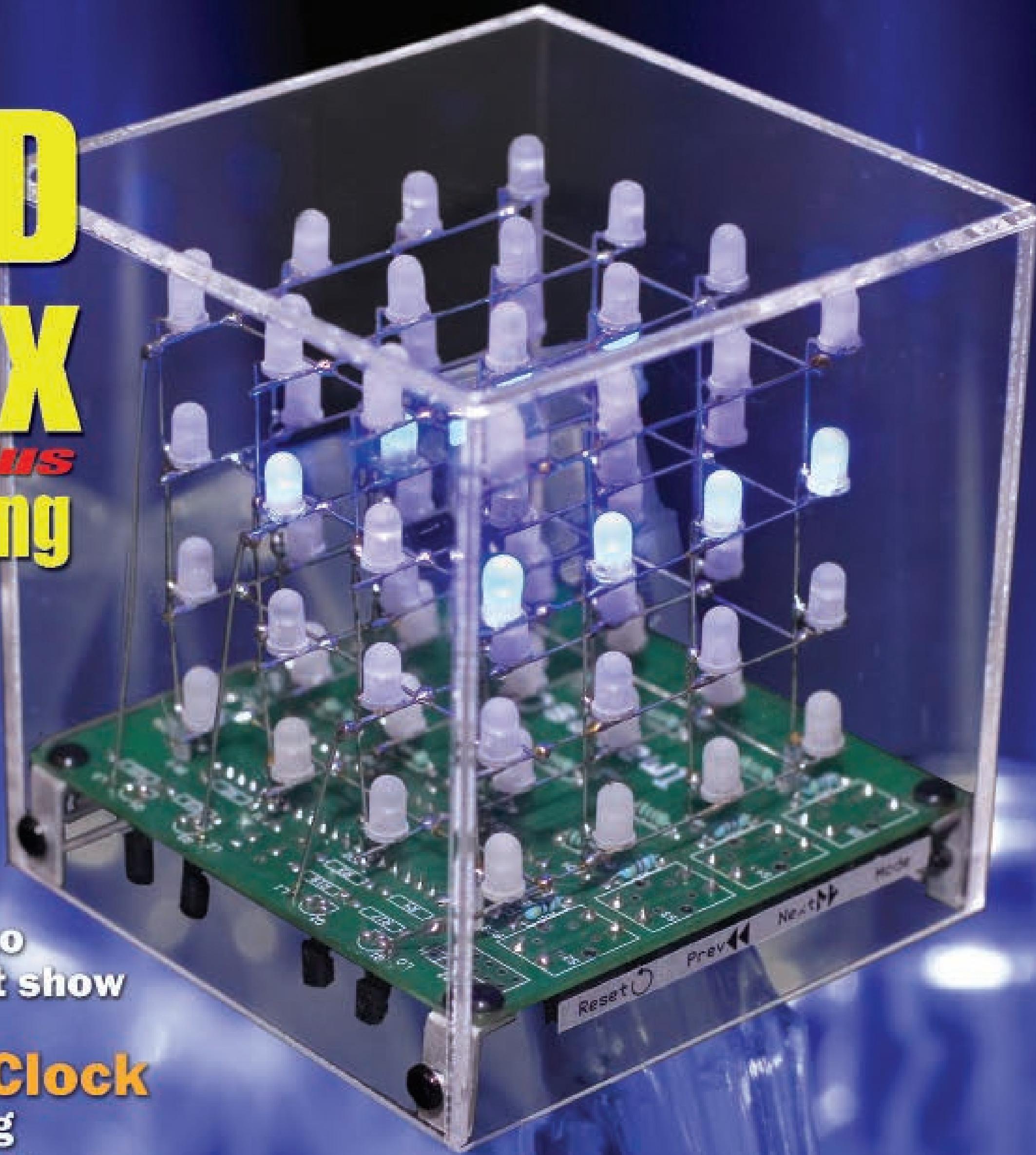
■ PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

NUTS AND VOLTS

EVERYTHING FOR ELECTRONICS

www.nutsvolts.com
August 2011



Build the 3D LED Matrix

Plus 3 Other Exciting LED Projects

✦ **Miniature Color Organ**
Attach to speakers, drums, or anything, to create your own light show

✦ **LED Analog Clock**
A great mix of analog and digital electronics

✦ **Infinity Portal**
Try this Arduino controlled upgrade to the Infinity Mirror concept

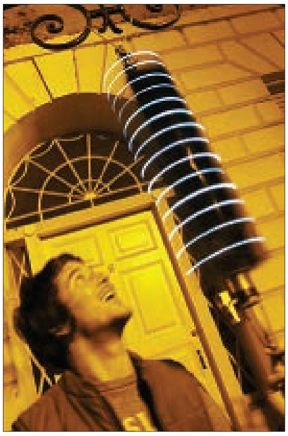
U.S. $6.50 CANADA $7.50

08

0 71486 02421 7

Nuts & Volts
August 2011
www.nutsvolts.com

All LED Projects this month!!



Projects & Features

Columns

FUNdamentals For Beginners

Just getting started in electronics? Try these simple demos to illustrate electronics principles and how components function in actual circuits!

Departments

Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

I2I CONTROLS
Tomorrow's programming revolution,
starts today!
The choice is yours to make
//--
*
SSD1963_8-bit
Program for writing to Ne-
whaven Display 5.7" TFT
*/
//--
Delay_ms(100);
Write_Command(0x01); //Software Reset
Write_Command(0x01);
Write_Command(0x01);
Delay_ms(10);
Command_Write(0xE0,0x01); //START PLL
Command_Write(0xE0,0x03); //LOCK PLL
Write_Command(0xB0); //SET LCD MODE SET TFT 18Bits
MODE
Write_Data(0x0C); //SET TFT MODE & hsync+Vsync+DEN
MODE
Write_Data(0x80); //SET TFT MODE & hsync+Vsync+DEN
MODE
Write_Data(0x01); //SET horizontal size=320-1 Hight-
Byte
Write_Data(0x3F); //SET horizontal size=320-1 LowByte
Write_Data(0x00); //SET vertical size=240-1 HightByte
Write_Data(0xEF); //SET vertical size=240-1 LowByte
Write_Data(0x00); //SET even/odd line RGB seq.=RGB
Command_Write(0xF0,0x00); //SET pixel data I/F
format=8bit
Command_Write(0x3A,0x60); // SET R G B format = 6 6 6
Write_Command(0xE6); //SET PCLK freq=4.94MHz ; pixel
clock frequency
Write_Data(0x02);
Write_Data(0xFF);
Write_Data(0xFF);
Write_Command(0xB4); //SET HBP,
Write_Data(0x01); //SET HSYNC Total 440
Write_Data(0xB8);
Write_Data(0x00); //SET HBP 68
Write_Data(0x44);
Write_Data(0x0F); //SET VBP 16=15+1
Write_Data(0x00); //SET Hsync pulse start position
Write_Data(0x00);
Write_Data(0x00); //SET Hsync pulse subpixel start posi-
tion
Write_Command(0xB6); //SET VBP,
more
; Using Newhaven display
with SSD1963 controller
; Do date and time
setio g, 0xF000 ;Set location for display
setio u, 0xFA00 ;Set location for USB drive
load "font.hex" ;Get font table from USB
setglcd i ;Initialize the graphics LCD
setglcd c ;Clear the display
setglcd D1 ;Select large font
do
setglcd p,(100,200);Set cursor postion
; Display time
glcdout using(2),!h,":",!m,":",!s," ",
; Display date
glcdout using 2),!mo,"/",!d,"/",!y
until s=1
; All done
Choose your path at
www.i2icontrols.com

by Bryan Bergeron, Editor

DEVELOPING PERSPECTIVES

Virtual Portable Instruments

Virtual Instruments — instruments that leverage the digital signal processing and screen display of a desktop or laptop computer — have been around for decades. Virtual instruments can be an affordable way to create a reasonable assortment of traditional desktop instruments — from oscilloscopes and DMMs to spectrum analyzers — on a budget.

Virtual instruments aren't perfect, however. They usually tie you to a desk or desktop, and the interface boxes and cables running to a laptop can be difficult to keep straight. In addition, in most cases you can't run multiple instruments simultaneously. For example, you can run either a virtual oscilloscope or virtual DMM, but not both at the same time.

Enter the smart phone and tablet computers. Companies are now introducing virtual instruments that convert your phone or tablet into an oscilloscope or other virtual instrument. I've been working with the iMSO-104 — an oscilloscope for the iPhone/iTouch/iPad by Oscium (**www.oscium.com**). The $298 scope consists of a small interface that plugs in directly to the connector port on the i-device and software from the AppStore. You supply the iPad.

Specs for the 5 MHz/12 MSPS (million samples per second) scope fit squarely in the hobbyist market, with one analog and four digital channels. In other words, it's a nice scope if you're working with audio, digital, or analog servos and the like. However, if you're working on a high speed digital circuit or RF communications circuit, you're out of luck. Still, I expect specifications to improve and the price to drop with time and competition.

An alternative to virtual instruments is ultra-portable dedicated instruments, such as the DSO-NANO DSONV-11 pocket oscilloscope. I picked up this $89, single channel digital storage scope from Seeed Studio (**www.seeedstudio.com**) about a year ago. I have to admit that I've only used the cell phone-sized unit a handful of times, in part because I seldom have the 1 MHz scope with me, and in part because it takes me a few minutes to figure out the interface every time I turn it on. The current model apparently provides an updated interface that's easier to use. There's also a four-channel version available for $200.

The potential advantage of a library of virtual instruments that abide by a single user interface guideline – say Android or iPad – is that any instrument is easy to pick up and use without wasting time to remember or learn the interface. Given the relatively lax standards for Android apps, this standardization may appear first on the iPad. At this point, it's too early to tell.

While using my Oscium oscilloscope, I discovered another issue with virtual portable instruments – sometimes you need the supporting tablet or phone for something else. On a stand-alone instrument, you're not likely to be interrupted by a phone call or a signal that you've got mail. Moreover, you can easily answer your smart phone while working on a circuit. Not so if everything sits in your cell phone. I'm sure someone will think of a work-around.

Given the falling price of computer-based instruments in general, dedicated instruments may go the way of the DMM – that is, nearly disposable. DMMs are so inexpensive these days that you can

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

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

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER/ASSOCIATE PUBLISHER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Russ Kincaid, Joe Pardue, Fred Eady, Ron Newton, John Iovine, Craig Lindley, David Welburn, Ron Hackett, Lou Frenzel

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@nutsvolts.com

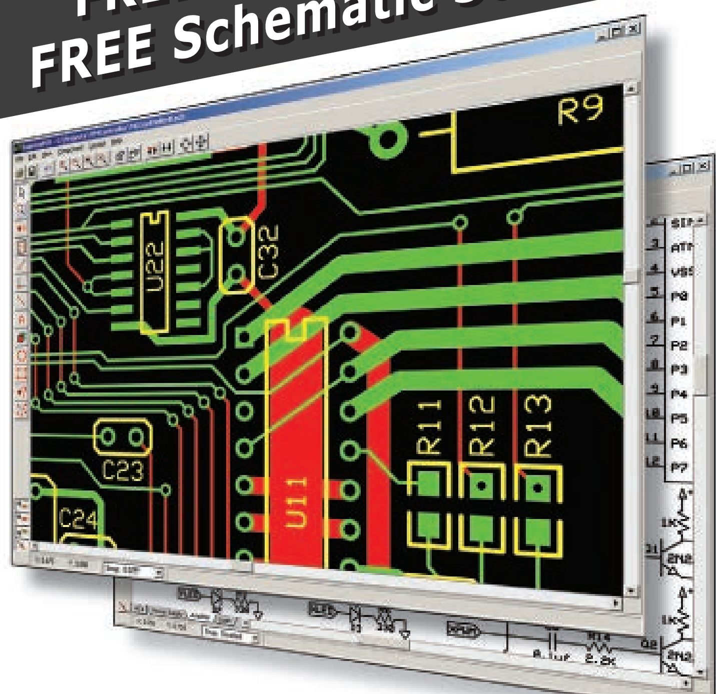
WEB CONTENT
Michael Kaudze
website@nutsvolts.com

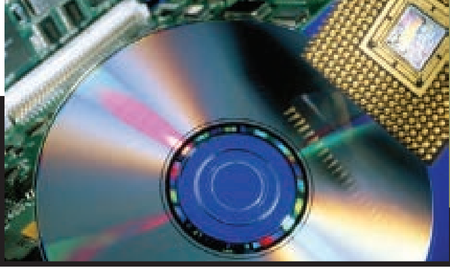
ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

dedicate one to, say, your power supply project. It's nearly the same with pocket oscilloscopes. At $89, it's almost economical to permanently affix a DSO-NANO scope to your audio amp or other device, versus building a separate audio level monitor circuit and display.

One thing is certain. The days of a hobbyist's workbench loaded with large, dedicated boxes of instruments are numbered. Economics, limited space, and technology are driving companies to offer instruments that leverage the processing power and displays of increasingly powerful phones and tablets. **NV**





TECH KNOWLEDGEY 2011

EVENTS, ADVANCES, AND NEWS

BY JEFF ECKERT

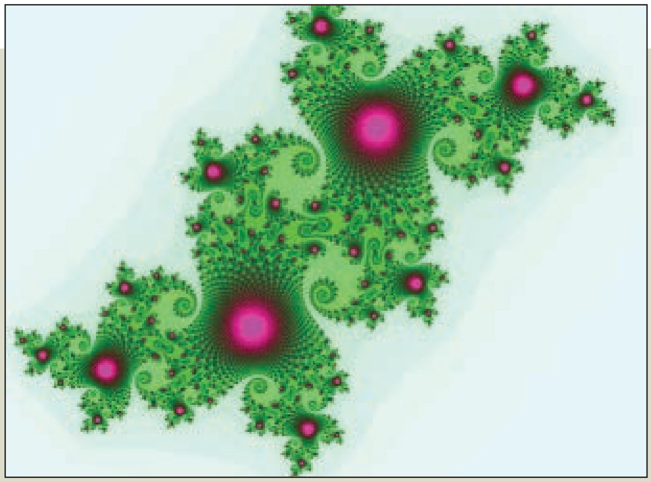
ADVANCED TECHNOLOGY

FRACTALS TO CURE BLINDNESS

You may already be asking, "what's a fractal," so let's get that out of the way. The term refers to a fragmented geometric shape that can be divided into smaller parts, each of which is more or less a smaller copy of the original. Not that it matters, but this property is called self-similarity. The fractals that have been of interest lately are mathematically generated shapes, but some examples exist in nature such as clouds, snowflakes, broccoli, and (most importantly for present purposes) neurons. Because they look pretty much the same at any magnification level, fractals are considered to be infinitely complex, at least in theory.

Now, if you have a degenerative eye disease such as retinitis pigmentosa or macular degeneration, blindness is probably ahead, but research is being conducted to develop retinal implants that can put a semiconductor on top of your retinal neurons and stimulate them such that you will be able to see again. One of the problems, however, is that the neurons that connect the human retina are fractals, but electronic chips are not. The latter are just little, square collections of electrodes. This becomes a problem when you try to connect the two, as the chips provide such minimal overlap with the retina's wiring that retinal implants so far can provide only about 50 pixels of resolution (as opposed to a healthy eye's 127 megapixels).

However, Richard Taylor — a physicist and director of the University of Oregon's Materials Science Institute (**materialscience.uoregon.edu**) — is working on a method of growing nano-sized "flowers" that self-assemble into fractal shapes. These metallic nanoflowers could be implanted in blind patients, "providing interface circuitry that would collect light captured by the retina and guide it with almost 100 percent efficiency to neurons for relay to the optic nerve to process vision."



A Julia fractal, named for French mathematician Gaston Maurice Julia.

Taylor has applied for a patent on his concept for fractal-based photodiodes, so he must be pretty serious about it. The main obstacle, he says, is figuring out which metals are best to pop into your eye while avoiding toxicity problems. Research in that area has already begun in New Zealand, so keep your fingers crossed. ▲

COMPUTERS AND NETWORKING

YOUR LAST RADIO STATION

There are thousands of Internet music services out there, and lots of them — such as Pandora (**www.pandora.com**) — offer the ability to create your own customized "stations" that at least attempt to play only stuff that you like. Generally, you just pick a few of your favorite artists, and the service plays tunes in what it considers to be the same category. A more ambitious version is Last.fm (**www.last.fm**) which also offers social networking based on your musical preferences, a listing of concerts and other musical events, and artist and track charts. Even more interesting is the site offers a piece of downloadable software (the Scrobbler) that keeps track of what songs you play, how often you play them, and so on. It also scans your hard drive to see what music you already own.

This information is compiled as a little note called a "scrobble" and sent back to Last.fm, which then compares your scrobble to those of millions of other listeners and bases its recommendations on their programming habits. Reportedly, it also shares that data with record labels, so if you're paranoid about that kind of thing, you may not want to go there.

Perhaps the most intriguing feature is that if you are an artist yourself, you can upload your songs and make them available to 40 million listeners, set up a fan network, and even get paid royalties when your tunes are streamed. Fame and fortune may be just a few clicks away! And maybe not. ▲

COMPUTERS AND NETWORKING *CONTINUED*

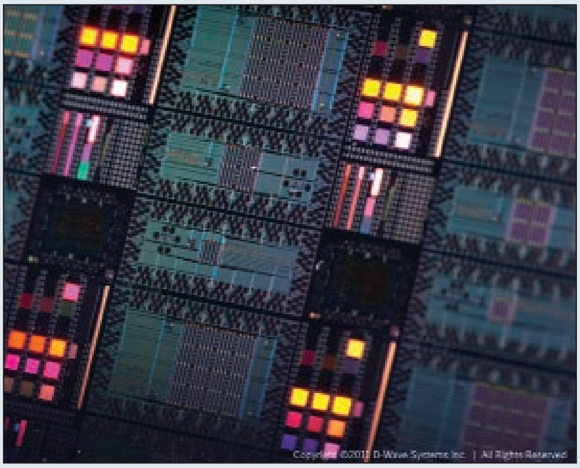
FIRST QUANTUM COMPUTER SOLD

For several years, Canada's D-Wave Systems (**www.dwavesys.com**) has claimed to have a working, commercially available quantum computer and, in fact, demonstrated one as early as 2007. The only problem is that almost no one believed it was for real. Critics complained that D-Wave had not published any major advances or breakthroughs in scientific literature, and the machine was actually located far away from the demo site and could not be inspected.

(D-Wave explained that, because of the machine's liquid helium cooling system and sensitive components, it could not be moved.) However, a recent paper in the journal *Nature* (**www.nature.com**) must have settled the matter, because Lockheed Martin has decided to buy one.

Relatively little information was made available, but we do know that the D-Wave One is designed for industrial problems "encountered by Fortune 500 companies, government, and academia."

The heart of the machine is a superconducting 128-qubit processor chip running in a cryogenic system that keeps it nearly at absolute zero. All of that is contained in a 10 square meter shielded room. This is considerably short of the 1024-qubit machine the company was predicting in 2007, but the R&D budget probably gets tight when you sell zero machines for several years.



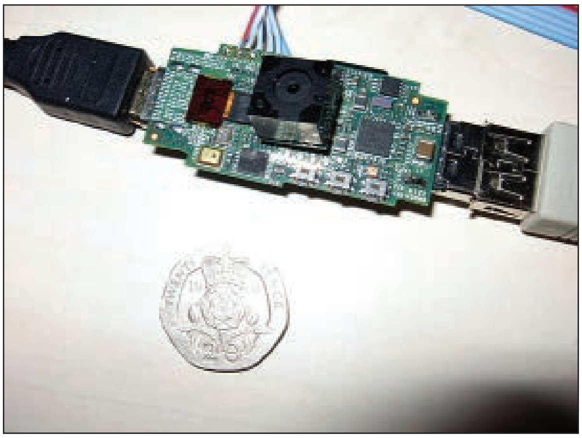
■ D-Wave processors on a wafer. (*Copyright © D-Wave Systems.*)

In any event, Lockheed Martin seems to have been sufficiently impressed to jump into the development pool. Even though the machine has exotic technology inside, the user interface is fairly straightforward. Supported programming environments include Python, Java C++, SQL, and MATLAB. The price was not revealed, but the Internet rumor mill pegs it at around $10 million. ▲

BOTTOM OF THE BARREL

Can't afford a D-Wave One? Not even a used Gateway? Well, how about a Raspberry Pi, which at $25 is probably the cheapest PC in the world. The machine is the brainchild of British game developer David Braben, who assembled a few friends and started the nonprofit Raspberry Pi Foundation (**www.raspberrypi.org**). The goal is to develop, manufacture, and distribute the ultra-low-cost computer for use in teaching programming to children.

Scheduled to hit the market later this year, its specs include a 700 MHz ARM11 processor, 256 MB of SDRAM, composite and HDMI video output, USB 2.0 and a general-purpose I/O, one memory card slot, and a collection of open software (Unbuntu, Python, etc.). This is not exactly a powerhouse, but it sounds pretty reasonable until you consider that it doesn't come with a keyboard, mouse, or monitor.

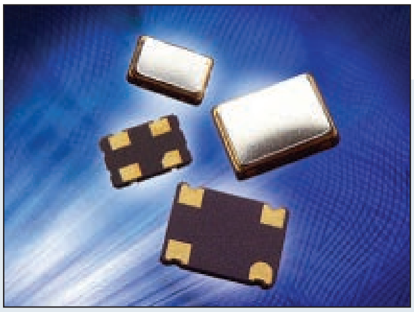


■ The Raspberry Pi computer with attached 12 Mpixel camera module.

This makes it compare pretty unfavorably with a touch-screen computer recently developed at the Indian Institute of Technology and Indian Institute of Science, slated to sell for $35 — if they can line up someone to move from prototype to commercial production. As unlikely as that may seem, remember that this is a country that came up with $2,000 open-heart surgery and the Tata Nano — a car that sells for $2,500 new. I guess the competition is even tough at the bottom these days. ▲

CIRCUITS AND DEVICES

OSCILLATORS OFFER EMI REDUCTION



■ Precision Devices' SSOC5 and SSOC7 crystal clock oscillators.

Major sources of electromagnetic interference (EMI) in electronic circuits include such frequency sources as phase lock loop (PLL) synthesizers, quartz oscillators, and other clock signal generators. This is often ameliorated by using filters, shielded enclosures, and complex grounding schemes. But employing a spread spectrum crystal clock oscillator — in which the output frequency is modulated — can reduce EMI at the source which can result in simplified regulatory testing and reduced system costs.

Two such devices — the SSOC5 and SSOC7 — were recently introduced by Precision Devices, Inc. (**www.pdixtal.com**), intended for applications including office equipment, computer systems, telecommunications, set top boxes, and embedded systems. The modulated output of these devices is said to reduce EMI by as much as 15 dBc as compared to fixed crystal oscillators. Both operate with a 3.3V supply voltage, and the CMOS clock generator produces a square wave fixed output ranging from 6.00 to 160.00 MHz (SSOC5) and 3.60 to 220.00 MHz (SSOC7).

Specs include a load termination of 15 pF maximum with rise and fall times of 4.0 ns (10 to 90 percent of the waveform), and the units offer a ±25, ±50, and ±100 PPM frequency stability option over an operating temperature range of 0 to +70°C (-40 to +85°C optionally available). Both come in a four-pin SMD package. ▲

THE OTHER LIGHTING ALTERNATIVE

There has been considerable controversy over tungsten vs. compact fluorescent (CFL) bulbs of late, but relatively little discussion about the other alternative: LED lighting. LEDs have become increasingly viable over the last few years and, in fact, can provide both energy efficiency and high quality light — particularly in recessed lighting installations where omnidirectional emission isn't so important.

■ Cree's LMR4 LED lighting module — a viable replacement for 100W incandescents.

An example is the new 1,000 lumen LMR4 module from Cree, Inc. The unit delivers 66 lumens per watt which they claim is 42 percent better than a 26 W CFL. The LMR4 is said to be the only 2,700K LED module that delivers more than 90 CRI, making it a good replacement for 100W incandescent bulbs in downlight applications. They are also available in color temperatures of 3,000, 3,500, and 4,000K. All are designed to last 35,000 hr and are dimmable to five percent.

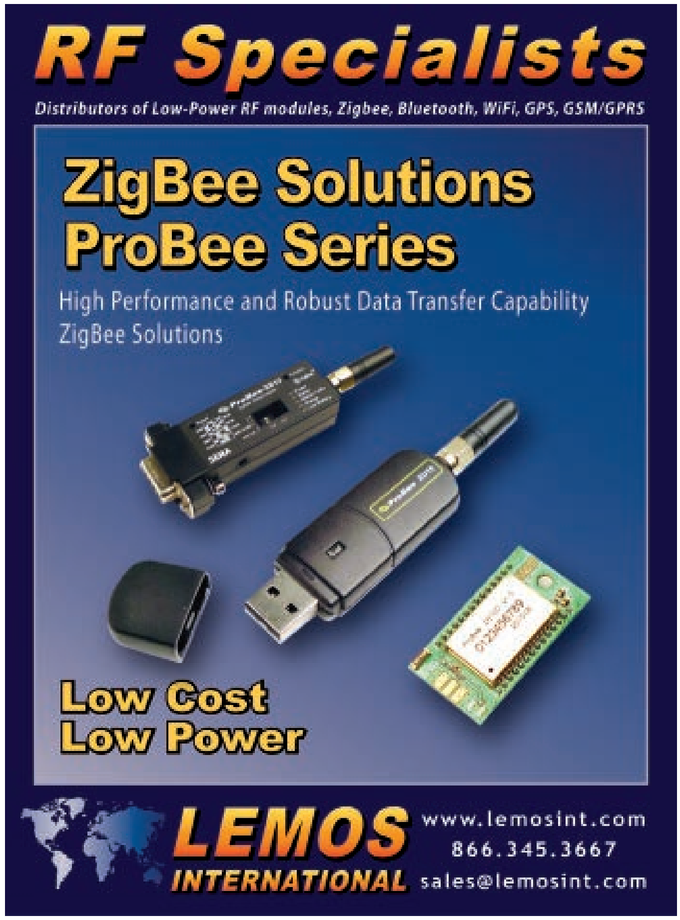
More info and sample evaluation kits are available at **www.cree.com/modules**. The company also markets blue and green LED chips, high brightness LEDs, lighting-class power LEDs, power-switching devices, and RF/wireless devices. ▲

JUST FOR FUN

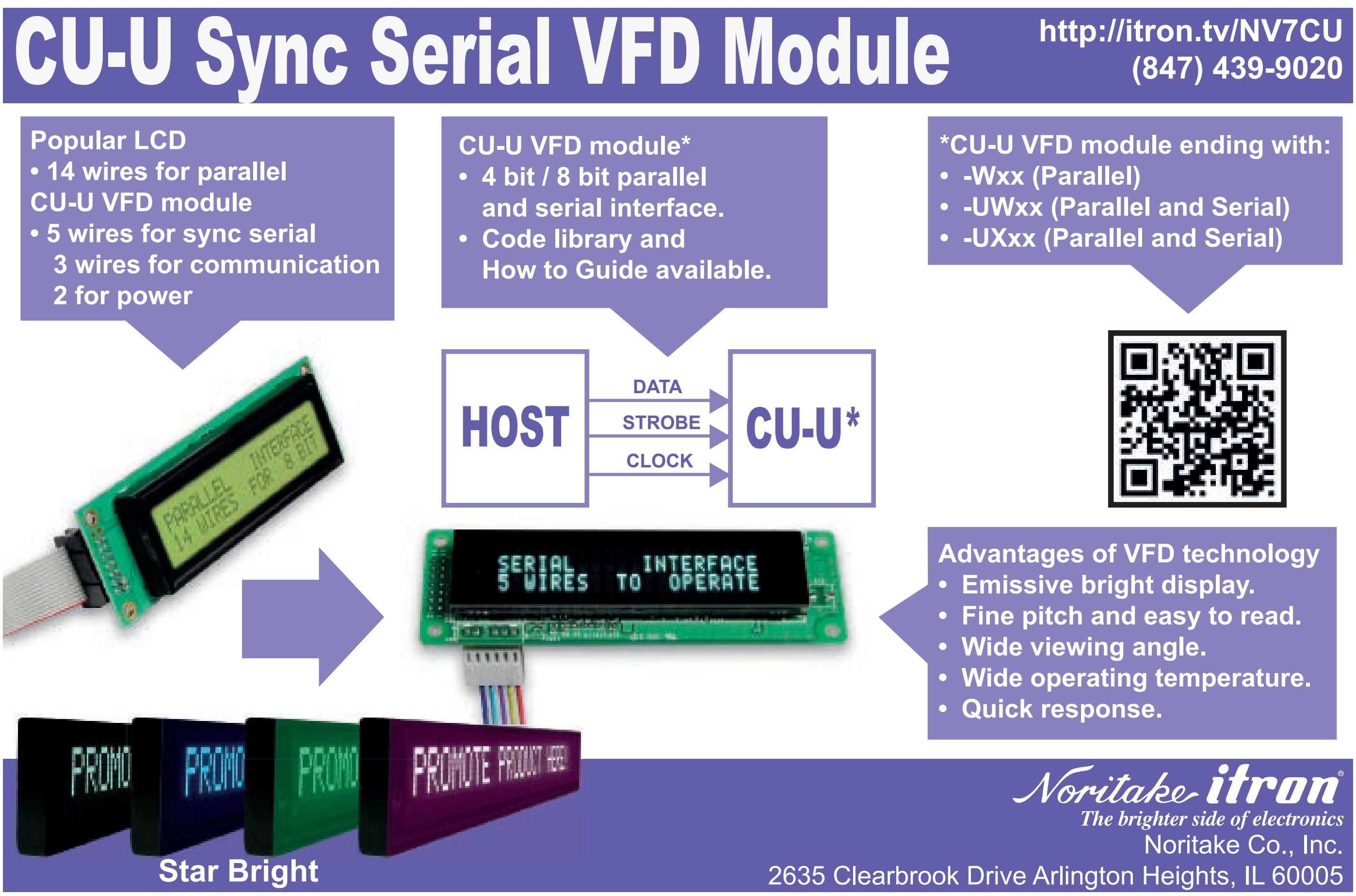
It's always nice to run into a gadget that has no practical function but is super cool, anyway. Such is the Firewinder LED Windlight, devised by the VP of Evil Schemes and Nefarious Plans (his real title, by the way) at ThinkGeek (**www.thinkgeek.com**). The Firewinder is a 100 percent wind-powered outdoor light, built from recyclable materials and designed to resist weather extremes. Drawing on wind power collected by a helical wing, it creates a spiral of light for your patio or front yard. It takes only a 4 MPH breeze to set the Firewinder spinning and power up the 14 ultra-bright LEDs, but it glows brighter if the weather gets more tempestuous. It comes with "high performance fixtures" for sturdy installation on a post or wall, and features a quick-release system so you can tuck it inside if you spot a tornado. It will run you $59.99, but what the heck. **NV**

■ The Firewinder LED Windlight visualizes wind energy.

