400

Using a 585500 to attach a bottom tapped clamping mount

- Weee

585506

545368 connected to 5/8" Tube

585494 (Beefy)

**SERVOCITY**.COM  
WE HAVE THE PARTS FOR YOUR IDEAS

Dual Ball Bearing Hub (One of our faves!) 545444

Pattern Plate to Clamping Mount to HD Motor

585588

Channel Hard Mounted Directly to X-Rail

## ACTOBOTICS X-Rail

- Great for Creating Strong Lightweight Structures
- Awesome as a Linear Motion Guide
- Actobotics Pattern on the End

## STANDARD & MINI V-Wheels

- Built-in Bearings for Super Smooth Motion
- Use with a variety of standoffs &/or spacers
- Standard and Mini Versions

Channel to X-Rail Roller Brackets to Standoffs to V-Wheels

1" OD Tube to 1" ID Clamping Hub 545352

Visit [ServoCity.com](http://ServoCity.com) for the complete lineup of Actobotics parts!

**ACTOBOTICS**  
DREAM • DESIGN • BUILD • REPEAT

All of these ideas are just the tip of the iceberg! What will you make?

Actobotics X-Rail Fits inside a 1" ID Clamp such as 545464

585628

Mini V-Wheels for Compact Builds

Standard V-Wheels for More Strength



PRINTED CIRCUIT BOARDS

# NO GIMMICKS!

## CHECK OUT OUR EVERYDAY PRICE

NO COMPROMISE ON QUALITY AND ON-TIME DELIVERY!



**THERE ARE NO GAMES INVOLVED IN OUR PRICING**



### Our Full Service Capabilities:

- From same day quick turn prototype to production in under 10 days
- Full CAD and CAM review plus design rule check on ALL Gerber files
- Materials: Fr4, Rigid, Flex, Metal Core (Aluminum), Polyimide, Rogers, Isola, etc.
- HDI Capabilities: Blind/Buried Microvias, 10+N+10, Via-in-Pad Technology, Sequential Lamination, Any Layer, etc.
- Our HDI Advantage: Direct Laser Drilling, Plasma De-Smear Technology, Laser Microvia, Conductive Plate Shut.

**Accutrace Inc.**

[www.PCB4u.com](http://www.PCB4u.com) [sales@PCB4u.com](mailto:sales@PCB4u.com) (408)748-9600

● ITAR ● I SO 9001:2008 ● UL Approved



# Take Off and Soar

With the Anaren Atmosphere 2.0 IoT Development environment and Atmosphere Cloud™ hosting, – *plus* – the new A43364 Wi-Fi development board, pilot your IoT project from concept to cloud connectivity at Mach speed. No delays. No runaway costs. First class all the way.



[www.anaren.com/air/win](http://www.anaren.com/air/win)

\* Must be 18 to participate. No purchase necessary. North America only. Void where prohibited. Other restrictions apply.



Scan the code to learn more and register for a chance to win a free Atmosphere MSDK!\*

Only the Atmosphere 2.0 IoT Development environment with Atmosphere Cloud™ hosting gives you the fastest trajectory for developing connected, embedded Wi-Fi applications, complete with mobile apps and cloud support. Think. Build. Connect.



**Anaren®**

What'll we think of next?®