Build Your Robot
The Way YOU Want!
Control Your Bot with an Android Powered Device!
With the Cross-link Robot Control System!
With this system, there's no need to write software! Plus, you get portability, connectivity, and reliability for one low cost! You can use ANY processor, ANY platform with ANY toolsuite, or run it directly from a PC or laptop!
The kit includes:
• 2CAN Ethernet Gateway
• CANipede Robot Control Module (RCM)
• TRENDnet N Pocket Router
• Cross-link/CAN Cables
CROSS the ROAD ELECTRONICS
Cross the Road Electronics, LLC
www.crosstheroadelectronics.com



RF Specialists
Distributors of Low-Power RF modules, Zigbee, Bluetooth, WiFi, GPS, GSM/GPRS
ZigBee Solutions
ProBee Series
High Performance and Robust Data Transfer Capability
ZigBee Solutions
Low Cost
Low Power
LEMOS INTERNATIONAL
www.lemosint.com
866.345.3667
sales@lemosint.com



CU-U Sync Serial VFD Module
http://itron.tv/NV7CU
(847) 439-9020
Popular LCD
• 14 wires for parallel
CU-U VFD module
• 5 wires for sync serial
3 wires for communication
2 for power
CU-U VFD module*
• 4 bit / 8 bit parallel and serial interface.
• Code library and How to Guide available.
*CU-U VFD module ending with:
• -Wxx (Parallel)
• -UWxx (Parallel and Serial)
• -UXxx (Parallel and Serial)
HOST
DATA
STROBE
CLOCK
CU-U*
PARALLEL INTERFACE
14 WIRES FOR 8 BIT
SERIAL INTERFACE
5 WIRES TO OPERATE
Advantages of VFD technology
• Emissive bright display.
• Fine pitch and easy to read.
• Wide viewing angle.
• Wide operating temperature.
• Quick response.
PROMOTE PRODUCT HERE
Star Bright
Noritake itron
The brighter side of electronics
Noritake Co., Inc.
2635 Clearbrook Drive Arlington Heights, IL 60005

BY RON HACKETT

USING EEPROMs TO STORE AND RETRIEVE DATA

WAY BACK IN THE THIRD INSTALLMENT OF THE PICAXE PRIMER (APRIL '08), I LISTED SOME OF THE PROJECTS WE WOULD BE COVERING IN THIS COLUMN. BY NOW, WE HAVE DISCUSSED MANY OF THOSE PROJECTS, BUT TWO IMPORTANT ONES STILL REMAIN: STORING DATA IN AN EXTERNAL EEPROM, AND IMPLEMENTING A REAL TIME CLOCK FOR ACCURATE TIME KEEPING. SINCE THE AXE401 SHIELD BASE THAT WE DISCUSSED LAST TIME INCLUDES A BUILT-IN EIGHT-PIN SOCKET FOR AN I^{2}C EEPROM, I THINK IT'S TIME WE TACKLED OUR EEPROM PROJECT.

In case you haven't been able to find the AXE401 anywhere, here's a direct link to it at the RevEd eStore: **www.techsupplies.co.uk/PICAXE**. Even if you don't yet have (or want) the PICAXE shield base, don't despair! Everything we'll discuss this month can easily be implemented on a simple breadboard circuit. I'll be using the updated 28X2 processor that comes with the AXE401, but the 18M2 or any X1- or X2-class processor will also work fine for this month's experiments. However, before we can delve into the details of EEPROM data storage, we need a basic understanding of the I^{2}C data bus and the relevant PICAXE BASIC commands; so let's get started.

UNDERSTANDING THE I2C DATA BUS

As you know, we have used the 5V version of the standard RS-232 serial protocol to implement a variety of PICAXE projects (including our serial LCD) and we'll most certainly continue to do so. However, the *Inter-Integrated Circuit* (I^{2}C) bus is another communication protocol that has some significant advantages for PICAXE projects:

- The physical implementation of the I^{2}C bus is simple; it only requires two I/O pins and two resistors.
- The same two I/O pins can be used to communicate with as many as 112 peripheral devices on the I^{2}C bus which greatly simplifies the design and programming of projects involving multiple peripheral devices.
- Many low cost I^{2}C devices are currently available, including EEPROMs, Flash RAM devices, real time clocks, LED drivers, LCD drivers, I/O expanders, ADCs, DACs, motor controllers, digital potentiometers, etc.
- Except for the M-class processors, all current PICAXE chips have built-in I^{2}C commands that greatly simplify I^{2}C data transfer. As the new M2-class chips become available, they will also support I^{2}C communication.

The two I/O lines that are required to implement the I^{2}C protocol are the serial data line (SDA) and the serial clock line (SCL). On the 28X2 processor that I'm using, the SDA line is available on pin C.4, and the SCL line is on C.3. If you use a different processor, check its pinout to see which two pins you need to use.

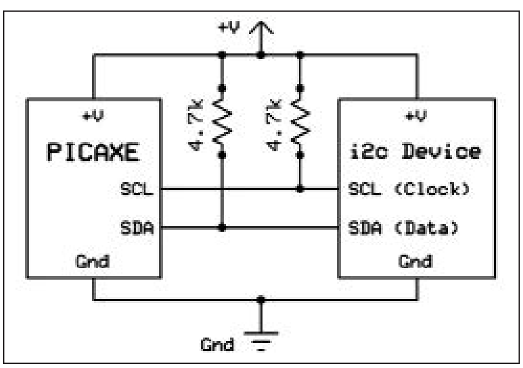


FIGURE 1. Implementing the I^{2}C bus for PICAXE projects.

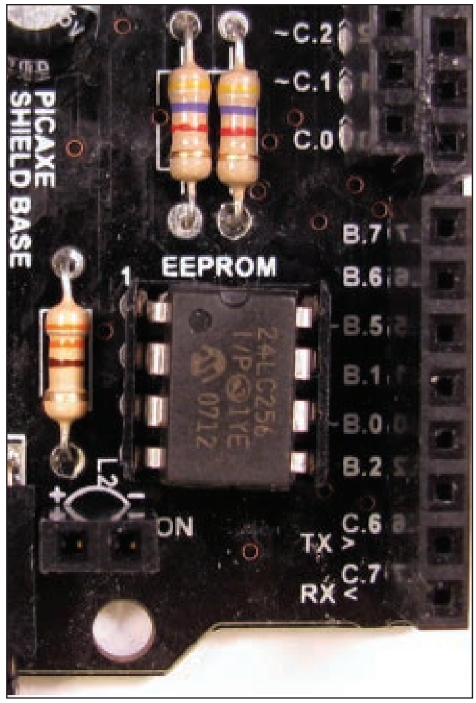
Figure 1 presents the complete implementation of the I^{2}C bus for PICAXE projects. As I have already mentioned, it's pretty simple; just connect the two PICAXE I/O lines to the corresponding I/O lines on the peripheral device, and add a 4.7K pull-up resistor to each line. If you're using the AXE401 shield base, there's already a place on the printed circuit board (PCB) for an EEPROM and two 4.7K pull-up resistors (just above the EEPROM; see **Figure 2**). These parts are optional, so you will need to add them yourself. If you're implementing a simple breadboard setup, don't forget to include the 4.7K pull-up resistors. On the I^{2}C bus, one device (typically a microcontroller) controls the bus and is referred to as the *master*; all other devices on the I^{2}C bus are *slaves*. The master and slaves can each transmit in both directions, but the master controls the timing of the transmissions. When I^{2}C capability was first added to PICAXE processors, they could only function as the master on the bus. However, the newer X2-class processors (20X2, 28X2, and 40X2) can now also function as slaves. This means that we can develop our own slave devices for use on any I^{2}C bus. We won't be exploring this capability in this month's Primer, but it does introduce some interesting new possibilities for PICAXE projects. You may want to read the relevant documentation in Section 2 of the PICAXE manual. Every slave on the I^{2}C bus is assigned a different address. This is what makes it possible for the master to individually communicate with many slaves using just two I/O lines. One specific slave address is always included in the communication, and all unaddressed slaves simply ignore the communication.

The master device also controls the clock speed on the I^{2}C bus. All current PICAXE processors can implement I^{2}C speeds of 100 kHz or 400 kHz. There are slave devices that can operate at even higher speeds, but all I^{2}C devices can also automatically function below their maximum speed. As a general rule, it's a good idea to use a 100 kHz clock speed to develop your project, and then when everything is functioning correctly, switch to 400 kHz.

CHOOSING AN EEPROM

EEPROMs are available from many different manufactures; and in many different storage sizes. The AXE401 is designed for use with the Microchip 24LCxxx line of I^{2}C EEPROMs. An EEPROM from another manufacturer might also work, but you would have to compare the datasheets to be sure.

Figure 3 is a listing of some of the Microchip EEPROMs that are readily available. There are smaller EEPROMs as well, but I think it makes more sense to use one with a fair amount of storage space. Also, the smaller Microchip EEPROMS are structured somewhat differently than the ones that are listed in **Figure 3**, which changes some of the details that I am about to discuss. Unless you are already familiar with the details of a different EEPROM, I would



■ FIGURE 2. The AXE401 EEPROM circuit.

recommend that you start with one of the chips listed in **Figure 3**.

I'm using a Microchip 24LC256 EEPROM because I happened to have one available. If you use a different size or an EEPROM from another manufacturer, make sure you have a copy of the relevant datasheet to check whether your EEPROM functions in the same manner as the 24LC256. To obtain the datasheet for any Microchip EEPROM, just go to **www.microchip.com** and enter the appropriate product number in the search box near the top-right corner of the page.

COMMUNICATING WITH THE 24LC256 EEPROM

PICAXE BASIC actually includes two different sets of I^{2}C commands. When I^{2}C capability was originally introduced on the 18X, 28X, and 40X processors, the following three commands were implemented: *i2cslave*, *readi2c*, and *writei2c*. However, those three processors have now all been superseded by more powerful versions (18M2, 28X2, and 40X2), and the original commands have been replaced by a new set of "hi2c" commands: *hi2csetup*, *hi2cout*, and *hi2cin*.

The original commands can still be used with the older processors, but if you're working with one of the more powerful processors, you will want to use the "hi2c" commands because they are more powerful and flexible.

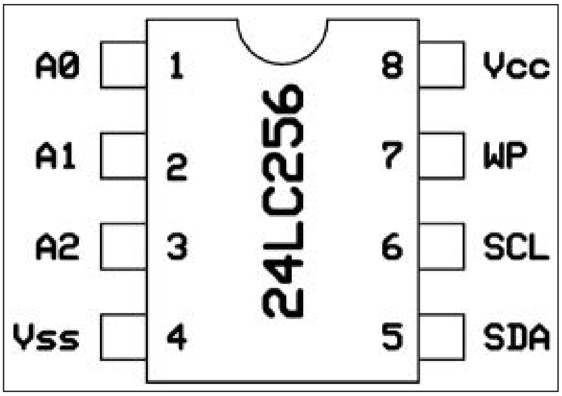
Understanding the *hi2csetup* command is our first priority because it must be properly executed before any I^{2}C communication can occur. There are actually three different versions or *modes* of the hi2csetup command. We won't be using two of them: *slave mode* (which configures the PICAXE processor as a slave on the I^{2}C bus) and the *off mode* (which simply "turns off" the I^{2}C capabilities so that the processor's SCL and SDA pins can be used for other purposes), so we won't discuss those two modes this time. (Don't forget, you can always refer to the relevant documentation in Section 2 of the PICAXE manual if you need more detail than I'm providing.)

The third mode (*master mode*) is the one we're interested in this time around; its complete syntax is *hi2csetup, i2cmaster, slaveaddress, mode, and addresslen*, so let's take a closer look at each of these terms:

- **hi2csetup** – This is simply the command name.
- **i2cmaster** – Tells the compiler that the PICAXE will be the master on the I^{2}C bus.
- **slaveaddress** – This one is a little complicated; we'll get to it shortly.
- **mode** – Assuming your processor is running at 16 MHz, this term should be replaced by either one

EEPROM	Bytes
24LC32A	4k
24LC65	8k
24LC128	16k
24LC256	32k
24LC512	64k

■ FIGURE 3. Selected Microchip EEPROMs.



■ FIGURE 4. The 24LC256 EEPROM pin-out.

of the following to set the I^2C bus clock. (If your processor is running at a different speed, you need to replace the "16" with the appropriate value.)

- **i2cfast_16** – 400 kHz
- **i2cslow_16** – 100 kHz

• **addresslen** – This term should be replaced by either one of the following to set the length of the slave's addressing method. (Refer to the slave's datasheet for which one is required.)

- **i2cbyte** – Slave uses byte addressing.
- **i2cword** – Slave uses word addressing. (All EEPROMs in **Figure 3** use i2cword.)

The *slaveaddress* parameter of the *hi2csetup* command still requires some explanation, but before we can understand how it's determined, we need to take a look at the 24LC256 pin-out which is presented in **Figure 4**. As you can see, four of the pins are straightforward: Vcc is the supply voltage; Vss is ground; SCL connects to the processor's I^2C clock line; and SDA connects to the processor's I^2C data line. WP is a *write protect* line; if it's pulled high, the EEPROM's data memory is write-protected (i.e., it can't be changed). If the WP line is pulled low, the memory contents can be freely updated. Of course, we're going to pull it low so that we can experiment with reading from **and** writing to the EEPROM.

The remaining three pins (A2, A1, and A0) can be pulled high or low to set a three-bit address for the EEPROM. Since there are eight possible three-bit addresses, we can access a total of eight different EEPROMs on the same I^2C bus — that's a lot of data storage! However, for our experiments this month, we're going to start small — just one EEPROM.

On the AXE401 shield base, the EEPROM's three address pins are all pulled low, so that's the configuration that I will be using. Of course, you can set them any way you want. However, if you do change the three-bit address, you will also need to modify the *hi2csetup* statement in this month's programs.

Keeping the EEPROM's three-bit address in mind, let's take a look at its complete eight-bit address which is presented in **Figure 5**. As you can see, the high nibble of the complete address is %1010. This value is assigned to all EEPROM devices by the I^2C protocol, so every EEPROM you purchase has this portion of its address already stored internally. (Each type of I^2C peripheral has a different high nibble assigned to its *slaveaddress*.)

The voltage levels we set for the EEPROM's A2, A1, and A0 pins determine bits 3, 2, and 1 of the slaveaddress. Bit 0 is the *read-write* bit; it's automatically set high for a read operation and low for a write operation. Both the PICAXE and the EEPROM ignore the value we place in bit 0 of the *slaveaddress*. They automatically manipulate it as necessary during read and write operations, so it doesn't matter what value we use. (For all our programming experiments this month, I used *%10100000* as the EEPROM's *slaveaddress*.)

FINALLY, WE GET TO PLAY

Congratulations — you made it to the fun part! It's time to set up your circuit for our experiments this month. As I already mentioned, I installed the 24LC256 EEPROM and the two 4.7K pull-up resistors on my AXE401. I'm also using a PCB version of the breadboard shield we worked on last time (see **Figure 6**). If you're interested in the bare PCB, see my website (**www.JRHackett.net**) for the details.

My circuit is ultra-simple — just a debugging LED on pin B.0. You really don't even need the LED for our experiments; I just like to see it blinking so I know each program made it through to the final loop. If you're not using the AXE401, just refer to **Figure 1** to set up your circuit on a breadboard. You will also need either an 18M2 or any X1- or X2-class processor to use the hi2c commands we're using this month.

We're going to experiment with four simple programs to demonstrate the processes of writing and reading data to and from the EEPROM. To save space, all four programs are available for downloading from the article link on the *Nuts & Volts* website. At this point, it would be a good idea to download them and print out a copy of each for reference during the following discussion.

When you have done that, you will see that some of the comments in each program end with a numbered "footnote" in brackets. In the following discussion, I will use the same numbering system to make it clear which program line is currently being discussed.

PROGRAM 1 (EEPROM_BYTEWRITE.BAS)

There are two modes of writing data to the 24LC256 EEPROM: *byte write* and *page write*. Our first program is a simple demonstration of the byte write mode. When you have completed your EEPROM circuit (breadboard or AXE401), download Program 1 to your PICAXE processor and give it a try. The following are more complete footnotes for the program:

1. I included a debugging LED on pin B.0, but it's not really necessary.

2. I'm running the processor at 16 MHz. If you have difficulties establishing reliable communication, you might try changing the mode parameter in the hi2csetup command

■ FIGURE 5. The EEPROM's complete eight-bit I²C address.

1	0	1	0	A2	A1	A0	R/$\overline{W}$

to i2cslow_16

3. This is the complete *hi2csetup* instruction. Note all the parameters from the above discussion. Also note that the clock speed of the processor (16 MHz) is included in the *mode* parameter (*i2cfast_16*). If you use a different clock speed, you also need to change the suffix of the *mode* parameter.

4. Here, we're specifying location 0 as the EEPROM address from which to fetch the data byte.

5. Next, we send the fetched byte to the terminal window for display. The "#" specifies we want raw digits, not an ASCII character. If the EEPROM has never been used before, the returned value will probably be 255.

6. Next, we store a capital "A" (ASCII 65) at location 0.

7. **Important:** The 24LC256 requires a delay of at least 5 mS to complete a write instruction. At 16 MHZ, that's *pause 10*. This required delay is a minor disadvantage of EEPROM data storage, especially if you're storing a lot of data.

8. Again, we read the data at location 0 and display it on the terminal window. You should see "65." If you omit the "#" character, you will see "A."

PROGRAM 2 (EEPROM_PAGEWRITEGOOD.BAS)

When you have the *EEPROM_ByteWrite.bas* program functioning correctly, we're ready to move on to the (much more useful) *page write* mode of writing data to the EEPROM. Earlier, I mentioned that the 24LC256 is structured into 64-byte *pages*. Since the 24LC256 is a 32K byte device, it has 512 (32768/64 = 512) storage pages. The *page write* mode enables us to write anywhere from two to 64 bytes using just one command.

Let's take a close look at Program 2 which demonstrates one method of using the page write mode to simplify the storage of larger amounts of data in the 24LC256. Without repeating the explanations from the previous program, here are the more complete footnotes for *EEPROM_PageWriteGood.bas*:

1. In the *hi2cout* instruction, note that I only had to specify the storage address (0) for the first data byte. The address is automatically updated for each additional character, greatly simplifying the task of storing multi-byte data (numeric or ASCII strings).

2. Another big advantage of the *page write* mode is that we only need one pause for the entire data string — not one for each character. In other words, we need one pause for each write operation, regardless of the length of the data string.

3. Here, we fetch and display the entire string by using the index of the *for...next* loop to update the storage address as we fetch and display each character.

4. Note that, this time, we're displaying ASCII characters, not raw data. Also note that the appearance of "garbage" characters in the terminal window indicates that you have somehow accessed unintended data locations which may contain any value from 0 to 255. Many of these values will produce the various garbage characters you may see. If this happens, check your code for errors.

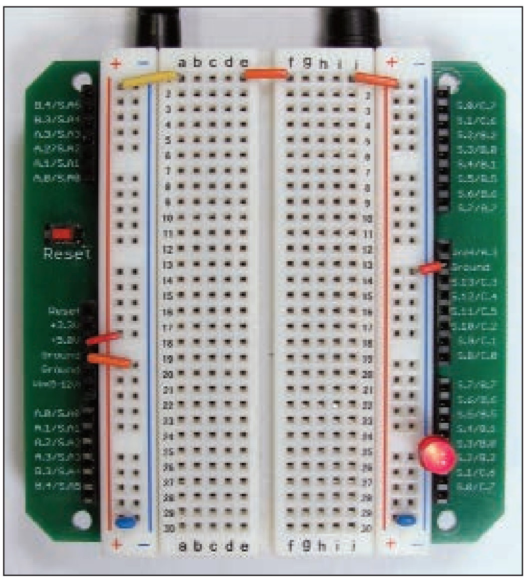
As you can see, the *page write* mode is a quick and easy way to store multi-byte numeric or ASCII data. I think we can make it even a little faster, based on the fact that the *hi2cin* command functions similarly to *hi2cout*, in that it's not necessary to update the EEPROM address for each data byte that we are reading. For example, try replacing the *for...next* loop in the *EEPROM_PageWriteGood.bas* program with the following code snippet:

```
hi2cin 0,(char)
sertxd (char)

for index = 1 to 33
  hi2cin (char)
  sertxd (char)
next index
```

In this version of the program, we're fetching the first data byte (from location 0) and then executing the *for...next* loop 33 times (rather than 34) to fetch the remaining data. The main difference is that we're not including the data address in the remaining 33 *hi2cin* instructions; the I²C protocol takes care of that automatically, just as it does for the *hi2cout* command. I suspect this approach may be a little faster, but I haven't yet tested this assumption.

So far, so good, but now we come to the point where we need to understand a basic rule about writing data to any page-structured EEPROM: **Never attempt to write data across a page boundary**. If you do, your data will be corrupted because it will automatically "wrap around" the page.

For example, if you are writing a 26-byte data string to page 0



■ FIGURE 6. The complete EEPROM circuit on a breadboard shield.

(locations 0 to 63) of the 24LC256 and you start to write data at address 50, there won't be enough room to store the string in page 0. Only 14 bytes can be stored in locations 50 to 63, so the remaining 12 bytes won't fit in page 0. You might think that the extra data bytes would "spill over" to page 1, but that's not what happens!

(At this point, I should also mention that, unlike write operations, read operations **can** freely cross page boundaries. In other words, you can read and process the entire contents of the EEPROM using one large loop. I haven't included a program that demonstrates this useful feature, so you may want to do some experimenting on your own.)

PROGRAM 3 (EEPROM_PAGEWRITEGOOF.BAS)

Our next program clearly demonstrates why we need to be careful to never attempt to write data across a page boundary. Before running it, read the following footnotes for Program 3 so that you know what to look for in the results that are displayed in the terminal window:

1. The two *hi2cout* statements simply fill page 0 with asterisks, so that the results of our mistake will be easier to see.

2. Here, we try to write across the page boundary by starting to write a 26-character string at location 50.

3. Here's what happens: "A" through "N" (14 characters) fit, but "O" through "Z" (12 characters) wrap around to the beginning of page 0.

All our programming experiments thus far have stored data of a specified length, but there are many situations where you don't want to bother figuring out the length of every collection of data you need to store. In Programs 2 and 3, for example, I had to manually count the number of characters in the string before I could store it. This is a minor nuisance at best, so let's take a look at one way of avoiding that task.

PROGRAM 4 (EEPROM_VARLENGTHTEXT.BAS)

In this program, we're going to define a special "end of text" character, and use it to indicate when we have reached the end of a data string. We could use any non-printing character, but the ASCII code already defines a value of 4 as "end of text," so that's what we'll use as well. Here are the more complete footnotes for Program 4:

1. This is simply our constant declaration for the "end of text" character.

2. This group of four constant declarations specifies the location of the first byte of each EEPROM page that the program uses for data storage.

3. Each of the four *hi2cout* statements in the program stores one of our data strings. Don't forget that our "end of text" character is also going to be stored in the EEPROM, so this approach limits the maximum length of the actual data that we can store with one *page write* operation to 63 bytes.

4. At this point, we have stored the four text strings, so we're ready to fetch and display them. We're going to use a subroutine for this purpose because we'll be doing the same thing four times. Each time before we enter the subroutine, we fetch the first data byte from the specified page. This way, when we get to the subroutine, the data location will be automatically updated for each byte that we fetch.

5. Since we have fetched the first character in the string before entering the subroutine's *do...until* loop, we need to display it in the terminal window before fetching the next character.

6. Here, we continue displaying each character and fetching the next one until we fetch the "end of text" character, at which point the *do...until* loop terminates. As a result, the "end of text" character is never displayed.

7. Just before exiting the subroutine, we move the terminal window's cursor to the beginning of the next line so that we're ready to display another string of text.

I realize that our programming experiments this month have largely focused on the storage and retrieval of text strings. This is probably because I have been thinking about how to use EEPROMs to store and display text on an LCD. However, I would be remiss if I didn't mention the fact that EEPROMs can also store large amounts of numeric data. For example, suppose you have a data collection system that measures the outdoor temperature every 15 minutes, and you want to be able to store the data that accumulates over a long period of time. As long as you're careful to avoid trying to write data across a page boundary, this is an easy task. For example, to store one hour's worth of data, you can use the variable names directly, and write:

```
hi2cout p500, (time1, temp1, …,
time4, temp4)
```

When you're ready to retrieve the stored data, you can write:

```
hi2cin p500, (time1, temp1, …,
time4, temp4)
```

We're just about out of space, but I do want to mention the fact that both the *hi2cout* and *hi2cin* commands include an optional *newslave* parameter that enables us to change the *slaveaddress* "on the fly" so that we can easily communicate with more than one device on the I²C bus.

In an upcoming installment of the Primer, we'll be taking a look at an I²C real time clock. At that point, we'll also cover the use of the *newslave* parameter, but in the meantime you may want to check out the details on your own. If so, see the *hi2cout* and *hi2cin* documentation in Section 2 of the PICAXE manual.

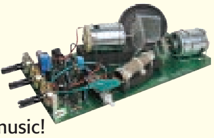
I hope this month's Primer has started you thinking about how you can use EEPROM data storage in your own PICAXE projects. If you come up with an interesting application for EEPROM storage, I would love to hear about it.

I also realize that I still haven't returned to our capacitive-touch doormat project. I guess I've been having too much fun playing with the AXE401 shield base and my breadboard shield. It's like I say to my wife about my chore list: "I'll get to it, I **really** will!"

See you next time ... **NV**

LED DISPLAY PROJECTS ARE FUN!

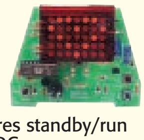
Laser Light Show



Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1 Laser Light Show Kit $49.95

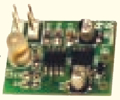
Rolling Message Display



35 bright LEDs provide a rolling message display, one character at a time! Built-in message editor allows you to create up to 16 characters. Display features adjustable speed. Also features standby/run to restart the display. Runs on 9-12VDC.

MK124 Rolling LED Message Display Kit $16.95

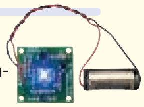
SMT Multi-Color Micro Blinky



Learn the latest multi-color LED technology as well as SMT construction techniques! Uses an 8-pin microcontroller to display up to 16,777,216 possible colors! Makes a great hi-tech attention getter for badges and caps. Runs on 6-12VDC.

SBRGB1 SMT Color LED Blinky Kit $29.95

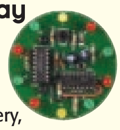
LED Switcher Blinky



A great little display and a great switching power supply learning kit besides! We have you start with 1.5VDC and build a switcher to step it up to 5VDC to power the display. But does it blink? Sure, but at 140kHz! Requires a single AA battery, not included.

LSW1 LED Switcher Blinky Kit $14.95

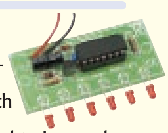
Wheel of Fortune LED Display



Just like the casino game, just push the button and 10 LEDs start spinning. They gradually slow down until the winner remains lit! Features auto on/off to save battery life. Requires a standard 9V battery, not included.

MK152 Wheel of Fortune LED Kit $7.95

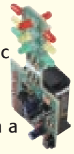
Mini LED Light Chaser



This little kit flashes six high intensity LEDs sequentially in order. Just like the K8032 to the right does with incandescent lights. Makes a great mini attention getter for signs, model trains, and even RC cars. Runs on a standard 9V battery.

MK173 Mini LED Light Chaser Kit $15.95

LED Traffic Signal



Impress your friends with this neat 4-way traffic signal! Bright red, green, and yellow LEDs are positioned just like a traffic signal. Adjustable timing permits you to change the auto sequence. Great for model railroads! Runs on a standard 9V battery

MK131 LED Traffic Signal Kit $7.95

High Power LED Driver

This little driver makes it easy to power the latest high power, high intensity LEDs without problems. Features short circuit protection and will deliver an accurate and constant 350mA or 700mA with an input of 1-12VAC or 9-18VDC. Power consumption is 650mA.

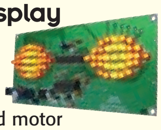
K8071 High Power LED Driver Kit $14.95

High Power LED Strobe Light

Replace that old Xenon strobe light with new LED technology! A bright plug in 3x3 light array provides a brilliant flash. Flash rate is variable by time, or bass or treble audio triggering. Optional display boards available.

LEDS1C High Power LED Srobe Kit $49.95

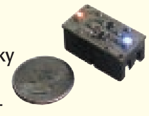
Blinky-Eyes Animated Display



The ultimate animated LED display that features 66 brilliant LEDs! The display is microprocessor controlled to provide numerous animated displays. Features auto light sensor and motor output. Runs on 9-20VDC.

BE66 Blinky-Eyes LED Display Kit $59.95

High Intensity SMT Blinky



A high-tech remake of our classic blinky kit! Alternating super bright red and blue SMT LEDs provide the classic alternating flash sequence. Great SMT learning kit and we even include spare parts so you can lose them! Runs on 2 LR55 button cells included.

BL2 High Intensity SMT Blinky Kit $17.95

Red LED Blinky

A great 1st kit to tweak a child's interest in electronics! Thousands and thousands have been sold and built by scouts, schools, and hobbyists young and old, creating a classic alternating flash for caps to name badges. Requires 3-12VDC, or a 9V battery (not inc.)

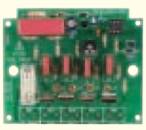
BL1 Red LED Blinky Kit $7.95

Shaking Electronic LED Dice

You don't have to throw it, you just shake it, and it acts like a regular dice! Number combinations roll out slowly and finally flashes the final "winning number". Micro controlled and runs on one 3V battery cell, not included. (Get a pair for snake eyes or box cars!)

MK150 Shaking Electronic Dice Kit $13.95

Running Light Controller



Controls and powers 4 incandescent lights so they appear to "travel" back and forth (Like the hood on KITT!). Great for the dance floor or promotional material attention getters, exhibits, or shows. Runs on 112-240VAC.

K8032 4-Channel Running Light Kit $38.95

Electronic LED Decision Maker

No more coin flipping, you can make your decision with a push of a button! One push and a green YES and red NO LED begins to flash, and slowly decreases until one remains lit! If you need a yes or no answer, just push the button! Runs on a 9V battery.

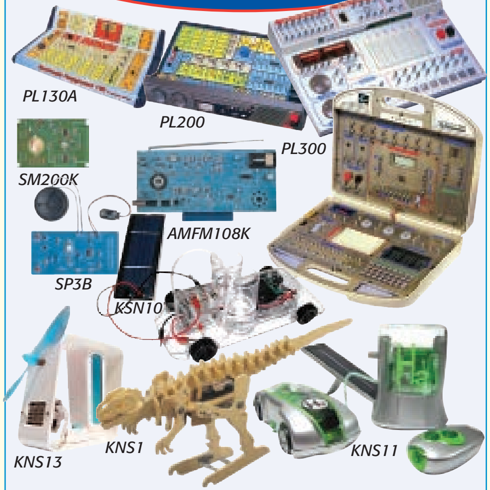
MK135 Electronic Decision Maker Kit $6.95

LED Digital Clock

Not your every-day digital clock! Features 7-segment LED displays in a modern attractive case. Auto time base gives you the option of line or internal clock. 12/24 hour mode. Runs on 9V AC or DC and uses 3 AA batteries for backup.

MK151 LED Digital Clock Kit $29.95

The Learning Center!



Beginners To Advanced... *It's Fun!*

- ✔ **Learn and build!**
- ✔ **130, 200, 300, & 500 in one electronic labs!**
- ✔ **Practical through hole and SMT soldering labs!**
- ✔ **Integrated circuit AM/FM radio lab!**
- ✔ **Fuel Cell, Solar Hydrogen, and Bio-Energy labs!**
- ✔ **Beginner's non-soldering kits!**

For over 3 decades we've become famous for making electronics fun, while at the same time making it a great learning experience. As technology has changed over these years, we have continued that goal!

PL130A Gives you 130 different electronic projects together with a comprehensive learning manual describing the theory behind all the projects.

PL200 Includes 200 very creative fun projects and includes a neat interactive front panel with 2 controls, speaker, LED display and a meter.

PL300 Jump up to 300 separate projects that start walking you through the learning phase of digital electronics.

PL500 The ultimate electronics lab that includes 500 separate projects that cover it all, from the basics all the way to digital programming.

SP3B Whether young or old, there's always a need to hone your soldering skills. Either learn from scratch or consider it a refresher, and end up with a neat little project when you're done!

SM200K Move up to Surface Mount Technology (SMT) soldering, and learn exactly how to solder those tiny little components to a board!

AMFM108K We not only take you through AM and FM radio theory but we guide you through IC's. When you're done you've built yourself an IC based AM/FM radio that works great!

KNS10 With a reversible PEM fuel cell that combines electrolysis and power conversion into a single device you end up building your own fuel cell car! Learn tomorrows technology today!

KNS11 Learn alternative fuel technology while you build your own H-Racer car and refueling station!

KNS13 Convert ethanol alcohol to run a PEM fuel cell and watch it all work in front of your eyes!

KNS1 A great beginner's kit for the dinosaur enthusiast in the family, young and old! A wooden hobby kit that teaches motor and gear driven operation that requires no soldering.

PL130A	**130-In-One Lab Kit**	**$39.95**
PL200	**200-In-One Lab Kit**	**$69.95**
PL300	**300-In-One Lab Kit**	**$89.95**
PL500	**500-In-One Lab Kit**	**$219.95**
SP1A	**Through Hole Soldering Lab**	**$9.95**
SM200K	**SMT Practical Soldering Lab**	**$22.95**
AMFM108K	**AM/FM IC Lab Kit & Course**	**$34.95**
KNS10	**Fuel Cell Car Science Kit**	**$82.95**
KNS11	**H-Racer & Refueling Station Kit**	**$144.95**
KNS13	**Bio-Energy Fuel Cell Kit**	**$129.95**
KNS1	**Tyrannomech Motorized Kit**	**$17.95**

Vintage Battery Eliminator

Collectors come across some great deals on antique battery-powered radios, but how to power them is a real problem. Many classic radios operated on batteries only, and in many cases a series of three batteries for each radio were required!

The new ABCE1 Battery Eliminator gives you an easy way to replace all these batteries with a simple household AC power connection to resurrect your vintage antique radios! Provides "A" filiment, "B" plate, and "C" control grid supplies, which are all isolated from each other. Complete with aluminum case. Runs on 110-240VAC.

ABCE1 Vintage Radio Battery Elim Kit $199.95

Four-Mode Keyless Entry Test Set

NEW!

Just like the days of "plugs, points, and condenser" are over, so are the days of having the hardware store grind out a spare key for your car! Now when your keyless access system doesn't work, you need to accurately detect what part of the system is malfunctioning. This could be anything from a dead battery in the key fob, a "brain-dead" key fob, to malfunctioning sensors, antennas, or other system components in the vehicle. Until now there was no way to determine where the system was failing.

Testing your system is easy. To test the complete 125 kHz/315 MHz communications path just stand close to the vehicle with the WCT3 and your key fob in hand. Press the test button and the WCT3 will detect and display the presence of the vehicle's 125kHz/20KHz signal and, if they "handshake", will also detect and display the presence of your key fob's 315MHz return signal. You can independently test key fob only signals (panic, lock, trunk, etc.) by holding the key fob near the WCT3, pressing the test button, and pushing the function button on the key fob. The same functionality testing can be done with IR key fobs. The modulated IR signal is detected and will illuminate the IR test LED on the test set. If you know a few "secrets" you can also see if the tire pressure sensors/transmitters are generating signals or the built-in garage door opener in your rear view mirror is transmitting a signal! Runs on a standard 9V battery. Also available factory assembled & tested.

WCT3 Four-Mode Keyless Entry Test Set Kit $59.95

Passive Aircraft Monitor

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1 Passive Aircraft Receiver Kit $89.95

Voice Activated Switch

Voice activated (VOX) provides a switched output when it hears a sound. Great for a hands free PTT switch or to turn on a recorder or light! Directly switches relays or low voltage loads up to 100mA. Runs on 6-12 VDC.

VS1 Voice Switch Kit $9.95

OBDII *CarChip Pro*

The incredible OBDII plug-in monitor that has everyone talking! Once plugged into your vehicle it monitors up to 300 hours of trip data, from speed, braking, acceleration, RPM and a whole lot more. Reads and resets your check engine light, and more!

8226 CarChip Pro OBDII Monitor-Asmb $79.00

RF Preamplifier

The famous RF preamp that's been written up in the radio & electronics magazines! This super broadband preamp covers 100 KHz to 1000 MHz! Unconditionally stable gain is greater than 16dB while noise is less than 4dB! 50-75 ohm input. Runs on 12-15 VDC.

SA7 RF Preamp Kit $19.95

Touch Switch

Touch on, touch off, or momentary touch hold, it's your choice with this little kit! Uses CMOS technology. Actually includes TWO totally separate touch circuits on the board! Drives any low voltage load up to 100mA. Runs on 6-12 VDC.

TS1 Touch Switch Kit $9.95

Doppler Direction Finder

Track down jammers and hidden transmitters with ease! 22.5 degree bearing indicator with adjustable damping, phase inversion, scan and more. Includes 5 piece antenna kit. Runs on 12VDC vehicle or battery power.

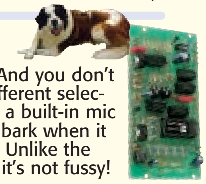
DDF1 Doppler Direction Finder Kit $169.95

Mad Blaster Warble Alarm

If you need to simply get attention, the "Mad Blaster" is the answer, producing a LOUD ear shattering raucous racket! Super for car and home alarms as well. Drives any speaker. Runs on 9-12VDC.

MB1 Mad Blaster Warble Alarm Kit $9.95

Electronic Watch Dog



A barking dog on a PC board! And you don't have to feed it! Generates 2 different selectable barking dog sounds. Plus a built-in mic senses noise and can be set to bark when it hears it! Adjustable sensitivity! Unlike the Saint, eats 2-8VAC or 9-12VDC, it's not fussy!

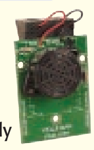
K2655 Electronic Watch Dog Kit $39.95

Retro Nixie Tube Clock

Genuine Nixie tubes popular in the 50's brought back in one of the neatest digital clocks around today! This compact unit is great to build into your own enclosure or order our custom plexiglass enclosure. Requires 12VDC 500mA.

K8099 Compact Nixie Clock Kit $129.95

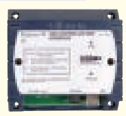
Water Sensor Alarm



This little $7 kit can really "bail you out"! Simply mount the alarm where you want to detect water level problems (sump pump)! When the water touches the contacts the alarm goes off! Sensor can even be remotely located. Runs on a standard 9V battery.

MK108 Water Sensor Alarm Kit $6.95

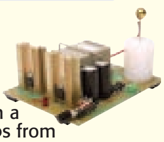
USB DMX Interface



Control DMX fixtures with your PC via USB! Controls up to 512 DMX channels each with 256 different levels! Uses standard XLR cables. Multiple fixtures can be simply daisy chained. Includes Light Player software for easy control. Runs on USB or 9V power.

K8062 USB DMX Interface Controller Kit $67.95

HV Plasma Generator



Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

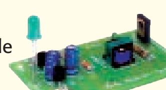
PG13 HV Plasma Generator Kit $64.95

Air Blasting Ion Generator

Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7 Ion Generator Kit $64.95

Tickle-Stick Shocker



The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 Tickle Stick Kit $12.95

Speedy Speed Radar Gun

Our famous *Speedy* radar gun teaches you doppler effect the fun way! Digital readout displays in MPH, KPH, or FPS. You supply two coffee cans! Runs on 12VDC or our AC125 supply.

SG7 Speed Radar Gun Kit $69.95

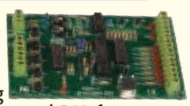
Tri-Field Meter Kit

THE GHOST DETECTOR!

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of ghosts! Req's 4 AAA batteries.

TFM3C Tri-Field Meter Kit $74.95

USB Experimenter's Kit



Get hands-on experience developing USB interfaces! 5 digital inputs, 8 digital outputs, 2 analog I/O's! Includes diagnostic software and DLL for use with Windows based systems. The mystery is solved with this kit!

K8055 USB Experimenter's Kit $49.95

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Assmb.

PR2 Broadband RF Preamp $69.95

Electret Condenser Mic

This extremely sensitive 3/8" mic has a built-in FET preamplifier! It's a great replacement mic, or a perfect answer to add a mic to your project. Powered by 3-15VDC, and we even include coupling cap and a current limiting resistor! Extremely popular!

MC1 Mini Electret Condenser Mic Kit $3.95

Sniff-It RF Detector Probe

Measure RF with your standard DMM or VOM! This extremely sensitive RF detector probe connects to any voltmeter and allows you to measure RF from 100kHz to over 1GHz! So sensitive it can be used as a RF field strength meter!

RF1 Sniff-It RF Detector Probe Kit $27.95

3-In-1 Multifunction Lab

The handiest item for your bench! Includes a RoHS compliant temp controlled soldering station, digital multimeter, and a regulated lab power supply! All in one small unit for your bench! It can't be beat!

LAB1U 3-In1 Multifunction Solder Lab $134.95

Q&A

WHAT'S UP:

Join us as we delve into the basics of electronics as applied to every day problems, like:

- ✔ Zeners at High Frequency
- ✔ LED Flasher Circuit
- ✔ Refrigerator Door Alarm

■ WITH RUSSELL KINCAID

In this column, I answer 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**

ZENERS AT HIGH FREQUENCY

Q My question is why don't you use zener diodes in high frequencies ("high frequency" is the formal HF frequency range of 3 to 30 MHz)?

— Selahattin SADOGLU

A Hmm, I had not thought about that; I don't know how anyone would use a zener in high frequency. I suspect the reason is that the zener has too much capacitance. To test that theory, I measured some diodes:

Part #	Type	Package	Capacitance at one volt bias
1N751	Zener	DO-35	130 pF
1N4738A	Zener	DO-41	230 pF
1N5846A	Zener	TO-92	135 pF
1N4005	Rectifier	DO-41	28 pF
1N457	Diode	DO-7	8 pF
1N4148	Diode	DO-35	11 pF
1N4937	Fast Rectifier	DO-41	30 pF

As you can see, the zener has a lot of capacitance, but if you wanted to use one as a DC coupling device, I don't see any reason not to. Tektronix — in their 540 series of oscilloscopes — used neon lamps as if they were 75 volt zeners to couple between vacuum tubes; these covered DC to 30 MHz.

TRANSISTOR REPLACEMENT NEEDED

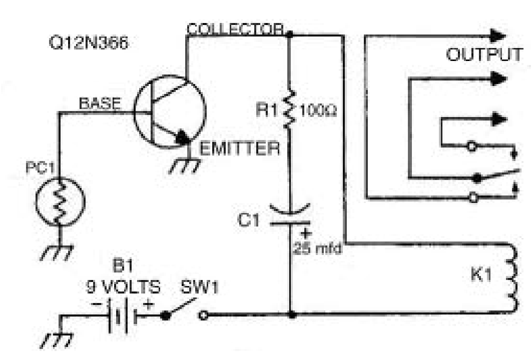
Q Can you PLEASE modify the diagram in **Figure 1** with a cheaper plastic transistor rather than the $7 NTE 103 Germanium TO-5 power output driver transistor? I enjoyed building this schematic, especially because of hearing the relay 'clicking' when the photocell was encountering light. There must be some cheaper plastic transistor that can perform the same function. Thank you.

— Don Franklin

A The 2N366 was a lousy transistor. That is why it is not made anymore. The collector-base leakage is high which will turn the transistor on if the base is open. This circuit takes advantage of that characteristic; in the dark, the photocell is nearly an open circuit which allows the transistor to turn on. When light is on the photocell, the resistance is low, bypassing the base-emitter junction and turning the transistor off. A silicon NPN like the 2N3904 (seven cents instead of $7) won't work because it does not have enough leakage. A resistor from +9 volts to the base will turn the transistor on so the photocell can turn it off. I think R = 510K will work.

LED FLASHER CIRCUIT

Q I want to make a high power LED flasher circuit with a PIC microcontroller by using 15 three watt power LEDs. We will have two pushbuttons. When the first button is pressed, the flasher will blink four times per second and will wait one second. When the second button is pressed, the flasher will blink two times per second and will wait one second. By the way, it will be working with an auto battery (12



B1	Nine volt battery
C1	25 mfd, 15 volt electrolytic capacitor
K1	4,000 ohm SPDT relay (Advance SO/IC/4000D or equiv.)
PC1	Photocell (International Rectifier B2M or equiv.)
Q1	2N366 transistor = NTE 103
R1	100 ohm resistor
SW1	SPST switch

■ FIGURE 1

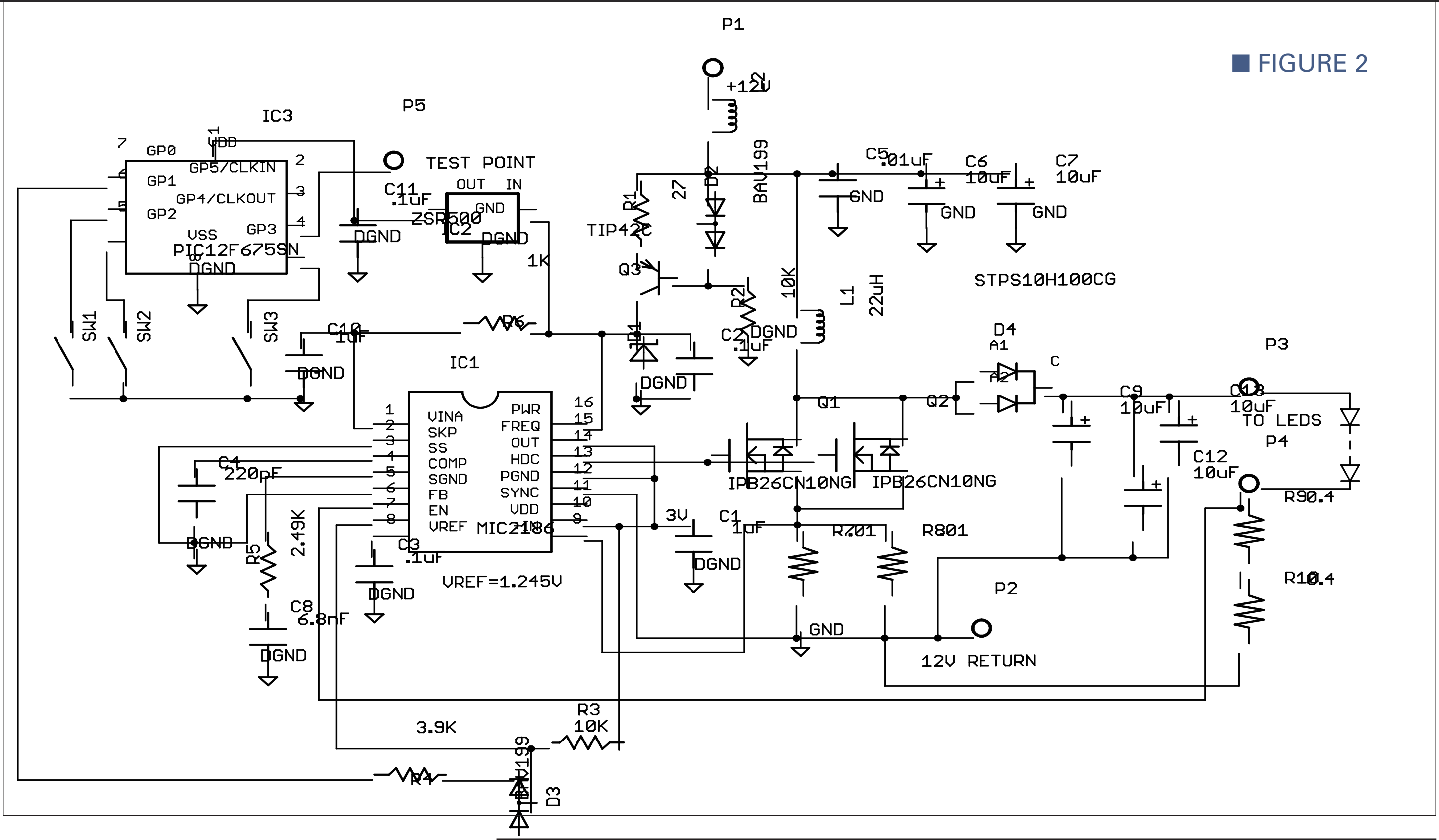


FIGURE 2

volt/72 amp) and will be used in an ambulance. How can I do that circuit?

— Selahattin SADOGLU

A This is a job for a microcontroller and simple enough that I can do it. The schematic is in **Figure 2**. Since it is operating from the vehicle's 12 volts, it must be protected from the "load dump" transients which can reach 60 volts. Since the blue LEDs drop over four volts, it is only possible to have two in series on 12 volts. Rather than having eight parallel sets of two series LEDs, I opted for all 15 in series, and generate 63 volts at 750 mA with feedback to regulate the current. IC1 is a current mode PWM device which takes feedback from R7 and R8 for pulse to pulse control. DC feedback to pin 6 controls the output current as sensed by R9 and R10. Q1 and Q2 are in parallel because the peak current can be quite high. Q3 protects from high voltage transients by providing a constant current drive to the PWM circuit and five volt regulator (IC2). IC3 is the PIC12F675

```
'****************************************************************
'*  Name      : FLASHING LEDS                                   *
'*  Author    : Russell Kincaid                                 *
'*  Notice    : Copyright (c) 2011 Russell Kincaid,             *
'*            : Kincaid Engineering All Rights Reserved         *
'*  Date      : 5/5/2011                                        *
'*  Version   : 1.0                                             *
'*  Notes     : TWO BUTTONS CONTROL THE FLASHING, RESET STOPS THE *
'*            : FLASHING.  WHEN THE FIRST BUTTON IS PUSHED, THE  *
'*            : FLASHING IS   PER SECOND; WHEN THE 2ND BUTTON IS *
'*            : PUSHED THE FLASHING IS 2 PER SECOND. HAVE TO PUSH *
'*            : RESET TO CHANGE FLASH RATE                      *
'****************************************************************
   REM DEVICE = 12F675
   REM CONFIGURATION: INTOSC CLOCKOUT, WDT DISABLED, PWRUP TIMER ENABLED,
   REM MCLR FUNCTION: RESET, BROWNOUT DISABLED, NO PROTECTION
   CMCON = 7                  'SETS DIGITAL MODE
   ANSEL = 0                  'GPIO.0 TO GPIO.3 SET AS DIGITAL
   DEFINE OSCCAL 1K 1         'TO SAVE OSCILLATOR CALIBRATION
   OPTION_REG = 0             'WEAK PULLUPS ENABLED
   VRCON.7 = 0                'TURN OFF VOLTAGE REFERENCE
   TRISIO = %00001110         'GP0 IS OUTPUT (PIN 7) PINS  , 5, 6 ARE INPUT
   C VAR WORD                 'DEFINE VARIABLE C (COUNT)

   LOW GPIO.0                 'INITIAL CONDITION
   START:                     'WAIT FOR BUTTON TO BE PUSHED
       IF GPIO.1 = 0 THEN FOUR 'PIN 6 MOMENTARIALY TO GROUND
       IF GPIO.2 = 0 THEN TWO  'PIN 5 MOMENTARIALY TO GROUND
       GOTO START

   FOUR:                      'FLASH ON 125mS, OFF 125mS
       FOR C = 1 TO 4
       HIGH GPIO.0
       PAUSE 125
       LOW GPIO.0
       PAUSE 125
       NEXT C
       GOTO FOUR

   TWO:                       'FLASH ON 250mS, OFF 250mS
       FOR C = 1 TO 2
       HIGH GPIO.0
       PAUSE 250
       LOW GPIO.0
       PAUSE 250
       NEXT C
       GOTO TWO
   END
```

FIGURE 3

```
FLASHING LEDS.hex
:020000040000FA
:100000003B28A301A200FF30A207031CA307031C87
:1000100036280330A100DF300F200328A101E83E7D
:10002000A000A109FC30031C1828A00703181528FC
:10003000A0076400A10F152820181E28A01C222844
:1000400000002228080A20001302628A80023086A
:100050002102031D2D282208200204300318013O3C
:10006000031902302805031DFF30362883130313BC
:10007000083126400080FF23831690008312073068
:10008000990083169F01810199130E308500831218
:1000900005108316051083126400851C5328640024
:1000A000051D70284C280130B800B9013808A0009F
:1000B0003908A100A301043023200031D6F28051473
:1000C000083160510 7D30831201200510831605105C
:1000D0007D3083120120B80A0319B90F562853281E
:1000E000130B800B9013808A0003908A100A30107
:1000F00002302320031D8C28051483160510FA30C6
:10010000831201200510831605 10FA308312012096
:0E011000B80A0319B90F7328702863008D28F0
:02400E00FC3F75
:00000001FF
```

■ FIGURE 4

which drives the enable pin of IC1 to flash the LEDs. The circuit uses the relation: E = L dI/dT which can be solved for dI = E*dT/L. First, the circuit charges the inductor with current, then the current is directed to the load when the transistors turn off.

The voltage at the output is V = I*R + Vd, where Vd is the diode drops. If there is no load, the voltage will be very high, possibly causing damage to the inductor or transistors.

Figure 3 is the PICbasic program; it is mostly self-explanatory. The subroutines (four: and two:) just keep going until you press the reset button or turn off the power.

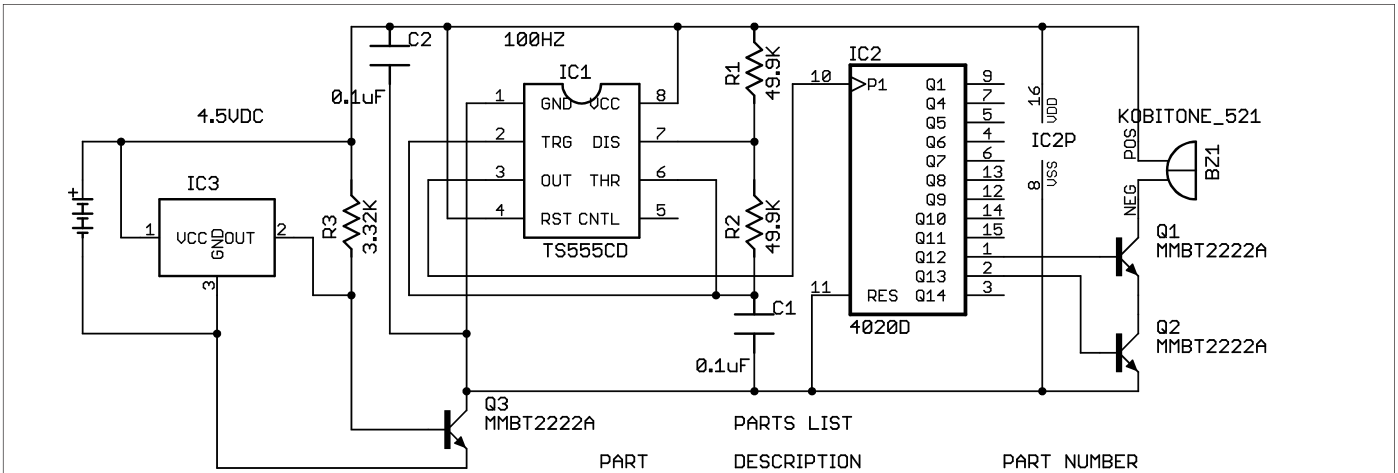
Figure 4 is the hex file which you will need to program the PIC12F675 (or I can send a programmed chip to you for $5.00). **Figure 5** is the Parts List.

BLUE LED FLASHER PARTS LIST

PART	DESCRIPTION	PKG	MOUSER PART#
R1	27.4 ohms, 1%, 1/8W	1206	290-27.4-RC
R2, R3	10K, 1%, 1/8W	1206	290-10K-RC
R4	3.92K, 1%, 1/8W	1206	290-3.92K-RC
R5	2.49K, 1%, 1/8W	1206	290-2.49K-RC
R6	1K, 1%, 1/8W	1206	290-1.0K-RC
R7, R8	.01 ohm, 1W, 1%	2512	71-WSL2512 R0100FEA
R9, R10	0.4 ohm, 1W, 1%	2512	71-WSL2512R 4000FEA
C1, C10	1 µF, 25v, 10%, CERAMIC	1206	80-C1206X10 5K3RACTU
C2, C3 & C11	0.1 µF, 50V, 10%, CERAMIC	1206	140-CC502B104K-RC
C4	220 pF, 50V, 5%, NPO	0805	140-CC501N221J-RC
C5	.01 µF, 630V, 10%	1206	81-GRM31BR72 J103KW01

PART	DESCRIPTION	PKG	MOUSER PART#
C6,C7,C9, C12, C13	10 µF, 100V, 20%, SMD	8MM	661-MVH100V100M
C8	6.8 nF, 50V, 10%, CERAMIC	0805	140-CC501B682K-RC
D1	15V ZENER, 1W	SMA	621-SMAZ15-F
D2, D3	DUAL, 75V, 100 mA	SOT23	771-BAV199235
D4	DUAL, 100V, 2X5A	D2PAK	511-STPS10H100CG
Q1, Q2	NMOS, 100V, 35A	TO263	726-IPB26CN10NG
Q3	PNP, 100V, 6A	TO-220	512-TIP42C
IC1	PWM MIC2186	SO16	576-1715-5-ND *(DIGI-KEY)*
IC2	5V REGULATOR, 100 mA	SOT223	522-ZSR500GTA
IC3	PIC12F675 MICRO	SO8	579-PIC12F675-E/SN
L1	22 µH, 11A	RADIAL	580-1422311C
L2	CHOKE, 4.9 µH, 15A	AXIAL	542-5219-RC

■ FIGURE 5



PARTS LIST

PART	DESCRIPTION	PART NUMBER
IC1	LOW POWER 555 TIMER	511-TS555CD
IC2	14 BIT RIPPLE COUNTER	595-CD4020BWE4
IC3	HALL EFFECT SENSOR	785-SS351AT
Q1, 2, 3	NPN TRANSISTOR	863-MMBT2222ALT3G
BZ1	BUZZER	254-PB521-ROX
R1, 2	49.9K, 1/8W, 1206 SIZE	290-49.9K-RC
R3	3.32K, 1/8W, 1206 SIZE	290-3.32K-RC
C1, 2	.01uF, 50V, CERAMIC	140-CC502B104K-RC

■ FIGURE 6

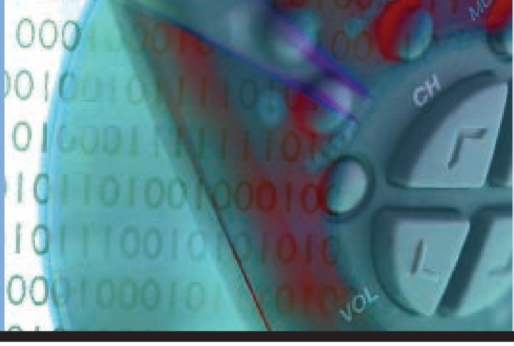
REFRIGERATOR DOOR ALARM

Q My wife has a bad habit of leaving the refrigerator door partially open quite regularly. It is a "French Door" system that is poorly designed so that the left hand door often does not close properly. We usually don't discover this for a few hours. I have tried commercial temperature alarms, but these require a large drop in temperature before they sound.

I was thinking of making an alarm with some magnetic proximity switches and a device that would sound an alarm after just a few minutes (if the left hand door was not completely closed). The delay of a few minutes (maybe adjustable for time) would allow for loading and unloading the refrigerator, without the alarm driving everyone crazy every time the door is opened. Any ideas? Thanks.

— Randy Marconi

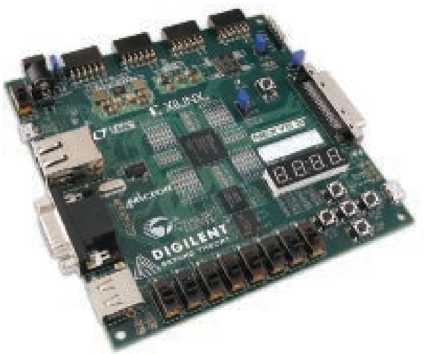
A The circuit in **Figure 6** is low power and can run on three AAA cells, but you would need to replace the batteries twice per month. So, you may want to invest in a low voltage wall wart. The circuit will work on 3V to 15V. IC3 is a unipolar Hall effect sensor; it turns on when the magnetic field is high and turns off when the magnetic field is low. Polarity of the field does not matter. The Parts List calls out a SOT23 packaged Hall effect device (SS351AT) but you may prefer a TO92 package which is SS451A. IC1 is an astable 555 running at 100 Hz, clocking the CD4020 14-bit counter which times out in two minutes. Removing Q1 will double the time. You would put the sensor inside the fridge along with the magnet on the opposite door. You will have to experiment with placement of the magnet. The circuit board and buzzer can be mounted on the outside of the door. Let me know how it works out if you build it. The Parts List calls out surface-mount parts, but all the parts are available as thru hole if you prefer to build it that way. **NV**



NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

NEXYS3 FPGA BOARD



Digilent, Inc., announces the launch of the NEXYS™ 3 board, the latest addition to their family of FPGA-based design kits. Featuring Xilinx's newest Spartan6 FPGA, 48 Mbytes of external memory (including two non-volatile phase-change memories from Micron), USB and Ethernet ports, and other common I/O devices, the Nexys3 board is perfectly suited for academic and industrial design settings. Students and engineers alike can build circuits ranging from simple logic circuits, to CODECs and custom device controllers, to embedded processors without needing any other equipment and for less than the cost of a textbook.

The Nexys3's high speed USB2 port provides board power, high speed FPGA programming, and user-data transfers at rates up to 38 Mbytes/sec. The USB2 port supports Digilent's latest Adept programming software as well as all Xilinx tools, including the free WebPack™, Chipscope™, and EDK™ tools. In addition to FGPA and ROM programming, the Adept system offers automated board tests, a virtual I/O system that extends I/O capabilities using a PC-based interface panel, and simplified user-data transfers. The Adept application, an SDK, and reference materials are freely downloadable from the Digilent website.

A large collection of low cost peripheral boards — including more than 40 Pmods™ and several new Vmods™ — can add additional features to the Nexys3, including A/D and D/A converters, breadboards, motor drivers, displays, etc.

The Nexys3 board is priced at $119 for academic customers and $199 for all others. Volume discounts are available. Most orders ship the day they are received.

For more information, contact:
Digilent, Inc.
Web:**www.digilentinc.com,**

POCKETPICO MEASUREMENT DEVICE

PocketPico Picoammeter — now available from Saelig — is a new, USB-connected, very low current measurement device for DC current from 20 pA to 2 mA. Tiny (2.7" × 1.4" × 3.7") and lightweight (8 oz) the PocketPico Picoammeter

eliminates harmful switching spikes and reads at a rapid 15,000 samples per second. Perfect for lab test setups, PocketPico Picoammeter's high accuracy, fast speed, and single measurement range brings new features previously unavailable in other picoammeters, like small size, single 20 pA - 2 mA measurement ranges at 15,000 Sa/s, high accuracy (±0.5%) and resolution (4 1/2 digits), and protection against voltage transients, etc.

PocketPico Picoammeter is dedicated to the task of measuring very low electrical current in optoelectronics, ion and electron beam monitoring, materials resistance testing, and more. When connected to a PC, PocketPico Picoammeter accurately measures current from 20 pA – 2 mA and displays it using the free Windows-compatible easy-to-use PocketPicoReader software on the PC's display. PocketPico Picoammeter is one of the easiest and most affordable ways of measuring a wide range of extremely low current. Simple to integrate with existing lab tasks, it avoids using bulky equipment simply to take low power measurements, especially for field use. Applications for the PocketPico Picoammeter include: photodiode current and dark current measurement; ion beam monitoring and measurement; SEM and TEM beam current measurement; optical fiber alignment; component, sensor, and device I-V characterization; materials resistance testing; leakage current testing; analog and mixed-signal circuit testing and analysis; teaching labs; product demonstrations; and more.

For more information, contact:
Saelig
Web: **www.saelig.com**

If you have a new product that you would like us to run in our New Products section, please email a short description and a photo of your product to:
newproducts@nutsvolts.com

PORTABLE A/D
WORKSTATION TRAINER

Global Specialties announces the release of a new and improved portable workstation trainer — Model PB-503C.

The new aluminium case provides additional protection, strength, and internal storage space over the previous case. Additionally, the PB-503C has completed an extensive electronics evaluation to improve quality and performance. The new PB-503C are in stock available for sale.

The PB-503C Analog & Digital Design Workstation is a portable robust electronics trainer suitable for all levels of electronics instruction and design. Utilizing the PB-503C, students will learn valuable hands-on breadboarding techniques and build a solid foundation in circuit experimentation, construction, and analysis.

Experienced designers will find the PB-503C an invaluable instrument, providing a reliable platform for the most advanced and demanding design applications.

The PB-503C can be used to construct basic series and parallel circuits to the most complicated multi-stage microcomputer circuits, incorporating the latest in industrial technology. It comes with a removable breadboard plate with additional plates available and a three year warranty on all parts and workmanship.

For more information, contact:
Global Specialties
Web:**www.globalspecialties.com**

SHOWCASE

By David Welburn

BUILD AN LED ANALOG CLOCK

Staring at an old clock on my living room wall one evening, I decided it was time for something new and original to hang up on the wall.

The clock presented here uses LEDs to represent the hands of an analog clock. It's a great mix of analog and digital all in one.

This clock has 24 hand positions giving a 2-3 minute resolution on each one; each hand position has seven LEDs, giving a total of 169 LEDs.

On the outer ring, there are 12 dimly lit LEDs showing where the hour positions are. Digital mode displays the number digits. This is shown once a minute for around 15 seconds. **Figure 1** shows the '10:20' position.

The LEDs are wired into a large matrix formation 12 columns x 15 rows (see **Figure 2**). The rows are connected via current-limiting resistors directly to the PIC16F877 I/O ports C and D. The columns are connected to a 0V rail via ULN2003 transistor arrays, and are controlled by the PIC16F877. The persistence of vision technique is used to light each column one by one for a very brief time, and then repeat the sequence several times a second to give the illusion that all the required LEDs are lit at the same time.

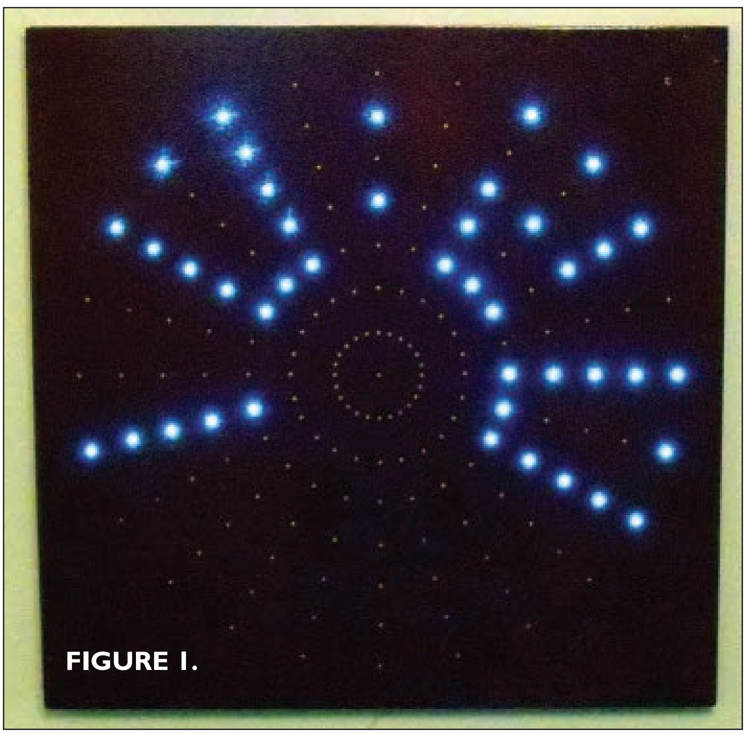


FIGURE 1.

Time To Build

First, build up a frame as shown in **Figure 3**, measuring 610 mm x 610 mm. The front of the clock is 3 mm thick hardboard and the frame is made up with wooden batons 30 mm x 30 mm.

With the aid of a compass and ruler, map out the LED positions for drilling. Draw out the outer circle using a compass to a diameter of 532 mm. Now, draw straight lines from the center pivot point to the outer edges of the wood plotting the 12, 3, 6, and 9 o'clock positions. To find the circumference of the outer circle, you multiply the diameter by pi (3.1416). This gives a circumference of 1,671 mm; divide this by 24 hands = 69.5 mm, so the LEDs on the outer ring are separated by 69.5 mm.

Starting at the 12 o'clock position, measure 69.5 mm to the next LED position and so on. By the time you reach the 3 o'clock position, you should have seven equally spaced marks; continue around until you reach the 6 o'clock position. I found this to be a bit of trial and error as measuring 69.5 mm on a curve is tricky. Use a pencil for marking; that way, you can rub out and retry without getting into too much of a mess.

Take a long ruler and draw lines from the marks on the outer ring across the pivot center mark to the opposite side of the outer ring. This will plot the 6 o'clock to 12 o'clock positions without having to measure.

The next step is to draw the other rings. On the 12 o'clock line, mark positions for the other rings 38 mm apart. Then using the compass, draw in the other rings at the marked positions. This should give seven equally spaced rings (including the outer ring). Use a small punch and press firmly into the wood to make a small dent at each LED position; this will help your drilling to be more precise. Using a 3 mm drill, start drilling the holes for the LEDs. Once drilled, sand away the burrs. Now, it's ready to be painted. I used two coats of gloss which took several hours to dry.

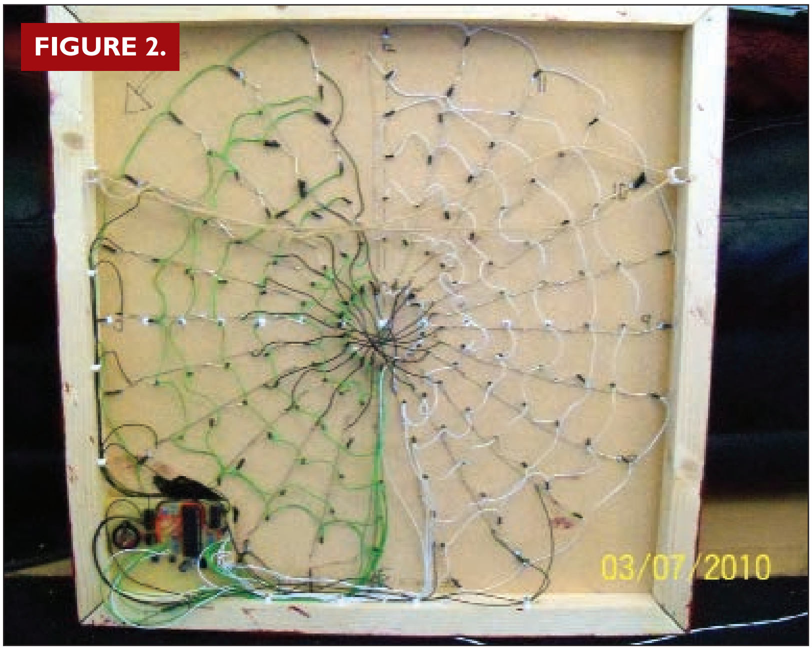


FIGURE 2.

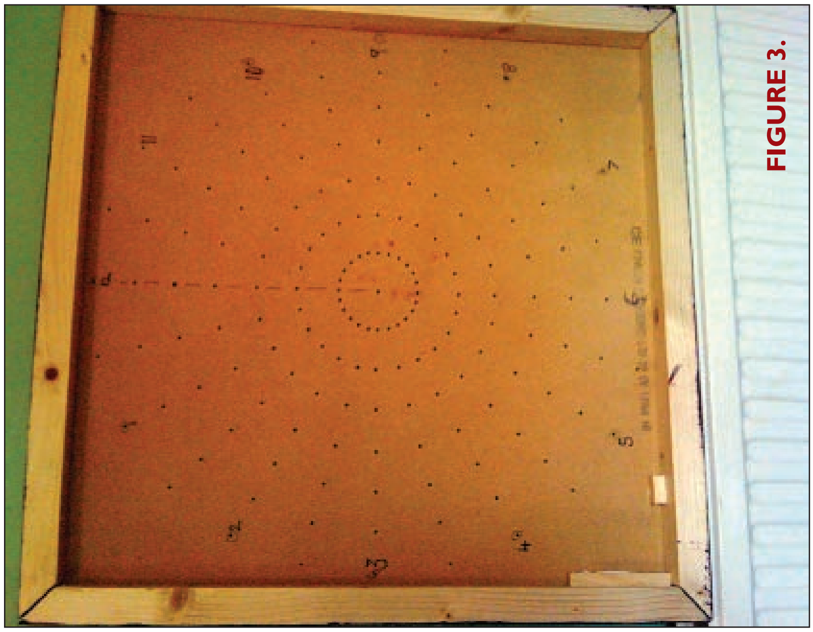


FIGURE 3.

Getting Wired

Now the wiring commences. The thing to remember here is when viewed from the rear, we are working in a counter-clockwise direction. In **Figure 4**, the outer ring is shown wired up. Each ring of LEDs is split in two. Green (LED Matrix A; see **Figure 5**) is hands 1 to 12 and white (LED Matrix B; see **Figure 6**) is hands 13 to 24.

The wiring connects the LED's outer ring anodes together, then connects via current-limiting resistors to the PIC16F877. In this case, the green is going to port C0 and the white to port D0 via resistors. On the second ring (next to the outer ring), the LED anodes are connected in the same way — via resistors to port C1 for green and port D1 for white on the PIC16F877, and so on.

PARTS LIST

QTY	ITEM
1x	16F877 microcontroller 467-1640
1x	DS1307 real time clock 540-2726
2x	ULN2003AN transistor array 436-8451
2x	16-pin DIL socket
1x	8-pin DIL socket
1x	40-pin DIL socket
1x	20 MHz XTAL 672-0306
1x	32.768 kHz 12.5 pF watch XTAL 547-6979
1x	20 mm coin battery holder 430-675
1x	3V CR2032 coin battery
3x	100 nF ceramic capacitor
2x	22 pF ceramic capacitor
1x	10 µF 35V electrolytic capacitor
3x	4.7K 0.6 watt resistor
2x	10K 0.6 watt resistor
15x	0.6 watt resistors for LED current limiting, value dependent on LEDs used; my build uses 43 ohm resistors
2x	Push to make button switches
169x	LED 3 mm high MCD rating diffused; my build uses 700 MCD 3.6V diffused white LEDs
1x	VeroBoard to solder components to
	Several meters of 7/0.2 mm wire (white, green, and black)

Some of the parts have RS part numbers available at **www.rswww.com**. Other parts come from various sellers on eBay.Check out **www.superbrightleds.com** or **www.ledtronics.com** for LEDs.

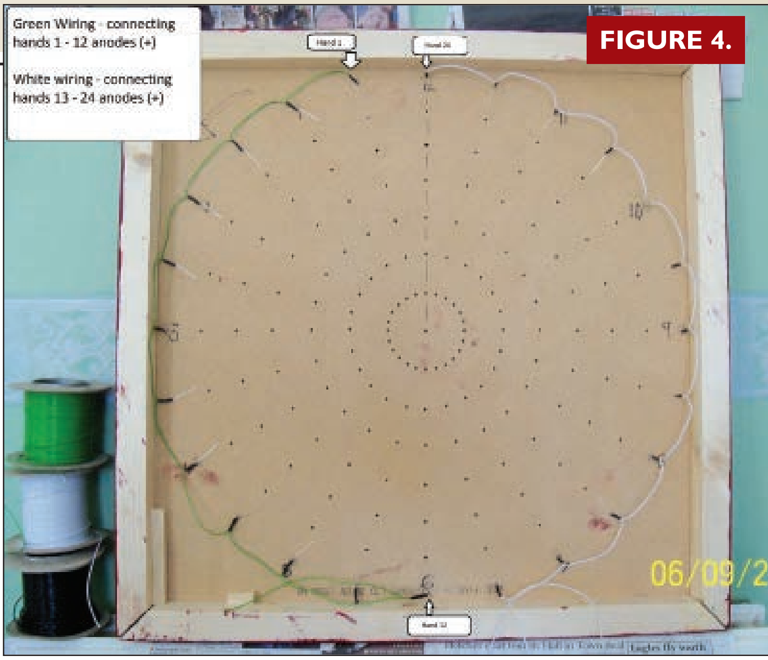


FIGURE 4.

The LED cathodes are connected together for each hand position, so all seven LED cathodes for hand 1 are connected together. This then connects to N1 on the clock circuit diagram. On hand 2, all seven LED cathodes are connected together, then are connected to N2 on the clock circuit diagram. There are 12 columns in the LED matrix which are labelled N1 to N12, so hand 1 and hand 13 connect to N1, hand 2 and hand 14 connect to N2, and so on (see **Figure 7**).

What Current-Limiting Resistor Should I Use?

The PIC16F877 has a maximum current rating of 200 mA over port C and D combined. Each I/O pin is 25 mA. The maximum number of LEDs in one column is 15 (N12 which contains the pivot LED). This means we can supply each row of LEDs with 13 mA to stay within the limits of the microcontroller.

The other thing to consider is the voltage drop over the collector/emitter junction on the ULN2003 transistor array. The datasheet indicates 0.9V-1.1V depending on the load; this needs to be added to the LED's voltage drop to calculate the current. The LEDs I used are 3.6V added to the 0.9V transistor array drop. This leaves 0.5V dropped across the resistor. In this case, I used 43 ohm resistors to give 11 mA current flow on each row (see **Figure 8**).

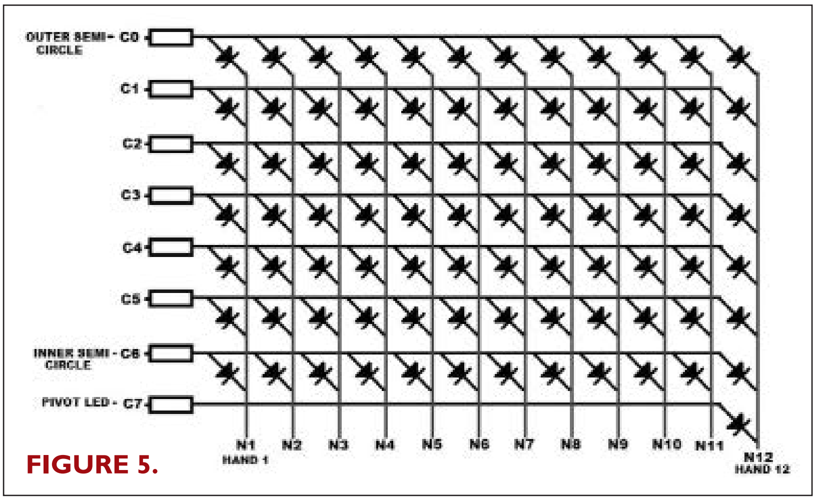


FIGURE 5.

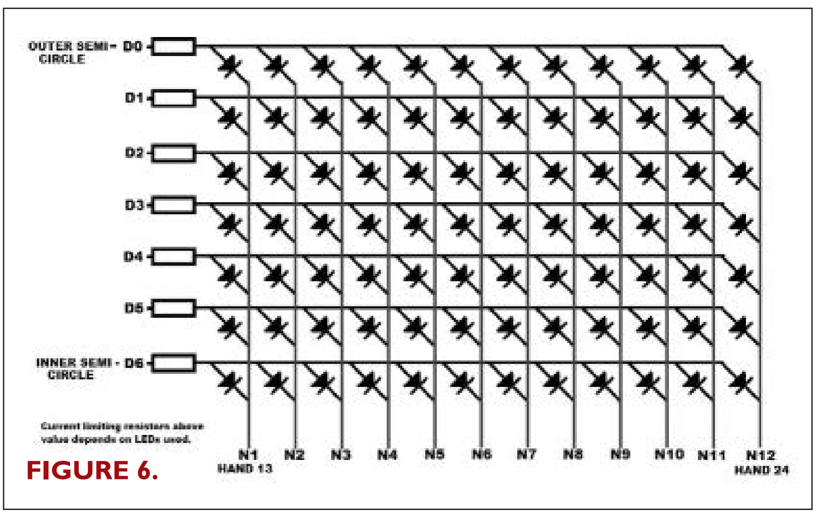


FIGURE 6.

What LEDs Should I Use?

I used 700 mcd diffused 3 mm white LEDs that I found on eBay. It's important to use LEDs with a high mcd rating to maintain the brightness since the LEDs are multiplexed and only really lit a fraction of the time.

The Power Supply

The PIC16F877 and DS1307 both need a stable regulated power supply source of 5V DC. This is best done with a 7805 voltage regulator with a couple of capacitors. In addition to these, I put 100 nF capacitors across the supply pins (VDD and VSS) on the PIC16F877 and DS1307. This removes spikes from the supply that could upset the operation of the microcontroller.

The DS1307

This is a dedicated real time clock (RTC) which keeps the time, date, and day of week. It can operate in 12 or 24 hr clock mode and has its own battery backup (a CR2032 3V watch battery) which will keep the RTC running during power failures. This has its own oscillator which uses a 32.768 kHz quartz crystal (see **Figure 9**).

The RTC and PIC16F877 use an I^2C bus to communicate with each other. The I^2C bus has 4.7K pull-up resistors for the clock and data lines.

PICBASIC PRO from **http://melabs.com** has I2CWrite and I2CRead commands to handle the communication protocol. The data that the DS1307 uses is in BCD format which is converted in the program to decimal.

Please see the datasheet for more information and guidance on the crystal layout on your particular board.

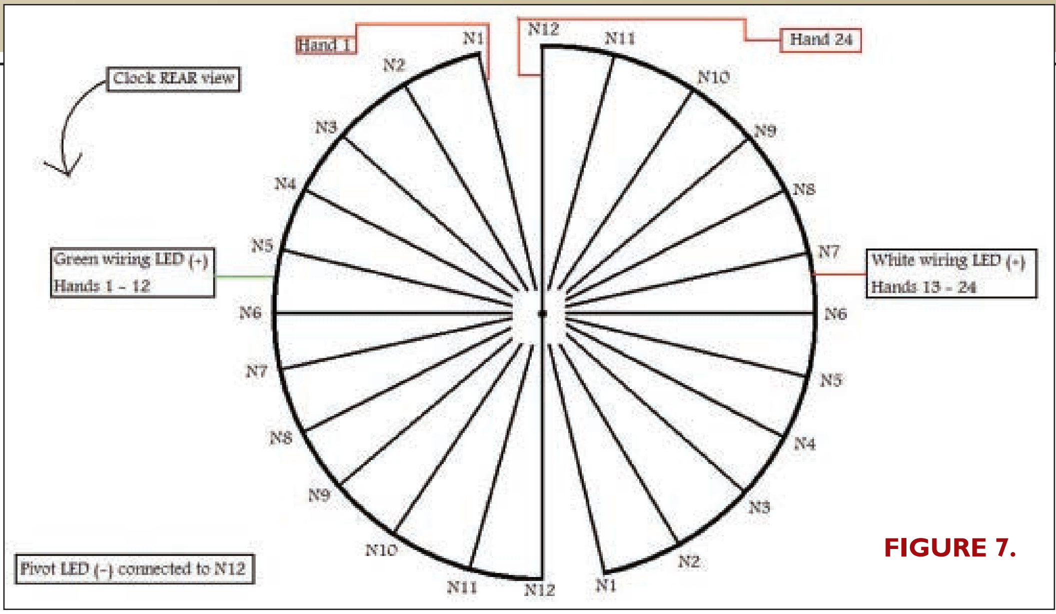


FIGURE 7.

The PIC16F877

This is the CPU of the project. It requires a HEX file to be flashed into the program memory (see **Figure 10**).

This microcontroller was selected for the number of I/O pins it has; almost all are used. It has 8K of Flash memory so it can run fairly big programs. You'll have plenty of room to add some graphic displays.

The ULN2003

These are transistor arrays that can switch higher current since the microcontroller can only sink/source 25 mA max per I/O pin. The transistor arrays can switch up to 500 mA, so it's perfect for switching the LED columns on and off. It can be directly driven from the microcontroller with 5V (see **Figure 11**).

Applying 5V to the input side (pins 1-7) virtually grounds the output side pins (10-16). When 0V is applied to the input side, the output side goes to high impedance. There is, however, a small voltage drop between the output and ground. See the datasheet for more information.

Going Soft

This clock keeps the display lit by scanning each of the column's N1 to N12 for 500 µs. To keep the scanning consistent, I'm using a timer interrupt. This utilizes a timer (TMR0) which counts up one every four clock cycles. So for a 20 MHz clock, this will increment every 200 ns; the timer will count to 255 before resetting to zero. Upon

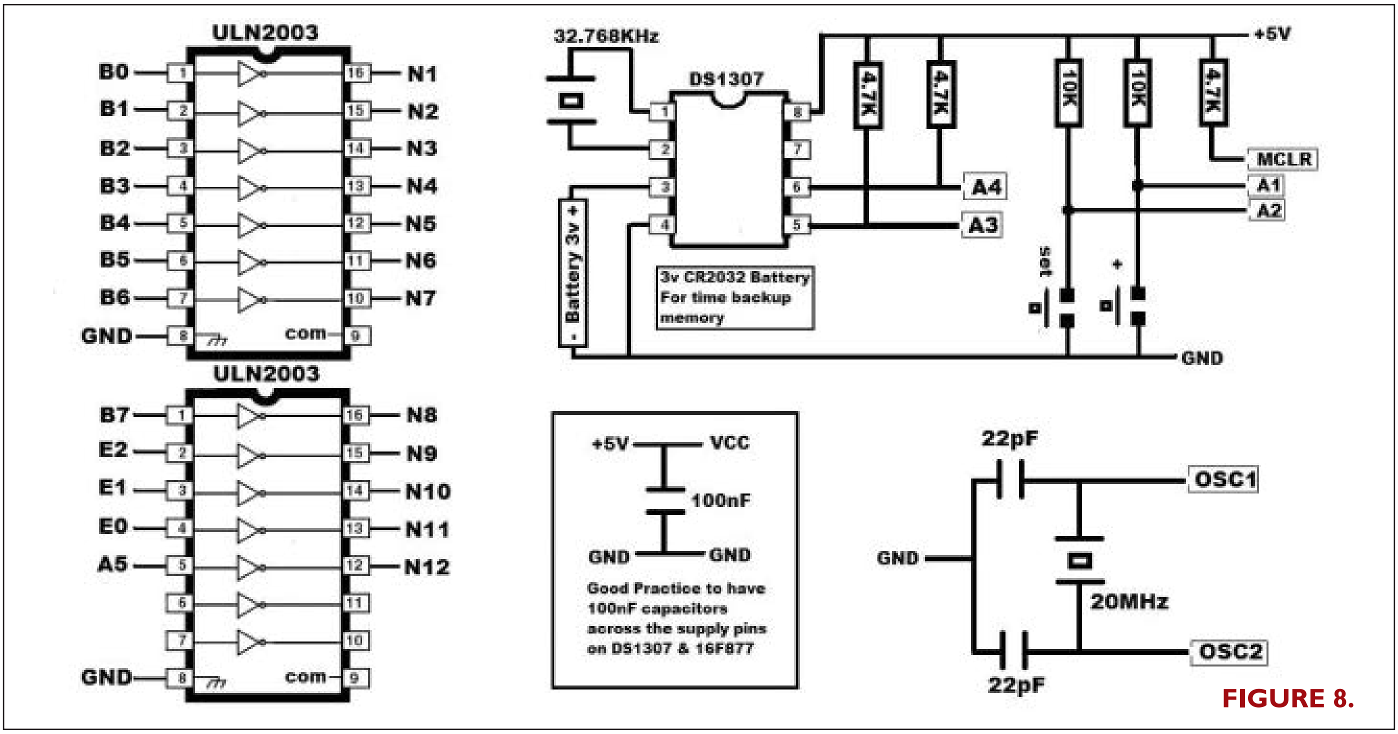


FIGURE 8.

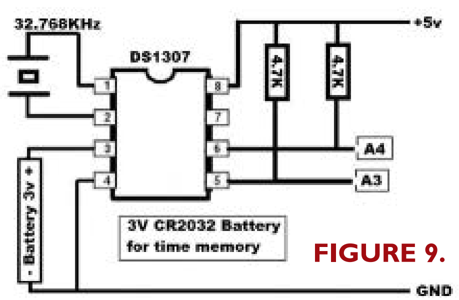


FIGURE 9.

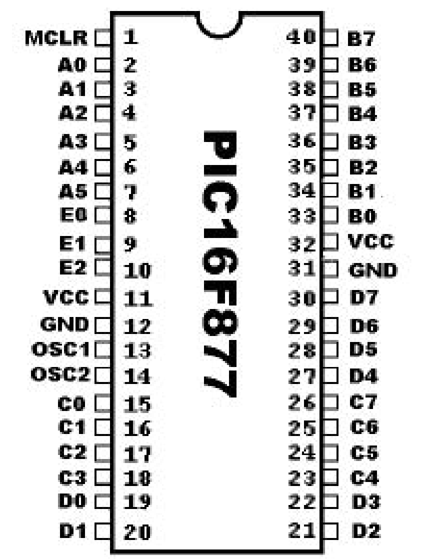


FIGURE 10.

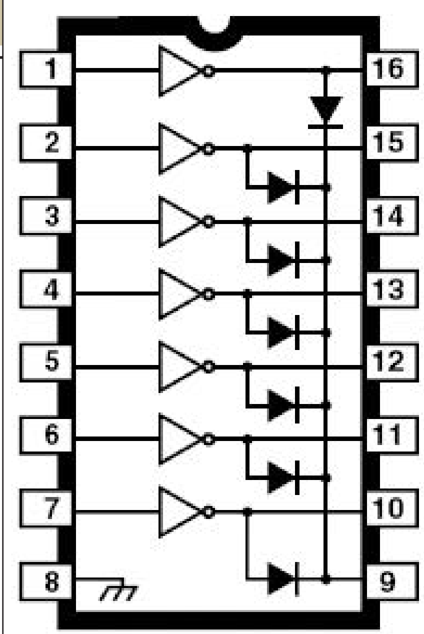


FIGURE 11.

resetting, an interrupt is generated; this means an interrupt occurs every 50 µs which is a bit too frequent. Therefore, a prescaler is used. The prescaler is set to 1:128. This effectively means that the timer has to reset 128 times before an interrupt is generated, giving an interrupt every 6.5 ms. Each time an interrupt is generated, the program jumps to an interrupt service routine which (in this case) is the 'display' section which keeps the display lit and also periodically requests time information from the RTC. When the interrupt service routine has completed, the program then jumps back to where it left off from.

The main part of the program — the 'loop' section which handles the time setting routine — decides what is going to be displayed by updating the variable arrays for LED information (row A), monitors the buttons for pushes, and generates the hands, digits, and patterns.

The Control Board

As you can see in **Figure 12**, I have used a piece of Vero board to build the controlling circuit board. I don't have a layout available because I tend to build my projects unplanned. Normally, I start with the microcontroller then place other components around it. As you can see, my voltage regulator just under the microcontroller is a L7805CV which barely gets warm, so I didn't use a heatsink for it. All ICs are in DIL sockets so they can be replaced easily. I have also fitted my two buttons to this board; you may decide to use better switches and fix them to the front of the clock.

Something to be mindful of is the I²C clock and data lines between the microcontroller and the RTC. I always try to keep the jumper wires short. I have seen on forums that this can be problematic if the jumper wires are long or different value pull-up resistors are used. I have never experienced these issues, but I try to keep the two jumpers under 5 cm and use 4.7K pull-up resistors.

Time Setting

The following will only work while the time is displayed and not during a pattern demo:

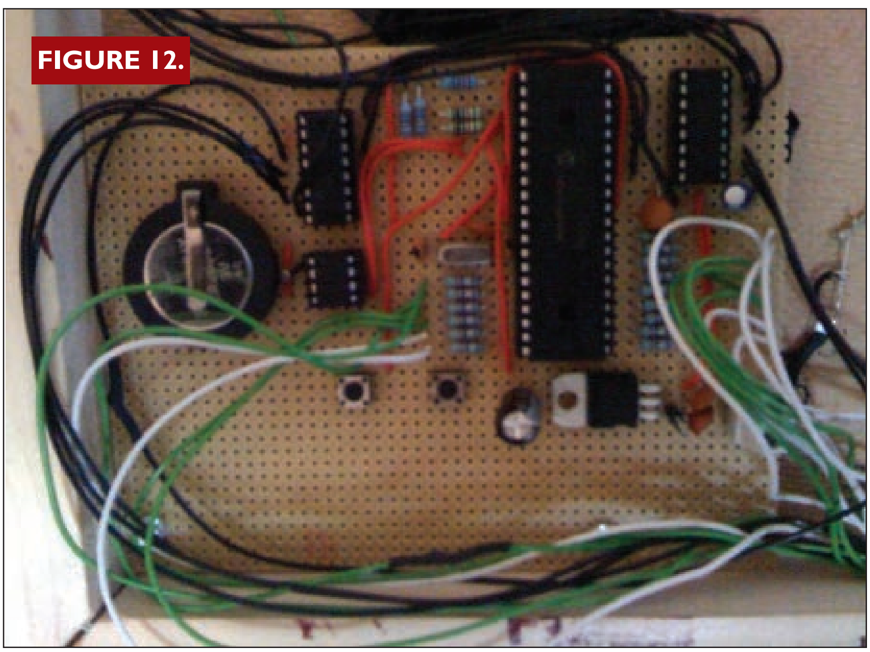


FIGURE 12.

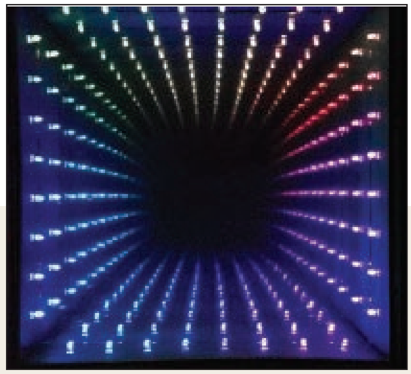
Press the 'Set' key and release.
The time will be displayed in digital format.
Press the '+' key.
This will increment minutes by one for each press.
Press the 'Set' key and release.
Press the '+' key.
This will increment hours by one for each button press (12 hour mode only).
Press the 'Set' key.
Now the time is set.
Press the '+' key.

While the time is displayed, it will switch between analog only mode and digital/analog mode.

Hope you enjoy this timely project. For further information, contact me at djwprojects@live.co.uk. **NV**

By Craig A. Lindley

BUILD THE INFINITY PORTAL



Okay, I'll admit it. I'm a sucker for colored flashing lights especially if controlled by a microprocessor (uP). So recently, when I came upon a really good deal on RGB LEDs on eBay and another really good deal on the new Arduino Uno processor board, I knew I had to build something but I wasn't sure quite what. As luck would have it, as I was clearing out my parents home I found an *Infinity Mirror* I had purchased for my father in the 1980's. All of a sudden I knew what I should build: an updated version of an infinity mirror with multi-color LED light sources controlled by an Arduino. Since this was to be a major update to the infinity mirror concept, I decided to call my creation an *Infinity Portal*.

For those not familiar with what an infinity mirror is, it is usually a smallish enclosure similar to a shadow box that one looks into which has light sources (typically white) around the interior perimeter. The single layer of lights looks to extend far beyond the depth of the enclosure with the lights seeming to repeat 20 to 30 times in concentric patterns. So, in other words, the reflected lights seem to go on for six to eight feet inside of an enclosure that is only inches deep.

I had to determine just how the infinity mirror effect was accomplished if I was going to build one. It turns out the effect is easy to reproduce. Regardless of the size or shape of an infinity mirror, they are all based on the same concept and construction. Internally, an infinity mirror consists of one fully reflective mirror, one partially reflective/partially transparent mirror, and the light sources sandwiched between them. When a light ray from a light source strikes one of the mirrors, the ray is reflected in accordance with the Law of Reflection: The angle of incidence equals the angle of reflection. The reflected image then reflects off the second mirror back toward the first, and so on to infinity, assuming the mirrors are parallel. The details of the pattern are altered by distortions and reflective losses in the semi-transparent mirror, small variations in the angle of the mirrors, and by the view point from which the pattern is observed.

So, to build an Infinity Portal requires an enclosure of some type, a fully reflective mirror mounted in the rear of the enclosure, a series of uP controlled RGB LED light sources, and a partially reflective mirror through which the observer looks. Hardware aside, we also require some clever software to drive the LEDs in bright, creative, and colorful ways. **Figures 1** and **2** show the Infinity Portal I built that will be discussed in this article. Commercial examples of infinity mirrors can be found at **http://lightenergystudio.com/infinitymain.html** among other places on the Internet. As with any one off project like this, I experienced a few issues while building my portal which I will describe at the appropriate points in the article. I will describe how the hardware and software for the Infinity Portal work but I will leave the packaging to you and your imagination. I made my enclosure out of half inch MDF (medium-density fiberboard) which I painted black with a facade made of walnut.

FIGURE 1. My enclosure measures 17" x 17" x 4.75" and is constructed from MDF and walnut wood.

The Portal Hardware

There are 32 RGB LEDs used in this design with eight mounted on each of the four sides of the enclosure. When you consider that each RGB LED is really three separate LEDs in a single package, you soon realize that we are really talking about controlling 96 LEDs in total. Individual control over each LED amounts to a lot of drivers that the uP has to control. Luckily, Texas Instruments produces a chip for just such applications called the TLC5940 (see **Resources** for a pointer to a datasheet). There is a lot to these chips but for

www.nutsvolts.com/index.php?/magazine/article/august2011_Lindley
Craig can be contacted at calhjh@gmail.com.

FIGURE 2. PLEASE NOTE: The USB programming cable and power cable protrude from the rear of the portal. I used Velcro® to secure the USB cable when not in use.

our application you can think of them as made up of a serial shift register driving 16 PWM (Pulse Width Modulated) constant current drivers. These chips are made to be cascaded together to control lots and lots of LEDs, but even with 16 channels of control per chip it would require six TLC5940s to control 96 LEDs in a typical design. Fortunately, I used multiplexing in this design to control the LEDs which reduces our TLC5940 chip count to just two devices. Each LED channel supports 12 bits of PWM brightness data which range in value from 0 (off) to 4095 (full brightness). **NOTE:** TLC5940s are constant current sinks so they must be used with common anode RGB LEDs. One issue I had while building my portal was the mass of wiring needed to connect all 96 LED channels to the controller hardware. I used small gauge solid wire to connect the LEDs but even so, the wire bundle was bigger than I had anticipated which made the space in my enclosure cramped to say the least. Keep this fact in mind when deciding how to package your portal. The insides of my portal are shown in **Figure 3**.

LED multiplexing requires both hardware and software support. In terms of hardware, I divided the 32 RGB LEDs into four rows of eight. Each of the rows have a common buss to which the anode lead of each LED in the row is connected. See the schematic in **Figure 4** for clarification.

A PNP power Darlington transistor — controlled via a digital output pin from the Arduino — controls the current source for each row. When the digital output is low, current flows to the LEDs in a given row. When the digital output is high, all LEDs in the row are turned off. While multiplexing, only one row of LEDs should be on at a time. Serial data for controlling each LED channel is fed to the two cascaded TLC5940 chips via the SPI (Serial Peripheral Interface) provided by the Arduino. Data is shifted out in big endian format with the most significant bits going first. The three color LEDs which comprise an RGB LED are connected to sequential TLC5940 channels. The blue LED channel is the least significant while the red LED channel is the most significant. As designed, only 12 of the 16 channels available on each TLC5940 are being used. You, however, could use all of the LED channels.

The current available for each channel's constant current sink is set by a resistor connected to pin 20 of each TLC5940 chip. In my design, I used 10K ohm, 20 turn trimmers for setting the current. I have the current set to around 20 mA and the LEDs are plenty bright enough even with multiplexing. As set, all of the components in the Infinity Portal run cool to the touch which is important for an enclosure without ventilation.

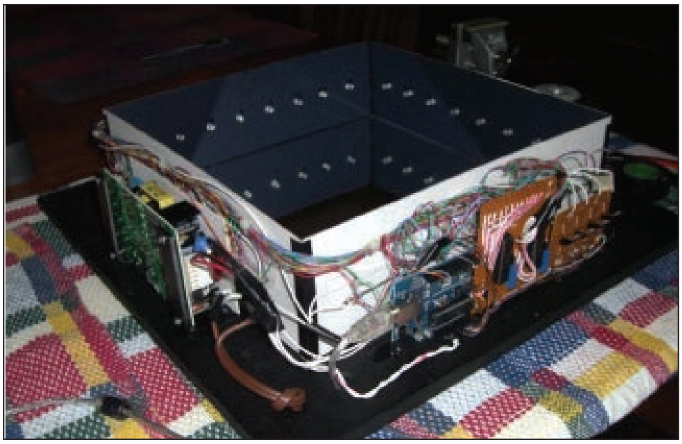
I had a few requirements for powering the portal. First, I didn't want to use a wall wart type of supply because I didn't want any electricity consumed when the portal was turned off. Second, I wanted to power the portal with just a single, 5V supply. Finally, I thought it better to buy a regulated power supply instead of building one myself. In looking around, I found a surplus 5V, 12 amp power supply with over and under voltage protection and current limiting that was small (3" x 5" x 1") and cheap ($5.75 from **allelectronics.com**). A 12 amp power supply was severe overkill as the portal will run from the power supplied by the USB cable from my MacBook Pro laptop. Any power supply capable of a couple of amps at 5V will easily power the

Parts List

Designation	Value	Source/Notes
C1,C3	.1 µF capacitors	Anywhere
C2,C4	10 µF capacitors	Anywhere
D0-D31	RGB LEDs common anode	**ebay.com**
Q1,Q2,Q3,Q4	TIP126 Darlington transistors	**allelectronics.com**
R1,R3,R5,R7	47K ohm 5% resistors	Anywhere
R2,R4,R6,R8	4.7K ohm 5% resistors	Anywhere
R9,R10	10K ohm 10 turn trimmers	RadioShack
U1, U2	TLC5940 LED drivers	**ebay.com**
Power Supply	5 volt 2 amp minimum power supply	**allelectronics.com**
Processor Board	Arduino Uno	**ebay.com**
N.A.	Perf board	RadioShack
N.A.	Hookup wire	Used phone wire

Check out **www.superbrightleds.com** or **www.ledtronics.com** for LEDs.

FIGURE 3. All components were mounted to the back of the enclosure with the exception of the power switch and the partially reflective mirror which are mounted on the front. The power supply is shown on the left. The Arduino Uno (blue PCB) is near the front and the custom circuitry is built onto two brownish perf boards on right. The 32 RGB LEDs are mounted on black matt boards which frame the rear 12" x 12" mirror. The USB programming cable is attached to the Arduino and passes out the back of the enclosure through a hole.



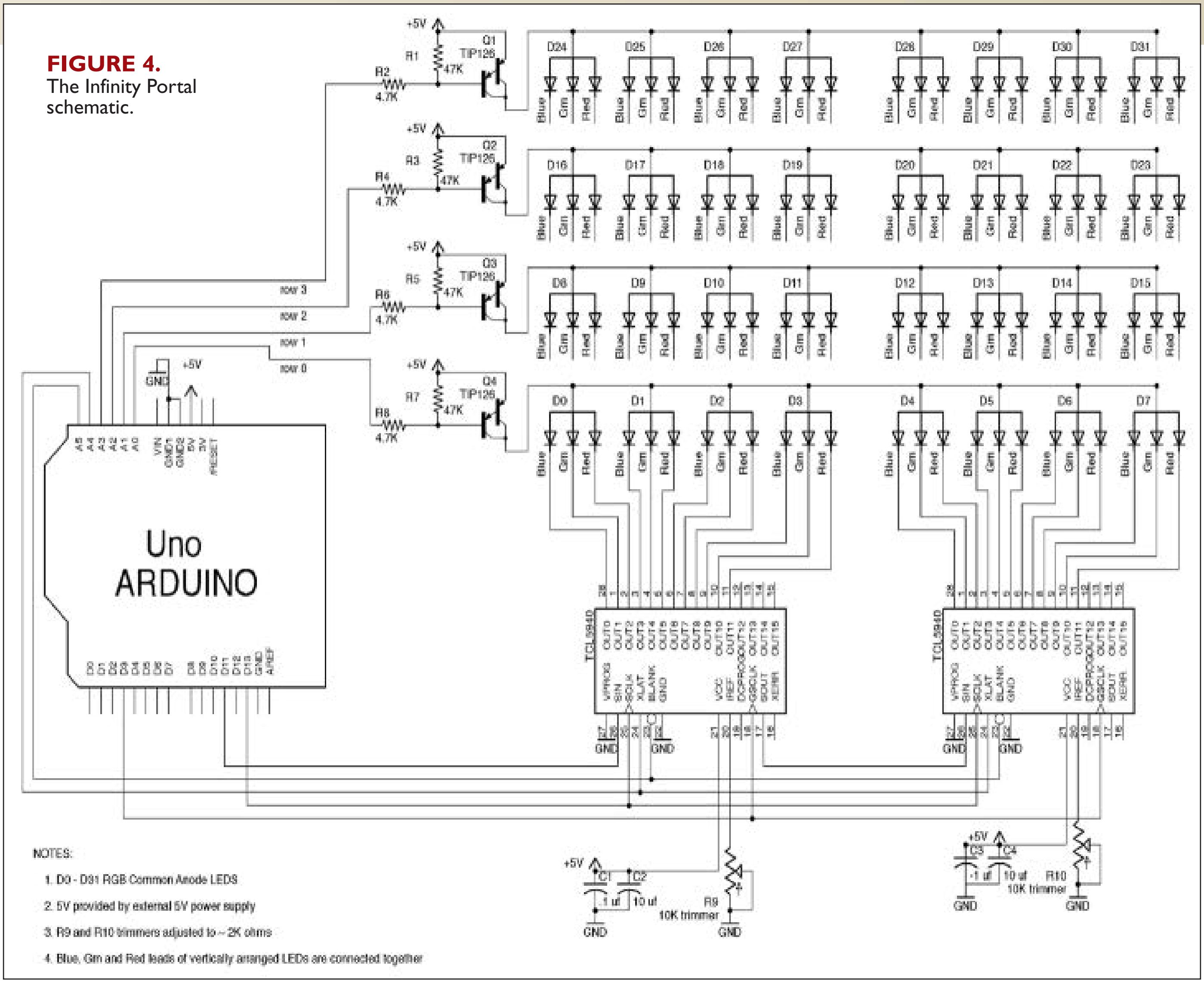


FIGURE 4. The Infinity Portal schematic.

portal with plenty of reserve power to spare.

The Portal Software

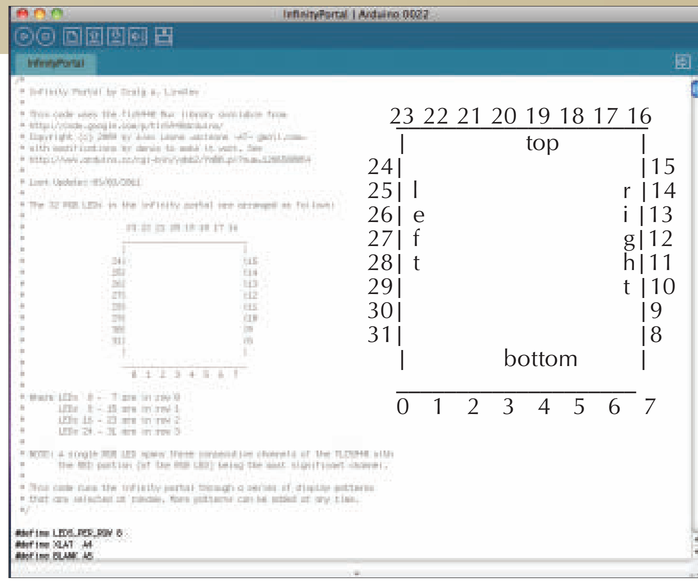
It turns out using an Arduino processor to control TLC5940 chips for control of LEDs and stepper motors is a popular pasttime. So much so that an Arduino library exists for TLC5940 chips. This library (see **Resources**) was provided to the Arduino community by Alex Leone and is controlled in accordance with the GNU General Public License. By using this library, you needn't understand all of the details of the Arduino/TLC5940 interaction.

The TLC5940 library exists in two forms. The first is for controlling TLC5940 chips serially chained together and the second one is for controlling TLC5940s used in multiplexing schemes. Unfortunately, the multiplexing version of the library doesn't work as provided, but can be made to work with a few small changes to the hardware design and the library code. See **http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1286580054** for the details if you are interested. The design presented here implements these changes allowing multiplexing to work. For those of you who haven't worked with the Arduino before, there is a free set of development tools available (see **Resources**). **Figure 5** shows the Integrated Development Environment (or IDE) which comes with the tools package. It is both simple and useful for Arduino code development. The Arduino programming language is based on C/C++ so it should be familiar to many. There are two entry points to all Arduino programs: setup() and loop(). The setup() function is called first and it is within this function where you do all hardware initialization. Subsequently, the loop() function is called which contains all of the operational code for your program.

Once you have downloaded and installed the Arduino tools on your development machine, you connect a USB cable to your Arduino processor board. After writing and verifying (compiling and linking) your code, you click a button in the IDE to transfer the executable code to your target. Print statements to the pseudo serial port in the code provide a means of debugging your program.

The file *InfinityPortal.pde* (available electronically; see **Resources**) is the sum total of all of the code I developed for the Infinity Portal. This file contains the fixes to the TLC5940 library, code for configuring the library for multiplexing, and the code for each of the display patterns (of which there are currently 24) used in the portal. In its current form, the

FIGURE 5. Displaying the *InfinityPortal.pde* code.



Infinity Portal's code selects display patterns randomly from the time the portal is turned on until it is turned off.

Before showing how display patterns are developed, it is interesting to note there are two asynchronous processes running simultaneously within the code. These are referred to as the foreground and the background process. Both these processes share a single data buffer; the foreground process puts data into the buffer and the background process shifts the data out of the buffer to the TLC5940 chips.

The background process runs at interrupt time when the PWM timer times out indicating the full PWM cycle for controlling LED brightness has completed. A counter called *shiftRow* is consulted at interrupt time to determine which row of the multiplexed LED data is to be shifted out to the TLC5940s, and then which of the four row selects is pulled low enabling that row of LEDs. Afterwards, *shiftRow* is incremented so that a different row will be selected next interrupt time. This background process repeats continuously without regard for the actual data that is being sent to the TLC5940s for each row. The foreground process is where display patterns run. Code implementing the display pattern writes data into the shared data buffer where the background process picks it up and sends it out. It is important to understand/remember that once data is placed in the data buffer, it stays there until overwritten. If you want to light an LED and then turn it off, you have to first set the data and then zero the data to turn it back off. There is a lot of code in *InfinityPortal.pde* but you can ignore most of it. What you must understand is that the 32 LEDs are addressed by number (0 .. 31) (as shown in the inset in Figure 5). Functions from the code that are used for pattern development are shown in **Table 1**. As an example, the following code turns all LEDs on in an alternating red/green pattern:

```
TlcMux_clear();        // Turn off all LEDs
```

Function Name	Operation
TlcMux_clear();	Clears the data buffer causing all LEDs to turn off.
delay(milliseconds);	Delays program execution for specified number of milliseconds.
setLEDOff(ledNum)	Turns the specified LED off.
setLEDRGB(ledNum, red, green, blue)	Set ledNum LED to the specified red, green, and blue values. Valid values 0 .. 4095.
void setLEDRGB(ledNum, struct RGB rgb)	Same as above except a structure contains the RGB value to set.

Arduino information can be found at **http://arduino.cc**.

The free development environment can be downloaded from **http://arduino.cc/en/Main/Software**

A TLC5940 datasheet is available at **http://www.ti.com/lit/gpn/tlc5940**.

The Arduino Tlc5940 and Tlc5940Mux libraries can be found at **http://code.google.com/p/tlc5940arduino/**

The Infinity Portal code (*InfinityPortal.pde*) is available from the *Nuts & Volts* website at **http://nutsvolts.com**

Reflective window film (Gila #PRS361) was purchased at Lowe's, as were the 12"x12" rear mirror and the window glass for the partially reflective mirror.

```
for (int i = 0; i < 32; i++) {
                         // Loop through all 32 LEDs
     if (i % 2 == 0) {
                 // Is this an even numbered LED ?
       setLEDRGB(i, 4095, 0, 0);
        // Turn ith LED to red, full brightness
     }
     else  {
       setLEDRGB(i, 0, 4095, 0);
        // Turn ith LED to green, full brightness
     }
}
```

As mentioned, the Infinity Portal has 24 display patterns built in. As the simple code above illustrates, it is really easy to add new patterns or change existing ones.

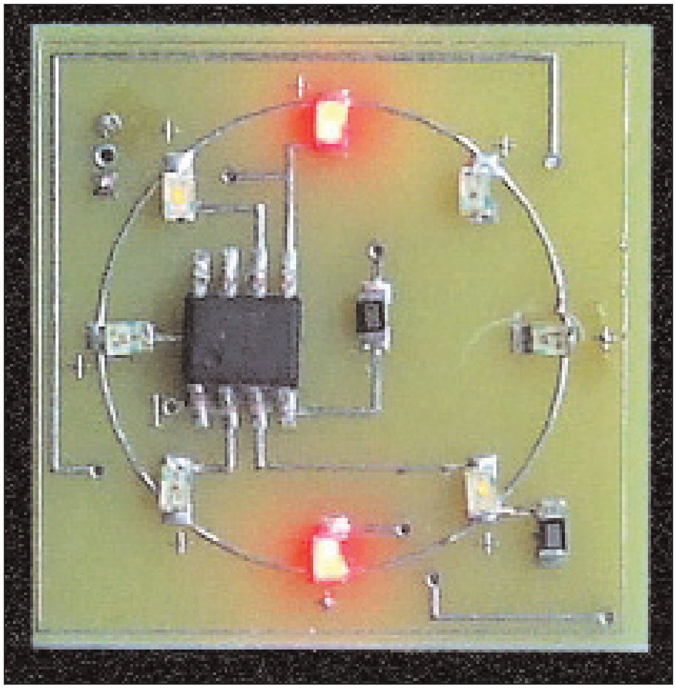
Portal Packaging

There is no end of possibilities for packaging an Infinity Portal. I've seen infinity mirrors packaged in a square box (similar to what I did), in a rectangular box, a triangular box, a round box, and even a star-shaped box. The important things are to have the semi-transparent mirror facing the viewer and to have the mirrors, mounted parallel to each other. The light sources (RGB LEDs) are mounted around the perimeter of whatever enclosure you use, spaced half way between the parallel mirrors. If the lights are not equally spaced between the mirrors the reflections you see in the portal will be unequally spaced. This may be an effect you want to utilize in your packaging. Since I have access to woodworking tools I made my portal out of MDF with walnut trim. Once finished, the MDF box was painted black as this color scheme matches the furniture in our living room where the portal resides. Finally, a word must be said about the partially reflective mirror used in the portal. Three possibilities were investigated: "one way mirror" glass, partially reflective acrylic, and the use of reflective window film over plain glass.

For cost considerations, I decided to use mirrored window film available at most hardware stores over plain glass. The film must be applied to the glass using soapy water so that it lays out flat with a minimum number of bubbles. This is easy to do after a little practice. I hope you'll enjoy this project for a long time ... to infinity and beyond. **NV**

By Ron Newton

BUILD A MINIATURE COLOR ORGAN



Being a jazz musician, I'm always looking for a way to show off the band. Here is a miniature light organ that displays when it picks up a sound. It is only one inch square, has eight LEDs, and runs off of a 12.5 mm 1025 Lithium battery. Most LED jewelry runs off of two hearing aid batteries which are short lived. The 1025 provides 30 milliamp hours and should last a long time. When not in use, the microprocessor puts itself to sleep and wakes up when the sub-miniature microphone detects a pulse. When attached to the bass drum or snare drum, each beat will cause it to flash. The flashing colored LEDs even appear to spin one way and then reverse.

This light organ can be used by bands in marching parades, football games, or even be placed on speakers. It is best for flashing at night, or it can be used on the bandstand to draw attention to the drummer.

The cost of the project is $3.50 per device plus the board. The whole project should take less than one hour to build. Board sets are available along with pre-programmed chips from the *Nuts & Volts* Webstore, or a complete kit is also available. All of the components are surface mount except for the microphone. However, I should point out that this is a great first project if you are interested in surface-mount, as I have used larger components; 805 components are manageable, whereas 402s and 603s are hard to solder without a lot of practice or even a reflow oven. Please watch the surface-mount demo video in the *NV* Webstore on the product detail page. I used four different colored LEDs: red, green, yellow, and blue. You can substitute white for any of the colors. Or, perhaps at Christmas you could just use red and green. For the 4th of July, use red, white, and blue. Halloween, yellow and orange. Use your imagination. The program allows you to program up to eight separate light programs. The one presented here has two.

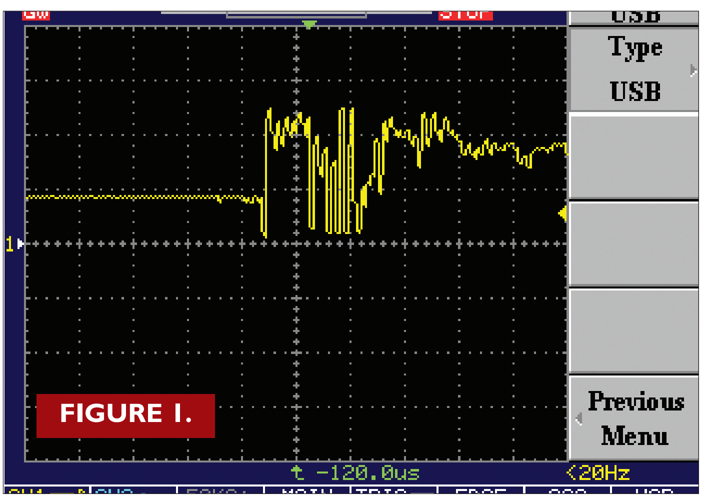


FIGURE 1.

CIRCUITRY

A Projects Unlimited sub-miniature microphone is

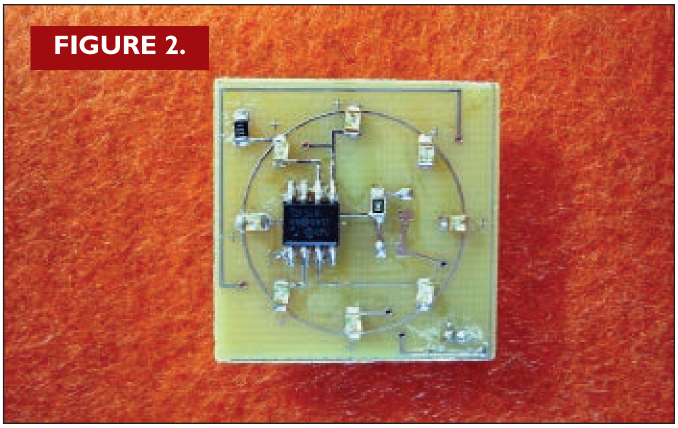


FIGURE 2.

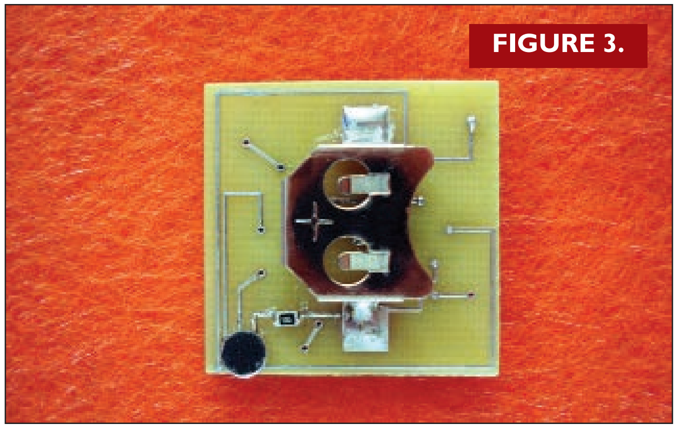


FIGURE 3.

used to detect the sounds and it is biased with a 15K 802 surface-mount resistor. When the microphone picks up a sound, this causes a change of voltage; this is connected to a PIC12F508 microprocessor. GP3 is used as an input pin and is TTL (Transistor Transistor Logic). This pin trips at approximately a positive .85 volts. This means that a voltage pulse crossing the .85 volts' threshold either way will cause the LEDs to flash.

At first, I had a difficult time getting the microphone to trip the microprocessor. Most of the time when using a microphone, you use a 1.0 µF or 10 µF capacitor. After trying several different circuits using resistor and capacitor networks, I found if I directly wired the mic into the micro and biased it with a 15K resistor it worked fined. The microprocessor is normally asleep, and a change of voltage on GPIO 3 pin 4 causes it to wake up and start its program. The output from the microphone to the PIC is shown in **Figure 1**.

The eight colored LED anodes are connected to a common trace. This trace is connected to a 100 ohm resistor going to three volts positive. This resistor prevents excessive current from being used by the LEDs. The LEDs turn on in pairs. To view the program, go to this article's download file and find the drum file. There are only about 100 actual lines of code, so it is a short program. Although the lights look random, there are seven different programs to make them flash in different ways.

The program starts at ORG 0 (the first line of code). The commands after the START label tell the micro which pins are IN pins and which pins are OUT pins. The option command allows GP3 to wake up the microprocessor from sleep. Four is loaded in the REPEAT. Changing this number will increase or decrease the number of times the LEDs blink or spin.

The 12F508 is unique when awakened; it jumps back to location zero and starts the program over. Most chips when awakened resume from where they were when they went to sleep. The trick I used to call a different light program when awakened is to use a flag register and advance the flags each time a program is run. A BTFSC (Bit Test File & Skip if Clear) is used to check the flags. This is located in the label call Flag_routine. The program continues until it finds a bit that is set, and jumps to that program. Once it arrives at the light program, clearing the bit BCF (Bit Clear File) will cause the LEDs to turn on. A delay timer is used to set how long the LEDs stay on. The repeat will determine the number of times they flash with each beat.

You will note on the board drawing that traces 3 and 6 are tied together on the 508. This was done to free up board space. One of the advantages in using microprocessors is that you can short some pins together and make one of the pins an input, and eliminate routing of an extra trace.

One of my favorite chips is the 12F508 and it was ideal for this project. The 12F508 is a great micro for cutting your teeth on. It first came out as a 12C508; however, you could only program it once. If you programmed it wrong, you threw it away. For experimentation, you had to buy a chip with a quartz window and erase it if you were going to reprogram it. These were not inexpensive (cost about $15 ea) and

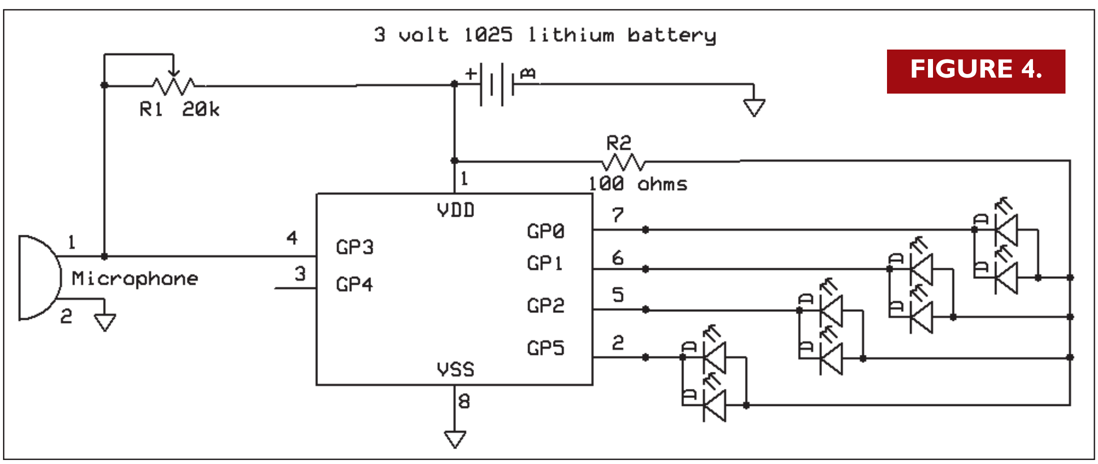


FIGURE 4.

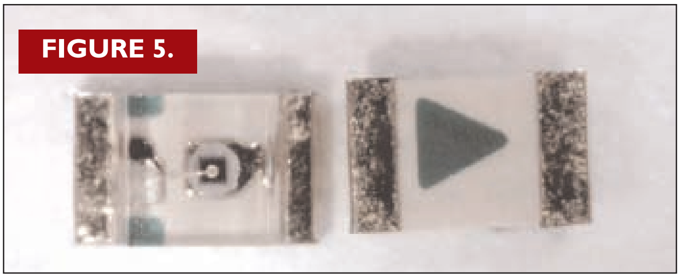


FIGURE 5.

PARTS LIST

ITEM	DESCRIPTION	QTY	SOURCE
Battery Holder	12 mm surface-mount	1 ea	
LED Red	805	2 ea	
LED Yellow	805	2 ea	
LED Green	805	2 ea	
LED Blue	805	2 ea	
IC1	PIC12F508	1 ea	Microchip
Microphone	AOM-6742P-2-R	1 ea	PUI Audio
R1	3 mm 20K potentiometer	1 ea	
R2	100 ohm 805	1 ea	

didn't last very long. It took 15-20 minutes to erase them with an ultra violet lamp. The 12F508 comes in an eight-pin DIP, surface-mount (SOIC), and as a DFN mount. In this project, I used the SOIC to save space. The major problem with the SOIC is that they need to be programmed with a special adapter.

I cut my teeth on 555 timers and used them for years. You could use four 555 timers with this project but you wouldn't be able to get them on a 1" square board. With the 508, it will do more than a 555 without any resistors and capacitors. Its timing accuracy with its internal oscillator is ±1%. It has five outputs or six inputs, and can sink or source 25 mA. It can run on a voltage from two to 5.5 volts, however, I often use six volts (two AAA batteries). When put to sleep, it only draws 100 nano amps (that's .0000001 amps). If used as a timer, you time from micro seconds to year. You can program it in circuit or out if you use programming pins or pads. Cost is $.84 single quantity vs. the 555 at $.52 ea.

CONSTRUCTION

All you need is a soldering iron with a fine point, solder, and solder wick. I use the smallest solder wick I can find. Solder paste is nice but the components are big enough you can use .60 mm solder.

The easiest way to solder the 12F508 is to locate pin 1 and make sure it goes to pad 1. Tack it with a bit of solder and check to make sure all the pins are on their own pads. You can solder the pins one at a time or flood solder on them all and then using solder wick to pick up the excess. Inspect with a magnifier for solder bridges. Next, solder the LEDs noting their polarity. Their polarity is marked by two dots on the top (cathode) and often a diode symbol on the bottom. (See **Figure 5**.) The board is marked with + for the anodes. Place the colors opposite each other, e.g., red at 0 degrees and 180 degrees, green 90 and 270, and so on. Solder the resistors; they have no polarity. Although a 15K resistor can be used, I made the board able to accommodate a 20K ohm potentiometer so that the sensitivity can be changed. This goes in the center and the 100 ohm resistor goes in the corner. Turn the board over and solder the microphone, noting its polarity.

Place a drop of solder in the center square pad for a contact. Solder the battery holder to the bottom of the board. Note that the opening should be opposite the microphone. Make sure they are not touching as the microphone case is grounded and the battery holder is positive.

FLASHING IT

Place the 1025 battery into the holder with its positive side up. Rap the board on the counter and the LEDs should light. Adjust R1 for the sensitivity you desire. For mounting, I would use Velcro or double-sided tape on the battery holder and also on the rim of the drum or the center of a speaker.

Make sure you remove the battery for storage. **NV**

A complete kit to go with this article can be purchased online from the *Nuts & Volts* Webstore www.nutsvolts.com or call our order desk at 800-783-4624.

BUILD THE 3D LED MATRIX CUBE

By John Iovine

This article shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. I chose this size because it provides the best value for construction time and programming time for 3D patterns. The advantage to using monochromatic LEDs is that they are very bright, so are therefore easy to see — even in a well lit room. The current crop of RGB LEDs doesn't have the same light intensity punch and when used in the 3D LED cube, can only be viewed in a dark room.

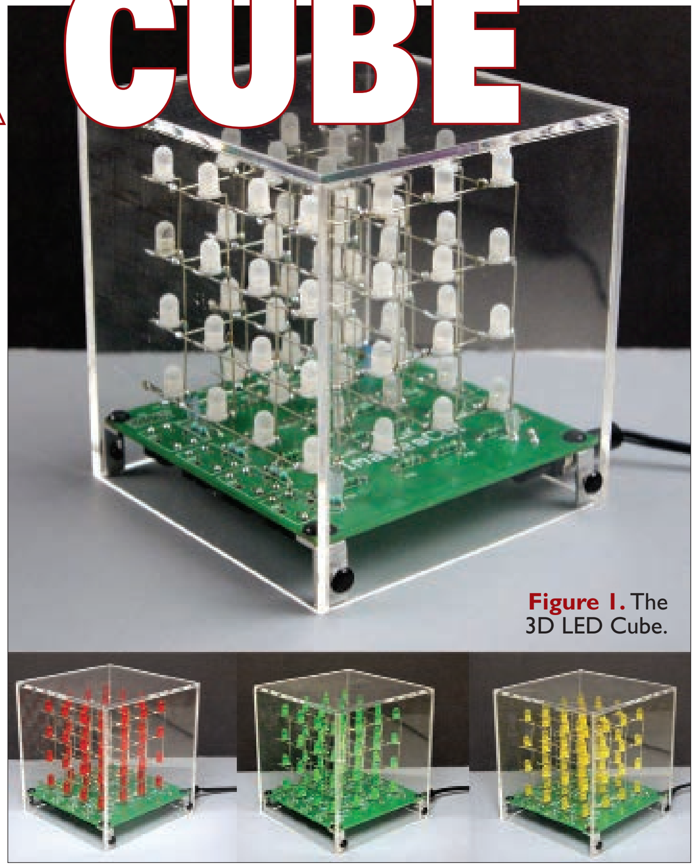


Figure 1. The 3D LED Cube.

You have a choice of four monochromatic colors: blue, red, green, and yellow (see **Figure 1**). The available kit from **www.nutsvolts.com** comes with a preprogrammed microcontroller that includes 29 pre-programmed patterns that will automatically play. Runtime for the 29 patterns is approximately 6-1/2 minutes. Instructions on how to program the 3D matrix yourself are provided in this article.

We will need the following tools for building both the LED cube and jig:

1. Soldering iron and solder.
2. Needlenose pliers.
3. Diagonal (lead) cutters.
4. 20 gauge solid wire; one meter in length.
5. One 2AAA battery holder.
6. Two AAA batteries.
7. 1K 1/4 watt resistor.

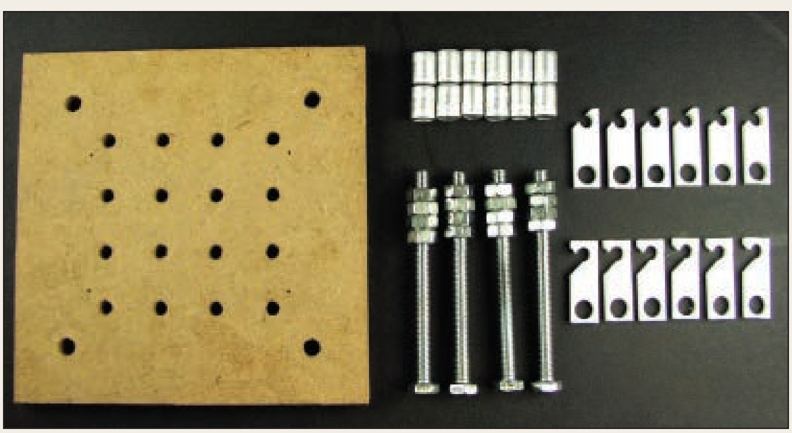


Figure 2. Jig kit including wood base, LED holders, spacers, nuts, and bolts.

LED Cube Construction

To accurately build the LED cube matrix so that everything is evenly spaced and aligned, I recommend building the jig described here. (I will have these jigs available for rent at **www.imagesco.com**). Instructions for using a jig are provided in this article. If you do not wish to use a jig, you can fabricate the LED matrix in any manner you find convenient. The jig contains the following parts: wood base; four 1/4-20 bolts, four inches in length; 16 1/4-20 nuts; 12 LED holders (two styles); and 12 spacers. See **Figure 2**.

Physical Layout

The LED cube has 64 LEDs in a 4 x 4 x 4 matrix (see **Figure 3**). There are four LEDs across the length, width, and height. Each of the four layers contain 16 LEDs. Physical dimensions are approximately 2.5″ x 2.5″ x 2.5″, with consecutive LEDs separated by 3/4″.

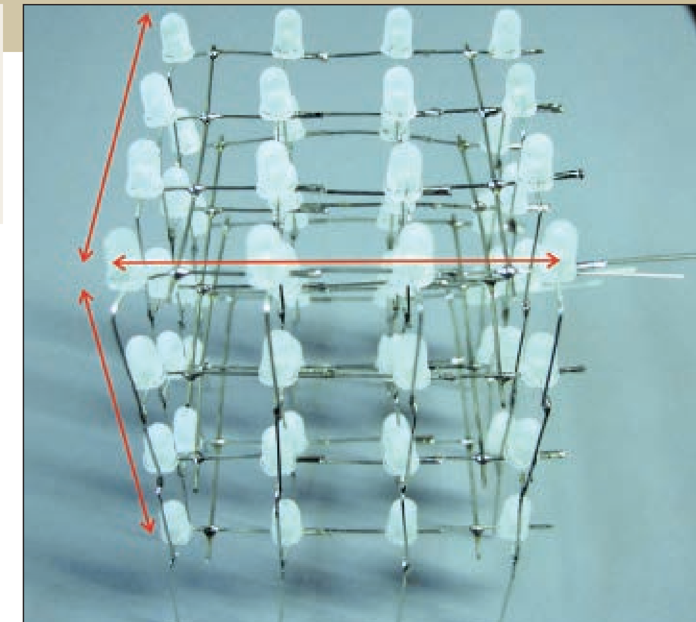


Figure 3. LED matrix (4 x 4 x 4).

LEDs

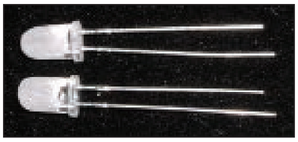


Figure 4. LED, showing lengths of the cathode and anode leads.

The LEDs are round and measure 5 mm (T1-3/4) (see **Figure 4**). Rectangular/oval shaped LEDs can be used, but may be a problem to hold during construction. A diffused lens will give you a wider viewing angle, especially for side viewing, so keep these things in mind if you're purchasing your own LEDs. For the kit, we purchase super bright clear LEDs and sand blast them ourselvs to diffuse the lens. LED leads should be 1″ long and are used to connect consecutive LEDs (which are 0.75 " apart).

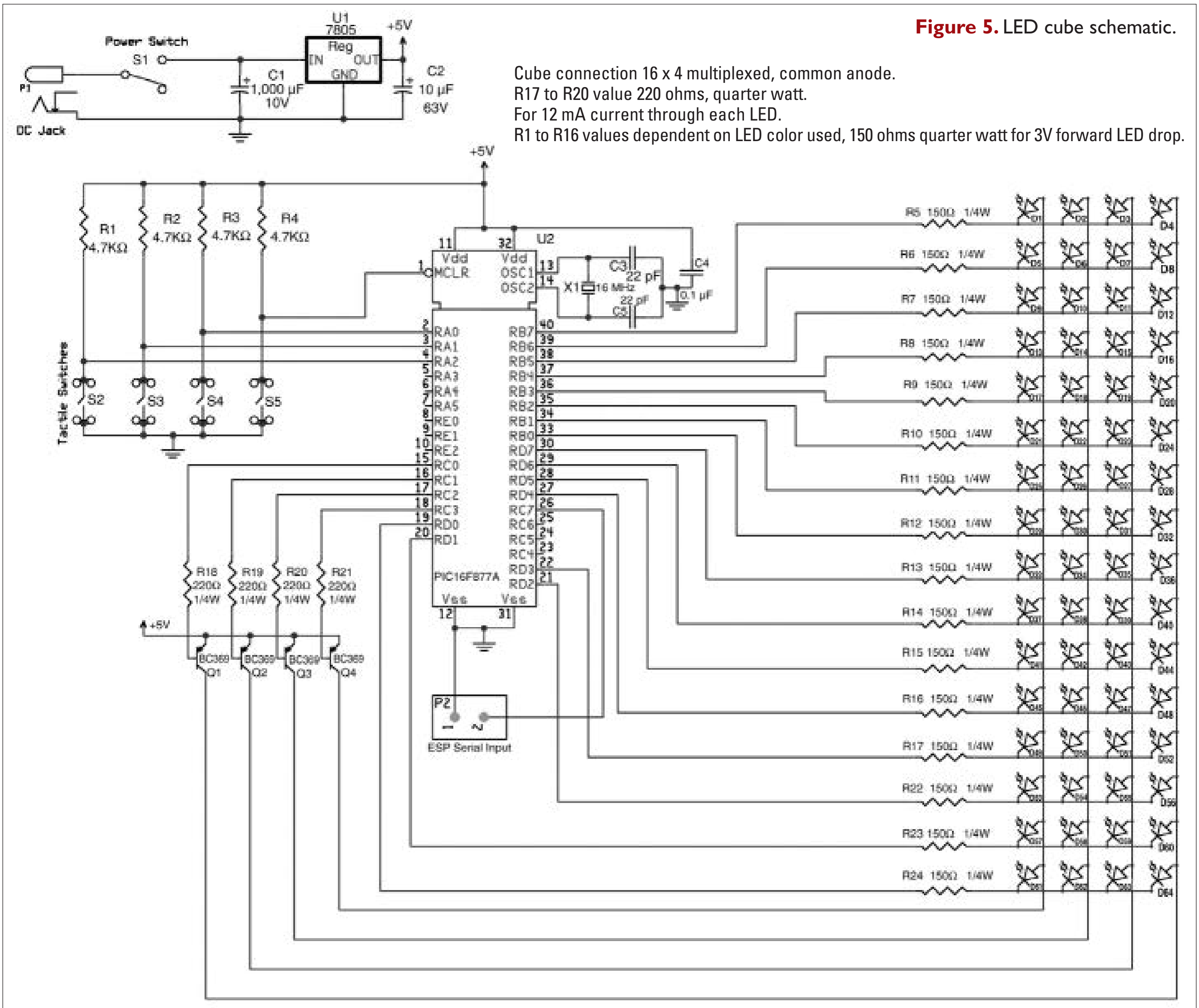


Figure 5. LED cube schematic.

Electrical Connection

The schematic for the 3D LED cube is shown in **Figure 5**. The cube is a 16 x 4 multiplexed display, with 16 common cathode connections and four common anode connections. Each LED layer has 16 LEDs with one common anode. So, a total of four layers means four common anode connections. Each of four LEDs in a vertical line have a common cathode connection. So, 16 vertical lines means 16 common cathode connections.

Check Those LEDs

While the failure rate of new LEDs is extremely low, I recommend checking your LEDs before soldering them into the 3D matrix structure. It is very difficult to replace a faulty LED once the matrix is constructed. A simple LED tester can be made to help with this (see **Figures 6** and **7**). Take a two-battery AA or AAA battery holder and solder a 1K resistor in series with the black lead to limit current to the LEDs. This is shown as the –ve terminal; the red wire probe is the +ve terminal.

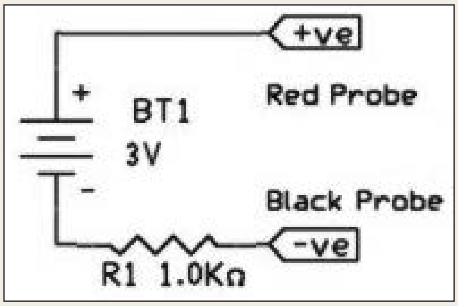


Figure 6. LED tester schematic.

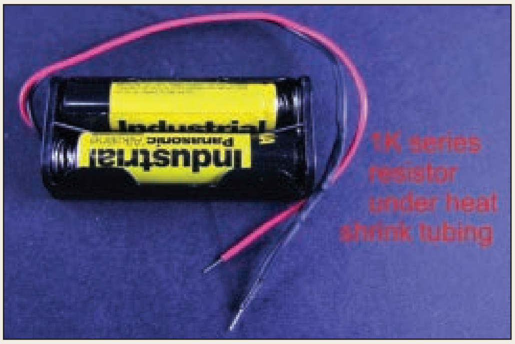


Figure 7. LED tester.

Cube Jig

The layout for the jig is shown in **Figure 8**. The jig itself is made from 6″ x 6″, 3/8″-1/2″ thick wood. (The construction manual that comes with the kit has information on building your own jig.) The 16 holes with red numbers are 7/32″ diameter, or a #7 drill bit. LEDs are inserted upside down into these holes.

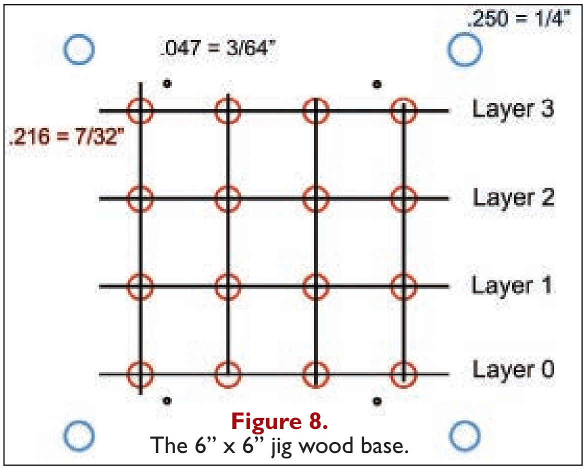


Figure 8. The 6" x 6" jig wood base.

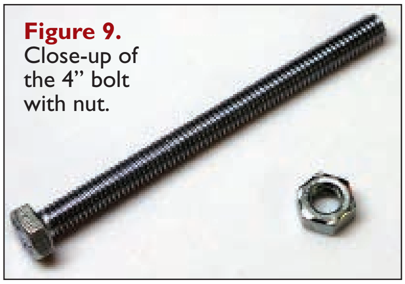


Figure 9. Close-up of the 4" bolt with nut.

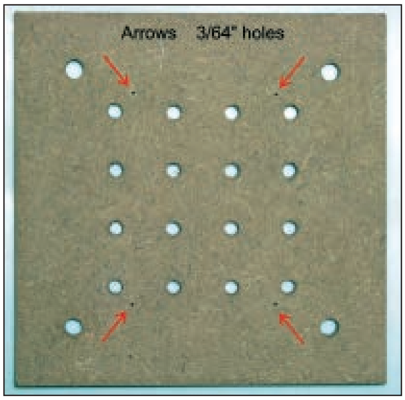
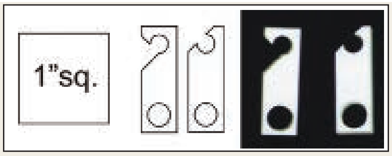


Figure 10. Wood base for the jig; arrows show the 3/64" holes for the tinned wire.

The four holes with black numbers are .047″ in diameter; so that's a 3/64″ or #56 drill bit. These holes are used to hold thick tinned wires (about 0.8 mm wire without sleeve) in place. The four holes with blue numbers are 0.25″ in diameter, or a 1/4″ drill bit. Drill these holes to hold 1/4–20 bolts at the required positions. Be sure to label the jig with the diameters of the holes so that you can orientate it properly later. You'll need to use spacers and nuts with the jig to achieve a height of 0.75″ between the layers. If you don't use a jig, your goal is to stack up three to four nuts to achieve a height of 2 cm (0.75″). The four 1/4″ holes hold bolts (like the one in **Figure 9**) to which the LED holders attach to keep the layers one above the other at accurate positions. (This is actually not as complicated as it sounds; the photos should help.) Bolts are a minimum of 4″ long**.** The 16 holes in the 4 x 4 grid are 13/64″ in diameter (or a #7 drill bit). These holes will be used for making individual layers. **Figure 10** shows the jig base.

LED Layer Holders

Figure 12. LED holders.



The holders are used to keep the layers at their required position above each other. We need a total of 12 holders, divided into two groups of six. Please note that both groups have minor differences (see **Figure 11**). **Figure 12** shows the strip designs. Use this layout in **Figure 12** to make the strips. Glue the layout on 16 gauge sheet metal. Cut the sheet metal along the border lines with tin snips. Then, use a hole-punch to mark the centers where the cross lines intersect and drill the holes with the proper bits. The holes in the LED holders are 17/64″ as opposed to 1/4″ in the jig base. This slight increase in the hole allows for adjustments when positioning the LED layers. Holders are made from 1/8″ acrylic.

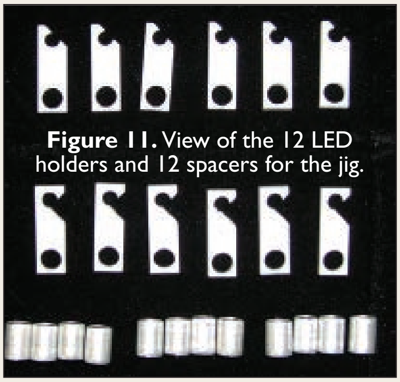


Figure 11. View of the 12 LED holders and 12 spacers for the jig.

Tin Copper Wires

We will need two pieces of tin wire 3.25” in length per layer, for a total of eight wires for the entire LED cube construction (see **Figure 13**). If you are using wire from a reel, it should be straightened for proper and easy soldering. Try straightening the wires as much as possible by hand. Then, put the wires on a firm surface (e.g., desk) and use the jig base to roll over the wires, using some pressure. This will straighten the wires out perfectly.

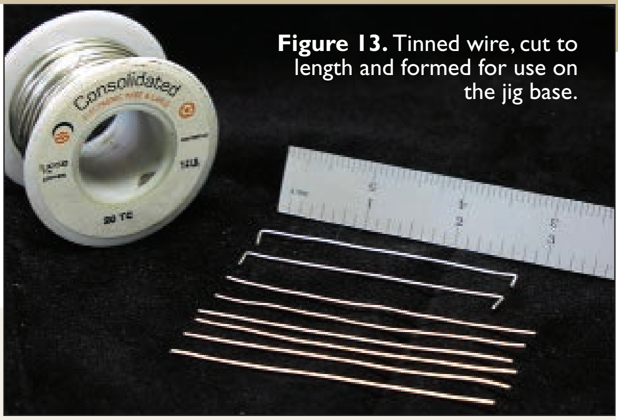


Figure 13. Tinned wire, cut to length and formed for use on the jig base.

Tin Wire Forming

Orient the jig such that the 3/64” drill holes are at the top and bottom of the jig. Each of the eight tin wire pieces needs to be formed into a “C” shape for accurate and easy soldering. There are four small 3/64” holes drilled into the jig; with correct orientation, a pair of them will be in a vertical line perpendicular to you. Take a tin wire piece and make a 90 degree bend at 1/4” from one end. Insert the 1/4” bent portion into one of the 3/64” holes and align the wire with the corresponding 3/64” hole along a vertical line. Hold the wire over the second hole with needlenose pliers. Lift the bent end from the previous hole and make a 90 degree bend at the position where the pliers hold the wire. (This will give the C shape to the wire.) Repeat this process for the rest of the eight wire pieces. **Figure 14** shows the jig with two tin wires preset into the 3/64” holes, ready to start accepting LEDs.

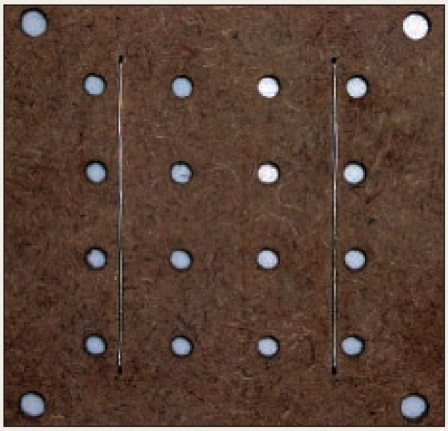


Figure 14. Jig base with two tinned wires loaded.

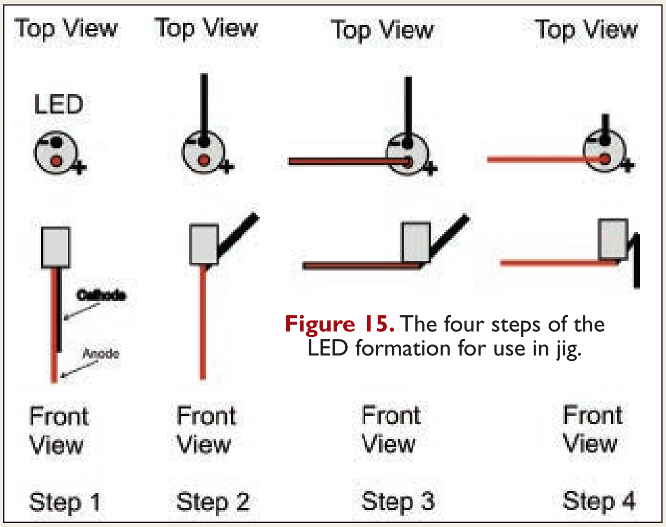


Figure 15. The four steps of the LED formation for use in jig.

LED Leads Forming

We are going to use LED leads to help solder the LEDs into each layer. The most important step is bending the LED leads properly. **Figure 15** provides an overview of the four steps. The longer lead of the LED is the anode and is colored red. The shorter lead is the cathode and is shown as black. All 64 LED leads will be formed in the following manner, so let’s proceed in a step-by-step process for the LED bending.

First, hold the LED in your hand so that the anode and cathode leads are perpendicular to you (**Figure 16**). The anode should be closer to you, while the cathode should face outwards. From the base of the cathode leads, bend the lead outwards by 90 degrees along the line perpendicular to you (**Figure 17**). Next, bend the anode lead from its base by 90 degrees to the left, along a line parallel to you (**Figure 18**).

Holding the leads from the middle portion is an efficient way of bending them. Now, the anode and cathode leads should be perpendicular to each other. Hold the bent cathode lead 1/4” from its base using needlenose pliers and make a **90** degree downward bend so the pliers are holding the cathode Lead (**Figure 19**). After bending the 64 LEDs in the above manner, we are ready to begin building the LED cube layers.

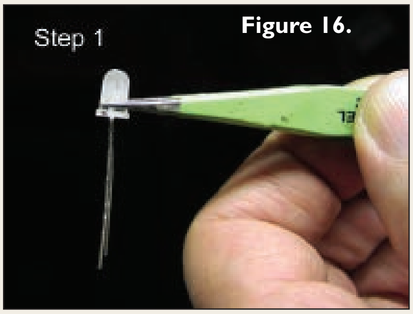


Figure 16.

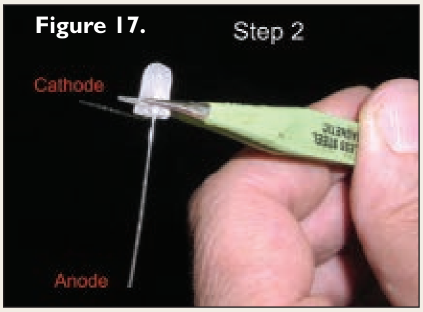


Figure 17.

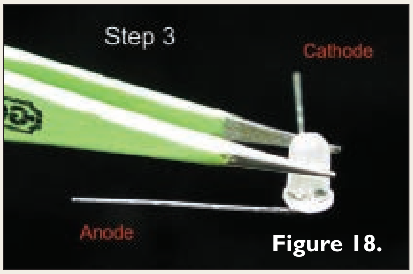


Figure 18.

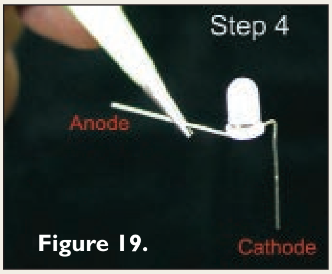


Figure 19.

Making Individual Layers

Orient the jig so that the 3/64″ holes are at the top and bottom of the jig facing you. Each layer is made up of 16 LEDs.

Put two C shaped tin wires in place along the four 3/64″ holes (**Figure 20**). The small bent sections of each tin wire should go inside the 3/64″ holes. The orientation of each tin wire is such that it is perpendicular to the anode lead's horizontal lines and is vertical to you.

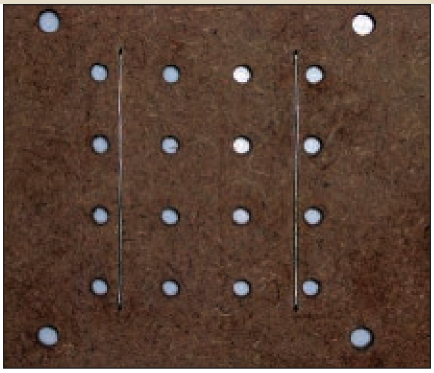


Figure 20. Jig base with two tinned wires loaded.

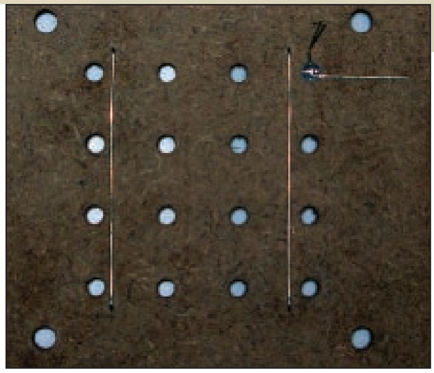


Figure 21. Jig base with two tinned wires and the first LED loaded.

Insert the bent LEDs upside down into the top right corner 7/32″ hole (**Figure 21**). The LED should be oriented such that the bent anode lead (which rests parallel with the jig surface) is along the horizontal line parallel to you. Insert a second LED into the hole to the left of the previous one using the same orientation. Repeat this process for the rest of the LEDs. There are 16 7/32″ holes. When all 16 LEDs are inserted (for each row of four LEDs), their bent anode leads should overlap some and form a horizontal line. The LEDs are soldered together at the overlaps as shown in **Figure 22**.

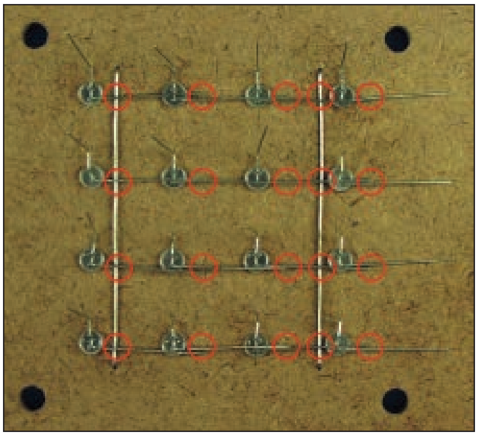


Figure 22. Layer 0's LEDs loaded into the jig base; red circles show the solder joint points.

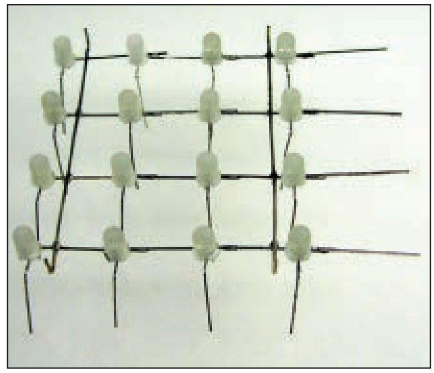


Figure 23. Layer 0 soldered and removed from the jig.

Solder the overlapping anode portions; there is a total of 12 solder joints. Solder the LED leads quickly since a longer contact period between the solder iron and the leads could damage the LED. Also remember to solder the eight solder joints where the two tin wires overlap the four horizontal anode lead lines. When you are finished, the LED layer may be removed from the jig (**Figure 23**). Cut off the extra portion of tin wire that went into the 3/64″ holes (**Figure 24**).

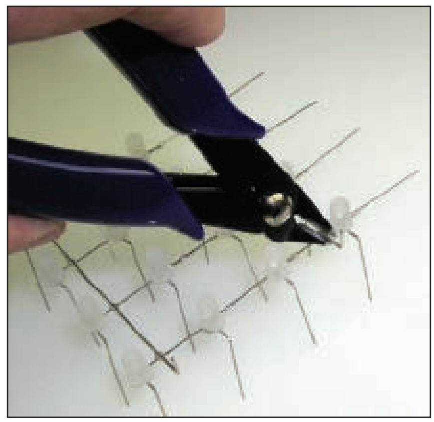


Figure 24. Clipping the ends of the common anode wires.

At this point, test the LEDs in the completed layer. To do so, touch the positive terminal to the uncut common anode lead on the right and the negative terminal to the upwardly bent cathodes one by one. Continue building three more layers in the same manner. Lift the layers from the jig carefully by pulling the bent cathode leads (**Figure 25**).

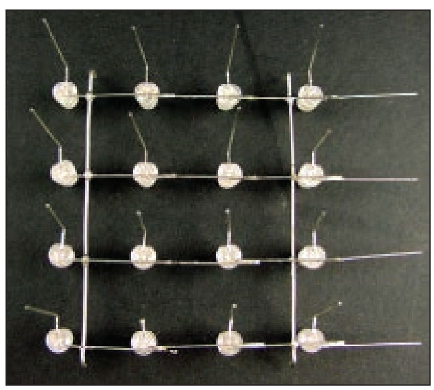


Figure 25. Bottom view of the additional layers.

Notice in **Figure 25** on the right-hand side of the layer that we have four extended anode leads. Each layer only needs one extended anode lead. Each layer differs from another in terms of the position of the uncut common anode layer lead; this is how we will be referring to the different layers from now on. Arrows in **Figure 26** indicate the uncut common anode layer leads.

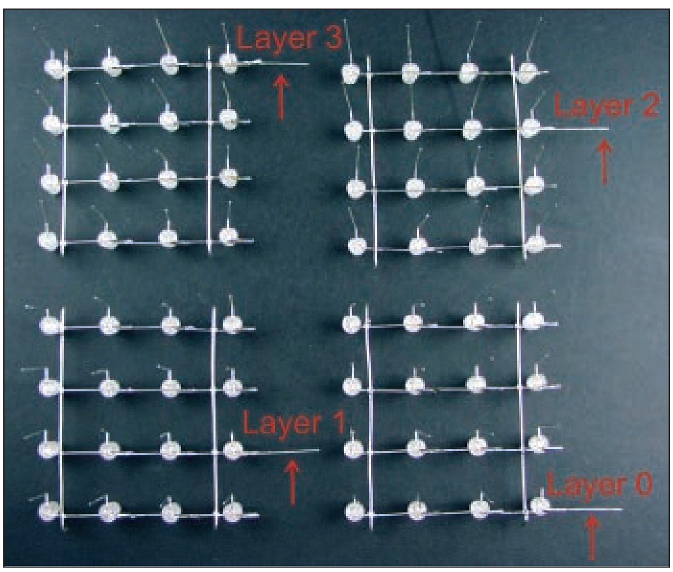


Figure 26. Bottom view of four layers, showing the extended anode lead for each layer.

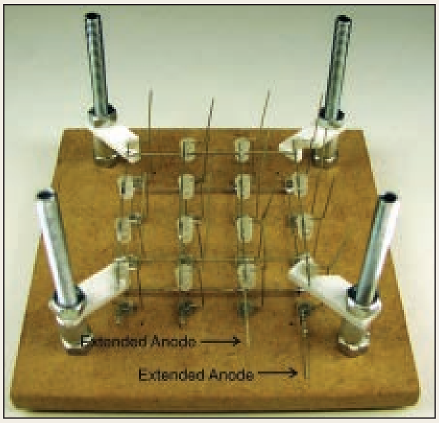


Figure 27. Layer 0 and Layer 1 in the jig, showing the placement of the extended anode leads.

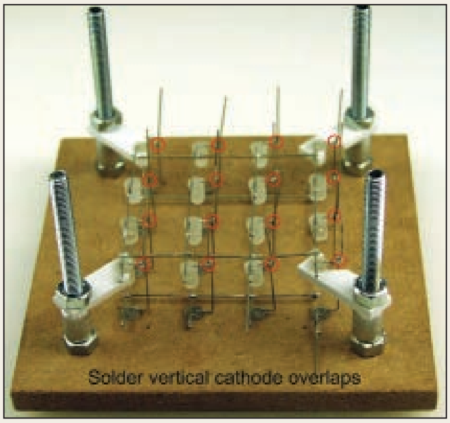


Figure 28. Layer 0 and Layer 1 in the jig; the red circles show the common cathode solder joints.

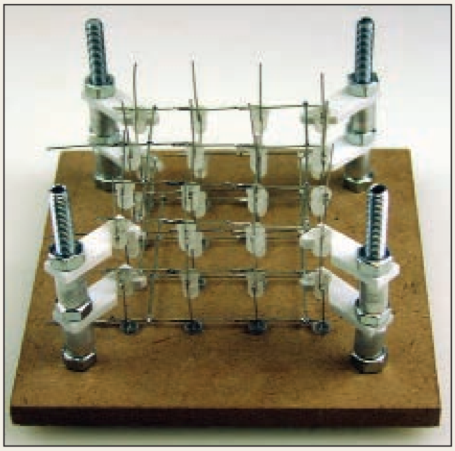


Figure 29. Layers 0, 1, and 2 in the jig.

Soldering Layers to Form the Cube

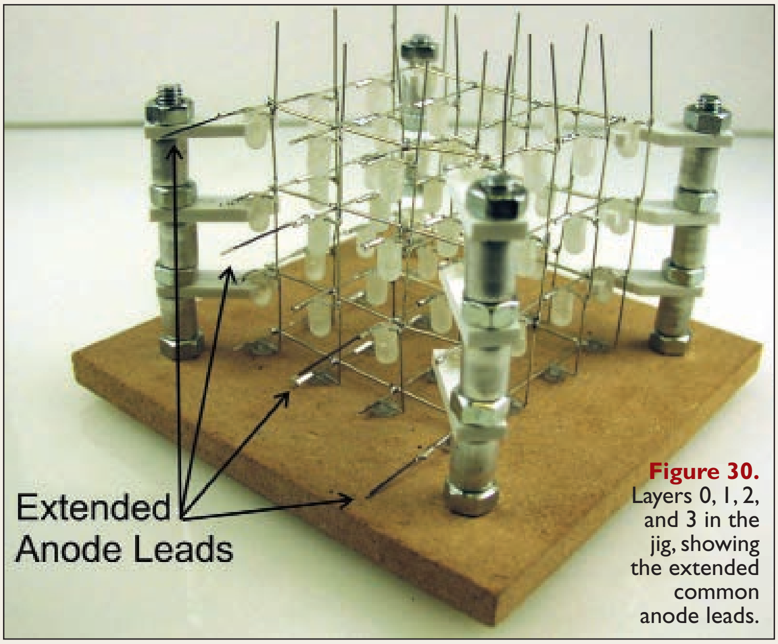


Figure 30. Layers 0, 1, 2, and 3 in the jig, showing the extended common anode leads.

Insert four bolts into the four corner 1/4" holes. Secure each bolt to the jig using a 1/4-20 nut. Place LED Layer 0 in the jig holes. Pay attention to which side has the extended anode lead. Go to each bolt. On top of the existing nut, place a spacer, then on top of the spacer place an LED holder. Then, on top of the LED holders, put another 1/4-20 nut to secure everything down. Fasten Layer 1 into the LED holders. Keep the extended anode leads on the same side (**Figure 27**). The bent cathode leads of the LEDs layers should overlap. If the cathode leads don't touch each other, use narrow tweezers to bend them so they are in contact with the previous LED layers. Solder the overlapping cathode leads; there are a total of 16 cathode solder joints per layer as shown in **Figure 28**. The red circles highlight the vertical cathode solder joints for Layers 0 and l. Place Layer 2 with the four corner LEDs into the four corner hole's LED holders (**Figure 29**). Solder the vertical cathode leads as before. Do the same thing for Layer 3. Notice how the extended anode leads are positioned on one side of the 3D LED matrix. The four extended common anode layer leads (one for each layer) end up in a staircase pattern (note the arrows). This is important for when we solder the matrix to the printed circuit board (PCB). You can test the LEDs before mounting the 3D LED matrix to the PCB. Unscrew all the nuts and remove the spacers and LED holders so you can get the LED matrix cube out of the jig. Your 3D matrix should look like **Figure 30**.

Transparent Case Mounting

There is an optional transparent case available for the LED cube (**Figure A**). The case itself may be purchased assembled or in a kit (also at **www.nutsvolts.com**). The instructions for assembly are provided with the cube, so we will not repeat them here. The holes in the transparent case line up with the holes for the L brackets and the DC jack on the lower end.

When it's time, you'll slide the case over the cube (with the PCB) with the correct orientation so that holes for the L brackets and DC jack match with those on the PCB. Insert the plastic push rivets from outside of the case into the holes on the case that match with the L bracket long arm hole.

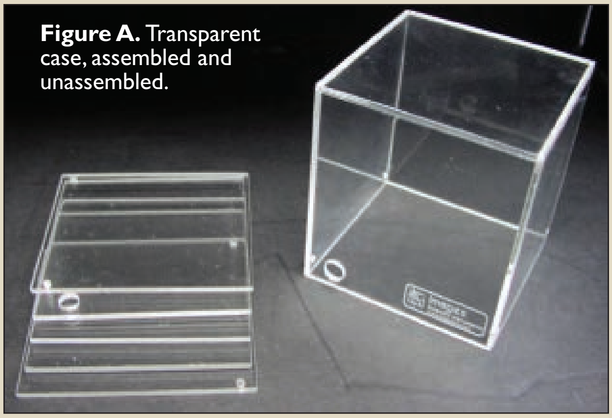


Figure A. Transparent case, assembled and unassembled.

Cube PCB Construction

Now it's time to mount the components on both sides of the PCB. The LED PCB is shown in **Figure 31**. The side with the text "ImagesCo." is the top side of the board.

Mount and solder the components as described next. Depending on which color cube you select, the resistors R1, R2-R4, and R5-R16 will have different values. For the RED cube, the value is 220 ohms. For the YELLOW and GREEN cubes, the value is 180 ohms. For the BLUE cube, the value is 150 ohms.

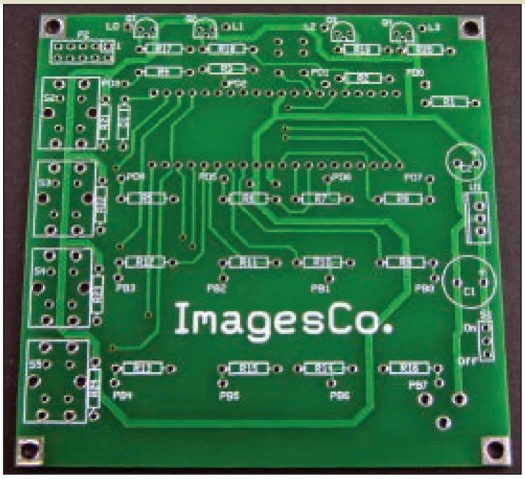


Figure 31. Top view PCB.

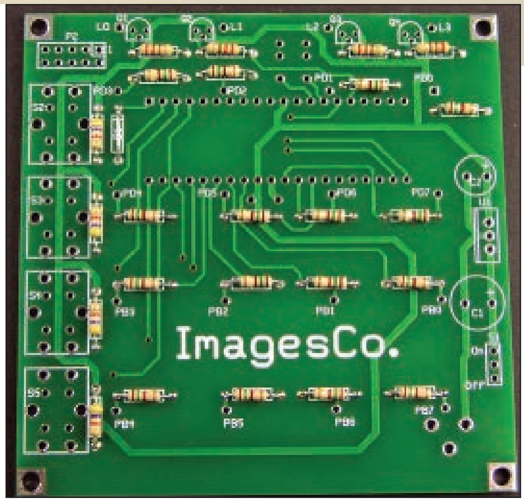


Figure 32. Top view PCB (populated with resistors).

On the top side of the PCB:

1. Mount resistors R1 (150/180/220 ohms), R5–R16 (150/180/220 ohms), R21–24 (4.7K ohms), and D1 (tin wire jumper in its place) on the top side (see **Figure 32**).

2. Mount resistors R2–R4 (150/180/220 ohms).

On the bottom side of the PCB:

3. Mount and solder R17–R20 (220 ohms) and crystal X1 (16 MHz).

4. Place the DIP 40-pin socket on the bottom side.

5. Place the capacitors C3 and C4 (22 pf).

6. Place capacitor C5 (0.1 µF).

7. Mount transistors Q1–Q4 (BC 369) on the bottom side. The flat side of the transistor faces in toward the PCB. The curve/arc of the TO-92 transistor package faces outward from the PCB. **Figure 33** shows a close-up of the transistors.

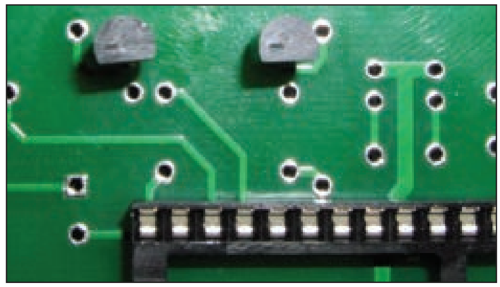


Figure 33. Bottom view PCB; close-up shows the transistor orientation.

8. Mount the tactile switches S2–S5 onto the bottom side.

9. Mount the slide SPDT switch S1 onto the bottom side. The footprint is smaller than the switch lead pitch, so you will have to bend the outer leads slightly inwards using needlenose pliers.

10. Mount electrolytic capacitor C2 (47 µF/50V).

11. Mount electrolytic capacitor C1 (1,000 µF/16V).

12. Mount regulator U1 (7805) onto the bottom side. Bend the regulator inwards from the bottom side completely so that it touches the PCB. Cut the extra (unwanted) leads from the top side, and then solder the leads from the top side.

13. The 10-pin header P2 is optional and doesn't have to be mounted. **The DC barrel jack P1 should be soldered after the LED matrix is mounted on the top side.**

14. Insert the programmed PIC16F877A into the 40-pin DIP socket with the correct polarity on the bottom side of the PCB. You can use a pin straightener for easier insertion into the IC socket (see **Figure 34**).

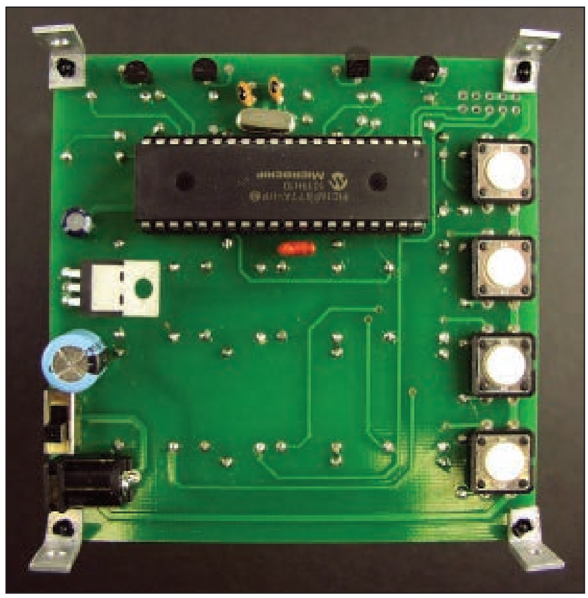


Figure 34. Bottom view PCB, populated.

15. Mount the four L brackets on the bottom side of the PCB with four small snap rivets. With the correct orientation, align the hole on the **small** arm of the L bracket to that of the mounting hole on the bottom side of the PCB. Place the small pop rivet from the top side into the mounting hole and press it until the rivet head touches the PCB. This will secure the L bracket into its place (see **Figure 35**).

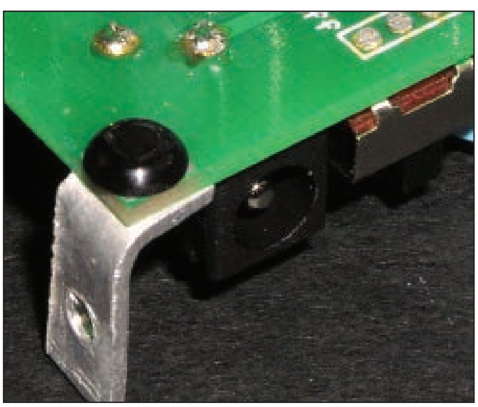


Figure 35. Close-up of the L bracket, held with a plastic push rivet.

Soldering the Cube to the PCB

Place the LED cube on the top side of the PCB with the correct orientation as shown in **Figure 36**. The uncut common layer anode leads should be facing the side of the board where the transistors are mounted. (See the edge of the PCB with four pads named L0, L1, L2, L3.) There will now be 16 pads named PD0 to PD7 and PB0 to PB7 for connection of the common cathode lines.

The 16 common cathode leads of the cube should be inserted into these pads using narrow tweezers. It would be helpful to tilt the cube a little and then insert the four common cathode leads by row. See a close-up in **Figure 37**.

Try to keep a distance of 1/2" between the PCB and the LEDs to avoid unwanted short circuits between the cube connections and PCB pads. Make sure the cube is symmetrically placed on the PCB. Now — **with the exception of the pins PB7, PD2, and PD1** — the common cathode leads can be soldered on the bottom side. You will need to hold the cube PCB upside down to solder the common cathode pins on the bottom of it (see **Figure 38**).

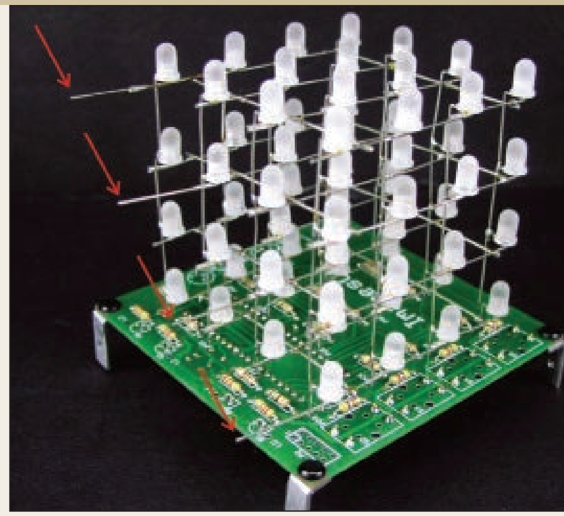


Figure 36. LED matrix on the PCB; note the extended anodes above the L0 to L3 PCB pads.

Trim the excess wire from the soldered common cathode leads leaving a tiny portion from the roots, and then solder the pins. Solder PB7, PD1, and PD2 from the top side of the PCB. On the bottom side, cut PB7 flush to the PCB, and the PD1 and PD2 leads as short as you can.

Place the P1 DC barrel jack on the bottom side. Since PB7 pin overlaps the P1 jack, it is necessary that PB7 is cut flush to the PCB (or else the jack won't sit flush to the PCB which again means the jack wouldn't sit in its place properly). The barrel jack pins are smaller than the hole diameters on the PCB, so make sure the jack is soldered such that it matches the component outline. Solder one of the pins of the jack on the bottom side.

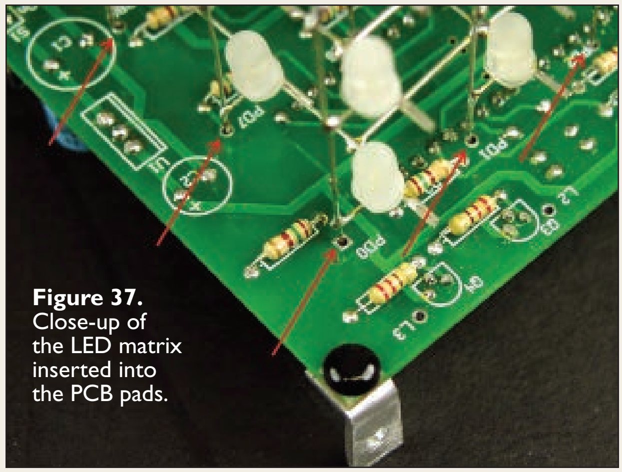


Figure 37. Close-up of the LED matrix inserted into the PCB pads.

Turn the PCB upside down and solder PB7 and the remaining P1 jack pins on the top side of the PCB. Also solder PD1 and PD2 from the top side of the PCB.

Now, the four common layer anode leads (the uncut staircase type leads; see **Figure 36** again) need to be joined to their corresponding pads (L0, L1, L2, L3) on the PCB. Use straightened tin wire pieces to make the connections. The common anode to bottom Layer 0 is connected to pad L0. The common anode on Layer 1 is connected to pad L1. The common anode to Layer 2 is connected to pad L2. Finally, the top (Layer 3) common anode lead should be connected to pad L3. Next, trim the extended anode leads to the solder joints as shown in **Figure 39**.

A close-up of the anode connection wires going into the PCB pads is shown in **Figure 40**.

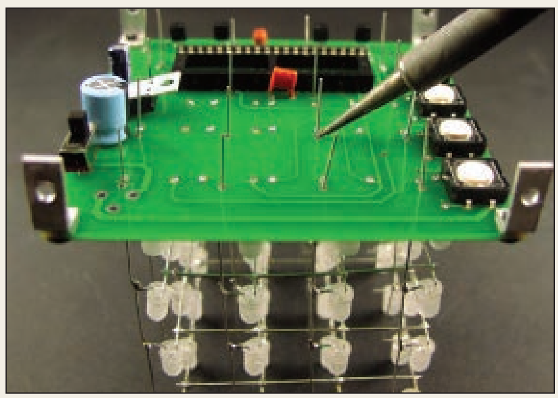


Figure 38. Soldering the LED matrix on the bottom side of the PCB.

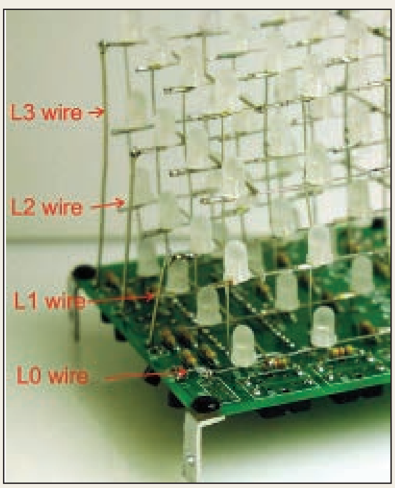


Figure 39. Connecting the common anodes to the PCB using tinned wire.

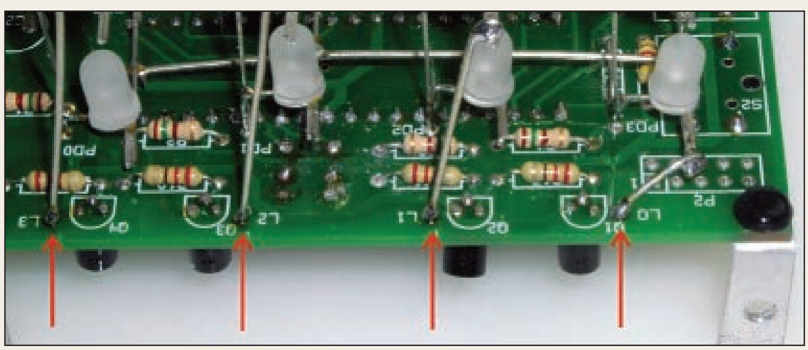


Figure 40. Close-up of the tinned wire soldered in the L0, L1, L2, and L3 PCB pads.

Operation

Insert the DC female jack into the male socket with the correct polarity and voltage/current ratings of the DC adapter (**Figure 41**). Slide the switch S1 to the on position by lifting the cube. Once powered on, the cube will blink all the LEDs five times before beginning to display the other patterns in the demo mode. Once all the patterns have been run, the process will start again.

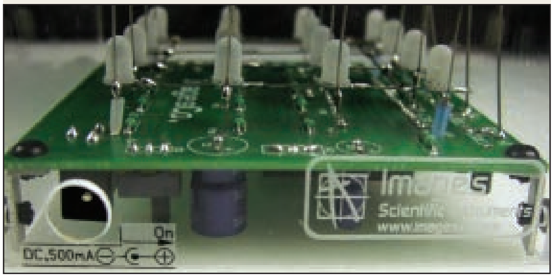


Figure 41. Power connection.

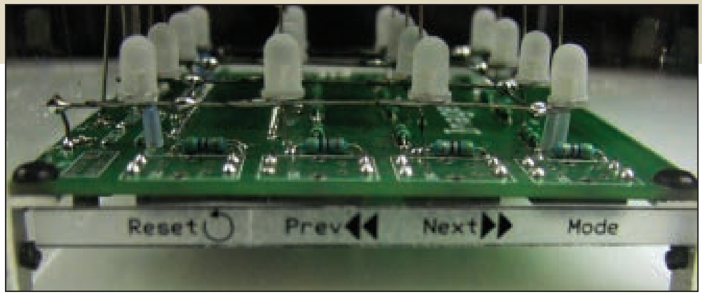


Figure 42. Button label.

Pushbutton Controls

There are four pushbutton switches (S2 to S5) for various functions. These buttons can be accessed by lifting the cube since the switches are mounted to the bottom side of the PCB (see **Figure 42**).

The S2 switch is for reset. The S5 switch is the mode switch. Pressing it will toggle between the demo mode and user mode. In user mode, a single pattern will run over and over again.

S3 and S4 act as previous/next (cycle) pattern switches in user mode for selecting a particular pattern.

DIY 3D LED Cube Programming

Code for the 16F877A is provided at the article link shown. If you aren't satisfied with the canned patterns provided in the PIC, you can try your hand at programming the LED cube yourself.

Programming the LED is not very difficult, but does take some thought. Look at **Figure 43**. Here's how our LEDs are wired inside the cube. We have common anode lines for each of the four layers. Each vertical column (1-16) consists of four vertical LEDs tied to a common cathode. We turn individual LEDs on and off by bringing the I/O lines of the PIC high or low. We control the current to the common anode lines using four I/O lines (RC0-RC3) connected to four power transistors. Bringing this line low will turn on the transistor, allowing current to flow into the common anode line to the LEDs. The individual LEDs are controlled by the I/O lines on port B and port D. Bringing a port I/O line low will turn on that LED.

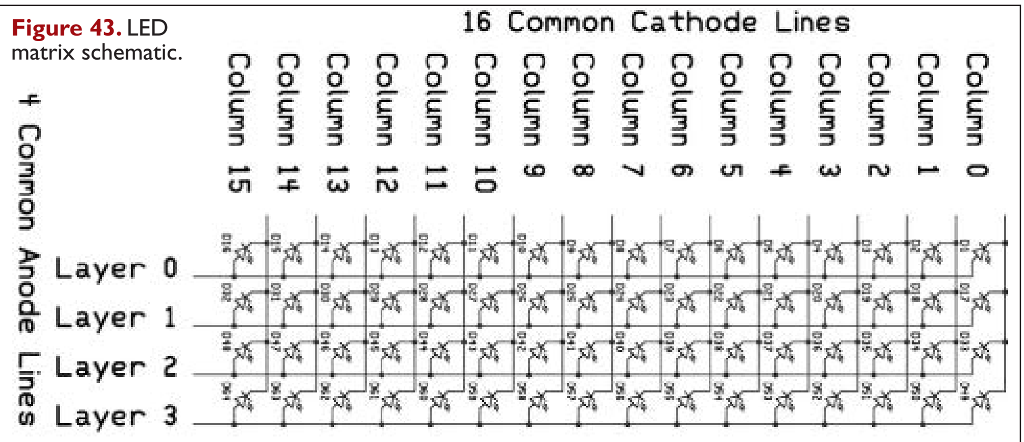


Figure 43. LED matrix schematic.

Essentially, by controlling the I/O lines we can turn on any individual LED inside the 3D matrix, any group of LEDs inside the 3D matrix, or all of the LEDs inside the matrix. Patterns are created by sequencing LED patterns on and off rapidly. Here are some general rules:

- A high output (1) on an I/O line on port D or port B will turn the LED off; low (0) will turn the LED on.
- A low (0) on port C will make the PNP layer transistor turn on; a high (1) will turn it off.
- Sixteen cathode rows can be turned on/off by the port pins directly (port B and port D).
- Four anode columns can be turned on/off by the port pins via PNP drivers (port C).

Test Program Function

Layers are scanned one by one. When a particular layer is active, a two-byte/16-bit word is simultaneously outputted on the cathode pins.

Byte-sized variables are:

- a[0] and a[1] are cathode data for Layer 0.
- a[2] and a[3] are cathode data for Layer 1.
- a[4] and a[5] are cathode data for Layer 2.
- a[6] and a[7] are cathode data for Layer 3.

Layer 0 is the bottommost; Layer 3 is the topmost.

To create a pattern, set the frame repetition rate (the time one frame stays on). Load the appropriate frame data into the array variable and use a subroutine frameout for outputting the frame. Load the next frame data into an array variable and follow the same procedure.

```
'PIC 16F877A
Define OSC 16  'oscillator 16 MHz, HS mode
adcon1 = 6     'make ports digital I/O

alloff con %11111111  'all cathodes off
allon con %00000000   'all cathodes on
cubeoff con %00001111 'all anodes off
layer0 con %00001110  'layer 0 Floor enabled
layer1 con %00001101  'layer 1 enabled
layer2 con %00001011  'layer 2 enabled
layer3 con %00000111  'layer 3 Top enabled

x var byte       'x,z general variables
z var byte
a var byte [8]   'array variable stores current
                 'frame data for LED cathodes
frame var byte   'single frame repetition rate
                 'variable

trisb = 0     'portb, pord connects to cathode,
              'portc connects to anode
trisd = 0
trisc = 0

portc = cubeoff   'disable all layers (pnp
drivers)
portd = alloff    'initialize all cathode driving
                  'ports to off condition
portb = alloff

inout:        'inout pattern - a subset square
              'and outermost square turn on
              'alternately
frame = 24    'set frame repetition rate
a[0] = %11111111     'set initial frame data
a[1] = %11111111
a[2] = %10011111
a[3] = %11111001
a[4] = %10011111
a[5] = %11111001
a[6] = %11111111
a[7] = %11111111
inouti:              'main loop of pattern
gosub frameout       'ouput current frame
for z = 0 to 7       'invert frame data to
                     'interchange squares
  a[z] = ~a[z]
next z
goto inouti          'go back to main loop
```

```
frameout:            'main frame out subroutine
for x = 0 to frame   'display patterm as per set
                     'frame repetition rate
  portd = a[0]       'set ports connected to
                     'cathode pins for Layer 0
                     'frame data
  portb = a[1]
  portc = layer0     'enable layer 0 anodes
  pause 4            'pause for layer on time -
                     'set refresh rate to avoid
                     'flickering
  portc = cubeoff    'disable all layers while
                     'changing cathode data from
                     'one layer to another
  portd = a[2]       'set ports connected to
                     'cathode pins for Layer 1
                     'frame data
  portb = a[3]
  portc = layer1     'enable layer 1 anodes
  pause 4            'pause for layer on time -
                     'set refresh rate to avoid
                     'flickering
  portc = cubeoff    'disable all layers while
                     'changing cathode data from
                     'one layer to another
  portd = a[4]       'set ports connected to
                     'cathode pins for Layer 2
                     'frame data
  portb = a[5]
  portc = layer2     'enable layer 2 anodes
  pause 4            'pause for layer on time -
                     'set refresh rate to avoid
                     'flickering
  portc = cubeoff    'disable all layers while
                     'changing cathode data from
                     'one layer to another
  portd = a[6]       'set ports connected to
                     'cathode pins for Layer 3
                     'frame data
  portb = a[7]
  portc = layer3     'enable layer 3 anodes
  pause 4            'pause for layer on time -
                     'set refresh rate to avoid
                     'flickering
  portc = cubeoff    'disable layers while
                     'changing cathode data from
                     'one layer to another
next x
return               'return to main program
end
```

TEST PROGRAM.

Wrap-Up

The **test program** shows how to display a simple repeating pattern with the stored templates. Programs can also run live, meaning that the running program determines which LED or group of LEDs light next without reading the information from stored data within the program.

That concludes our 3D LED cube build. This project is a real attention-getter and I'm sure you'll get many hours of enjoyment from it. I know I have. **NV**

PARTS LIST

QTY	ITEM
(1)	PCB-43
(70)	LED-MB (color blue, red, green, or yellow)
(4)	Resistors 220 ohm 1/4 watt
(4)	Resistors 4.7K 1/4 watt
(1)	Resistor 1K 1/4 watt
(16)	Resistors 1/4 watt (150 ohm blue, 220 ohm red, 180 ohm grn and yel)
(2)	Capacitors 22 pf Mono
(1)	Capacitor 10 µf 16V
(1)	Capacitor 1,000 µf 10V
(1)	Capacitor .01 µf 100V
(1)	PIC16F877A
(1)	ICS-40
(1)	SW-06
(4)	SW-25
(1)	Xtal-16
(1)	PJ-102B
(1)	7805
(4)	BC369
(4)	Hex feet
(28")	Tin wire
(4)	440 x 3/8 screws
(1)	ACA-9V DC 300 mA power supply

■ BY FRED EADY

www.nutsvolts.com/index.php?/magazine/article/august2011_DesignCycle

GIVE YOUR BITS SOME AIR

If you think that you need ZigBee or 802.15.4 to move small chunks of data with a low power data radio, you are absolutely correct. If you think that you don't need ZigBee or 802.15.4 to move small chunks of data with a low power radio, you are absolutely correct.

ZigBee and 802.15.4 are wireless data communication standards that require packet-descriptive information to be sent along with the packet's payload data. If you only need to send a couple of bytes in a peer-to-peer or multicast environment, you really don't need (or want) the network overhead that comes with an official ZigBee stack or 802.15.4 network.

When NASA put men on the moon, there was no personal computer for the masses, no USB, no Ethernet (as we know it), no Internet (as we know it), no 802.15.4, and no ZigBee. In addition to these shortcomings, the RS-232 standard was in its infancy. Yet, NASA still managed to get one of the world's largest rockets, three men, a command module, a service module, and a lunar module to make the 500,000 mile round trip. For those of you born in the 1970s and beyond, the lunar module and the Saturn V booster only made half of the trip or less. Apollo 13 was the exception as the lunar module was retained and used on the return leg to get the astronauts (along with the crippled service module) safely back to Earth. Lacking today's technology, it's a certainty that NASA used sophisticated one-off computing devices and radio equipment to fly manned and unmanned missions in the early days of the space program. In the spirit of 1969 technology, I'm going to show you how to wirelessly transfer data between multiple nodes without resorting to ZigBee, 802.15.4, specialized computers, or one-off radios.

In this installment of Design Cycle, we're going to use a modular approach to construct some embedded data radio hardware. Once the hardware comes online, we'll put on our software hats and spin some code. When we finish the coding, we'll dump the bits into the hardware, put on our pointy hat that is decorated with stars and moons, and observe data magically move from one radio platform to the other.

■ PHOTO 1. The ultra-compact size and low power consumption of the AIR module make it an ideal radio platform for low power wireless embedded projects.

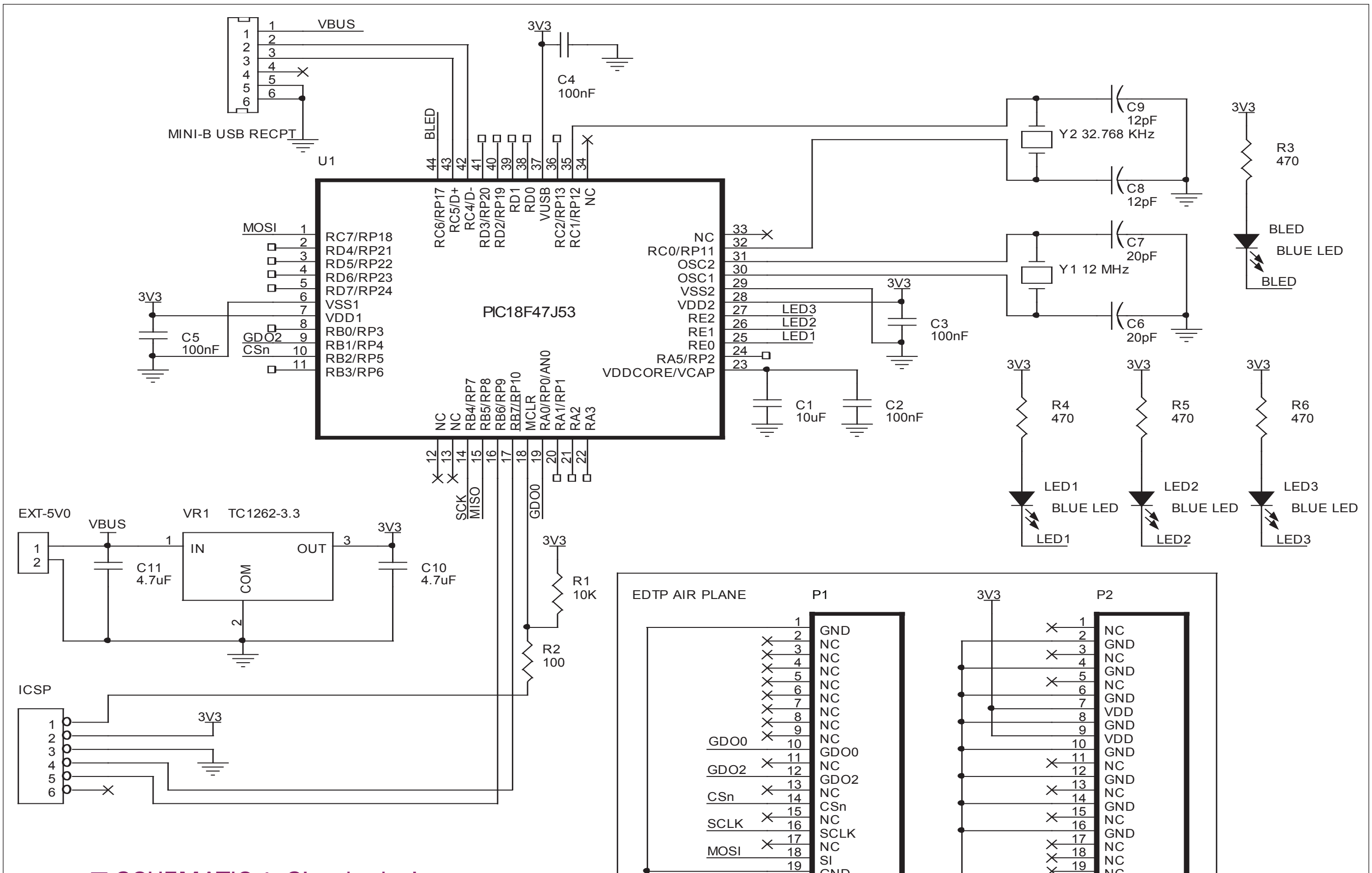
GETTING ON THE AIR

In this case, AIR is short for Anaren Integrated Radio. The 2.4 GHz AIR modules measure in at 9 x 12 x 2.5 mm. The sub-postal stamp-sized AIR module you see in **Photo 1** houses an integrated crystal, a voltage regulator, and all of the associated RF circuitry necessary to support its CC2500 transceiver core. In idle mode, the AIR module draws a paltry 1.5 mA. When sleeping, the 2.4 GHz module current requirement drops to a nearly nonexistent 400 nA. Anywhere from 13.3 mA to 19.6 mA current draw is typical in receive mode, and at maximum transmit power only 21.5 mA is consumed. This level of power consumption allows the AIR module to fit nicely into very low power telemetry and embedded control applications.

The 2.4 GHz A2500R24A AIR module pictured in **Photo 1** is equipped with an integral antenna. If your module is to be imprisoned in a metal enclosure, you'll need the A2500R24C variant which is fitted with a compact U.FL antenna connector. This month, we will work exclusively with the A2500R24A module. So, from now on the A2500R24A will be referred to simply as the AIR module. The AIR module is documented extensively on the Anaren website. So, there's no need to rehash in this text the information you can easily access online. The main goal this month is to design, assemble, and code a microcontroller-based support system for the AIR module.

AIR SUPPORT

Just because NASA sent men to the moon sans USB doesn't mean that we can't employ the services of USB in our AIR design process. In the firmware debug phase of the design process, we'll use USB as a power source and as a stand-in for RS-232. That implies that our microcontroller of choice must be USB capable. The next design point our microcontroller must meet concerns the compiler we will use to forge the AIR module firmware driver. We will need to select a compiler that supports USB. Access



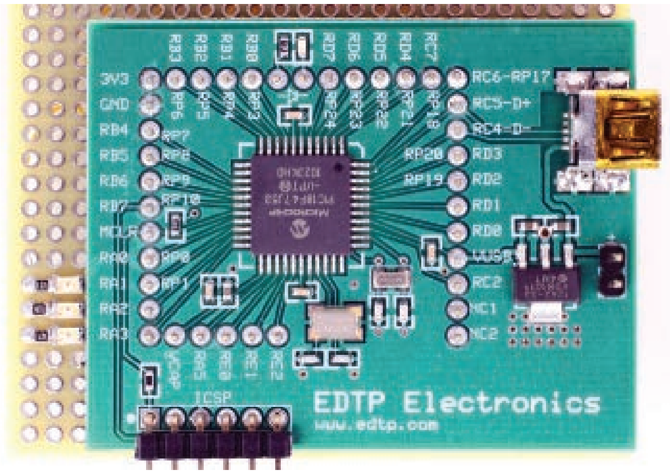
■ SCHEMATIC 1. Simple design, powerful application.

to the AIR module's internal registers is enabled via a four-wire SPI portal. The SPI protocol can be emulated with user-written bit bang routines. So, our compiler need doesn't built-in SPI functionality, but it would be nice if it did. We don't have to look far for a suitable microcontroller. The PIC18F47J53 natively supports USB and offers an on-chip hardware-based SPI engine. We'll be storing tables in Flash and data in buffers carved from SRAM. The PIC18F47J53 has ample memory resources that we can call on. On the power plane, the PIC18F47J53 is a perfect fit for the AIR module as both it and the PIC18F47J53 operate on a 3.3 volt power rail. The equality at the power plane level eliminates the need for logic level shifting of the PIC18F47J53's SPI portal and I/O pins. Thus, we can directly connect the I/O subsystems of the PIC18F47J53 and AIR module. I would like to use the CCS C compiler as it runs under the influence of MPLAB and allows the use of Microchip's PICkit3 as a debugger and programmer. As it turns out, the CCS C compiler also fully supports the PIC18F47J53's USB and SPI engines.

The PIC18F47J53 design is chronicled in **Schematic 1** and realized in **Photo 2**. A 32.768 kHz crystal is included in the PIC18F47J53 design to enable the PIC's internal RTCC (Real Time Clock Calendar). The option to employ the PIC18F47J53's analog-to-digital converter (ADC) is available by way of the free PORTA I/O pins. I've included some optional debug/status LEDs on the PORTE pins. If you need the PORTE pins as additional analog-to-digital inputs, you can move LED1-LED3 to other available PIC18F47J53 I/O pins or eliminate them all together. The PIC's USB portal doubles as a power source and a USB CDC (Communications Device Class) device. A companion CCS USB driver on the PC side sets up a virtual COM port that allows the PIC18F47J53 to communicate with a terminal emulator using its embedded USB portal.

THE AIR PLANE

Now that we have the microcontroller host portion of the project under control, we can begin work on the AIR frame. The AIR module grinning at you in **Photo 1** is an SMT device that is more suited to projects that have been tested and finalized. The Anaren engineers that wear those pointy hats brewed up yet another AIR module variant. The A2500R24A-EM1 was originally conceived to allow AIR modules to ride on Texas Instruments SmartRF evaluation boards. We're going to hijack the A2500R24A-EM1 tied up in **Photo 3** and put it on another plane. With a little help from our friends at SAMTEC and ExpressPCB, I whipped up the AIR PLANE which is basking in the light of **Photo 4**. The AIR PLANE is a hardware conversion tool that pulls the SAMTEC-based 40-pin A2500R24A-EM1 interface into eight pins that are placed on convenient 0.1 inch centers. Contained within the AIR PLANE's eight-pin interface are the four-wire SPI portal, the AIR module GDO0 and GDO2 I/O pins, and power and ground points. The AIR PLANE's converted interface is everything we need to fully access the AIR module's configuration and data registers.



■ PHOTO 2. The PIC18F47J53 hardware shown here is designed to drive an AIR module in stand-alone mode using battery power, or to control an AIR module under the influence of a PC's USB portal.

As you can see in **Photo 5**, all of the modular hardware components are mounted in the 0.1 inch pitch fiberglass grid of an EDTP plated-through perf board. The short ends of standard 0.1 inch pitch male headers are soldered on the component side of the PIC18F47J53 module. The extended portions of the male headers are long enough to pass through the perf board and act as wire wrap posts. I chose to use a female header to socket the AIR PLANE and its A2500R24A-EM1 evaluation board cargo.

GETTING AIRBORNE

I applied power to the collaboration of modules assembled in **Photo 5** and did not release any magic smoke. So, we're ready to put down a firmware foundation that will allow us to take our AIR module and supporting equipment down the runway, and ultimately go AIRborne. To those RF types that wear the pointy witch hats, the AIR module is a collection of well-placed coils and capacitors that transfers data by disrupting small portions of the Earth's magnetic field. To a hardware type, the AIR module is a tiny building block that sits on a particular layout of printed circuit board (PCB) pads. To a programmer, the AIR module is a logical collection of registers and FIFO (First In First Out) buffers. All of the RF plumbing has been done for us by the folks at Anaren. We took care of the AIR hardware build ourselves with the fabrication, assembly, and integration of the PIC18F47J53 and AIR PLANE modules. There's no one else here to fly this thing but us. So, let's start flipping software switches and see if we can't get this baby on the AIR.

A SOFTWAIR RADIO

My very first serious radio was a Knight Kit Ocean Hopper shortwave radio (**http://nostalgickitscentral.com/allied/products/knight_radio.html**). I recall the various coils that could be plugged in for listening in at different frequencies. The in-band tuning was done mechanically via a dial that was attached to a large variable capacitor. The only software involved in tuning the Ocean Hopper were my — at the time — itty bitty little fingers.

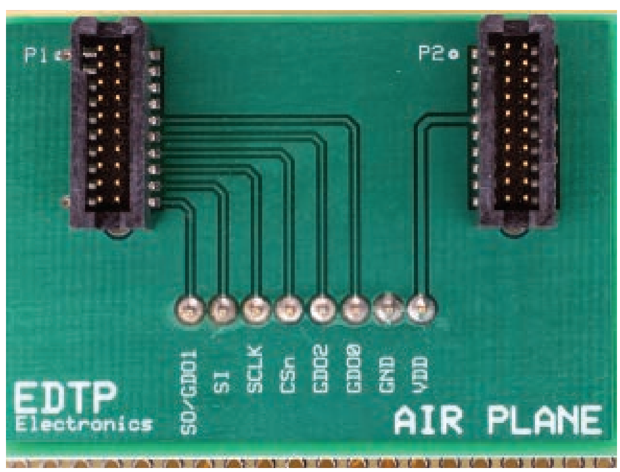
A couple of HC49-packaged microcontroller crystals would drown the tiny AIR module. Obviously, as far as the AIR module is concerned, there's not enough real estate available for any mechanical RF controls. Thus, the AIR module is a fly-by-wire device. Registers and the values contained within them replace the variable capacitors and frequency selection coils. In the case of the AIR module, all of its RF and data handling parameters are controlled by the contents of the 47 registers enumerated in **Listing 1**.

After staring a hole into the front panel of my Ocean Hopper, I longed for some visual feedback on the signals I was receiving in my headphones. Back in the day, the more sophisticated shortwave receivers came equipped with signal strength meters. Although a mechanical meter could be electrically adapted to the AIR module, an external mechanical or electronic metering device would be overkill as the AIR module has a built-in set of digital meters in the guise of status registers. The set of digital status meters are contained within the register set you see in **Listing 2**.

Listing 2 also exposes the AIR module's PATABLE and FIFO registers. The PATABLE consists of eight bytes and is instrumental in dialing in the AIR module's transmit output power. Our PATABLE setting looks like this:

```
//*********
//* AIR
//*PATABLE SET
//*FOR 0dBm
//*********
const unsigned
int8 AIR_PA_
TABLE[8]=
{
```

■ PHOTO 3. The A2500R24A-EM1 is the marriage of an A2500R24A AIR module and a Texas Instruments-inspired daughterboard.



■ PHOTO 4. The AIR PLANE eliminates the need to permanently mount an AIR module in the hardware/firmware development phase of the design cycle.

```
       0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
```

The transmit FIFO and receive FIFO handle the module's outgoing and incoming data, respectively. The AIR module manipulates the configuration register values and FIFO data using a state machine that runs within its CC2500 core. The host microcontroller can control the movement between states by issuing strobes. Strobes are really commands such as start receiving (SRX) or start transmitting (STX). The AIR module's available command strobes are contained within **Listing 3**. The AIR module's registers, FIFOs, and command strobes are all laid out for us. It's up to us to manipulate these resources in such a way as to cause the transmission and reception of digital data. So, let's get with it.

AIR TOOLS

We must initiate an AIR module RESET before any register manipulation can take place:

```
//************************************************
//* RESET AIR
//************************************************
void reset_air(void)
{
       DISABLE_SPI;
       output_bit(SCLK,1);
       output_bit(MOSI,1);
       output_bit(CSN,1);
       delay_ms(1);
       output_bit(CSN,0);
       delay_ms(1);
       output_bit(CSN,1);
       delay_ms(1);
       output_bit(CSN,0);
       while(input(MISO));
       ENABLE_SPI;
       data_out = AIR_SRES;
       write_data;
       DISABLE_SPI;
       while(input(MISO));
       output_bit(MOSI,0);
       output_bit(SCLK,0);
       output_bit(CSN,1);
       ENABLE_SPI;
}
```

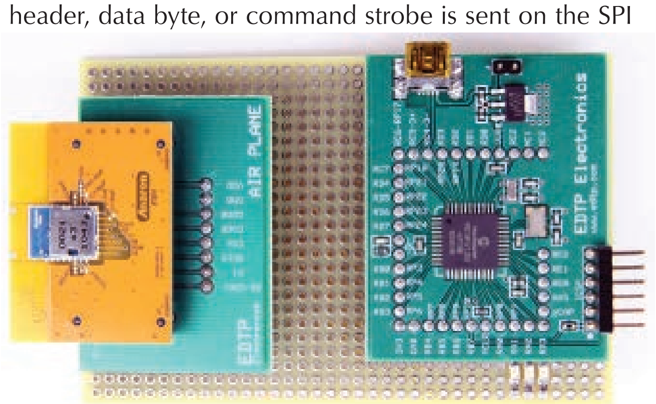
The first command strobe (AIR_SRES) is executed within the AIR module RESET function. Note that the SPI portal is alternately enabled and disabled in the *reset_air* function. The reason for this is that we must wait for MISO to fall logically low before engaging the AIR module via its SPI portal. Following the assertion of the CSn signal by the host microcontroller, the AIR module forces its SO pin logically low to indicate that its crystal oscillator is running. This logic low state on the AIR module's SO pin is identified as the CHIP_RDYn signal. You'll see this wait for CHIP_RDYn sequence often in the AIR support functions we will create. In that there are default values loaded into the AIR module's registers after reset, the logical starting point in our firmware generation process is to write a function to read the AIR module registers:

```
//************************************************
//* MACROS
//************************************************
#define NOP          delay_cycles(1)
#define LED_ON(led)  output_bit(led,0)
#define LED_OFF(led) output_bit(led,1)
#define enable_air   output_bit(CSN,0)
#define disable_air  output_bit(CSN,1)
#define xfer_data    data_in =
spi_read(data_out)
#define read_data    data_in = spi_read(0)
#define write_data   spi_write(data_out)
#define ENABLE_SPI   setup_spi(SPI_MASTER|SPI_
       L_TO_H|SPI_XMIT_L_TO_H|SPI_CLK_DIV_16);
#define DISABLE_SPI  setup_spi(SPI_DISABLED)
//************************************************
//* READ AIR REGISTER
//************************************************
void read_air_reg(unsigned int8 baddr)
{
       DISABLE_SPI;
       enable_air;

       while(input(MISO));
       ENABLE_SPI;
       data_out = baddr | 0x80;
       write_data;
       if(spi_data_is_in())
       {
              read_data;
       }
       disable_air;
}
```

I've posted the macro definitions here for clarity. The *read_air_reg* function disables the microcontroller's SPI portal to allow the microcontroller's MISO pin to act as a digital input to detect the CHIP_RDYn signal. Once MISO goes low, the SPI portal is reactivated, the desired register address with the read bit enabled (baddr | 0x80) is transferred to the AIR module, and the contents of the addressed register are returned to the microcontroller. Now that we have a method to read the AIR module registers, we can build a write function:

```
//************************************************
//* WRITE AIR REGISTER
//************************************************
void write_air_reg(unsigned int8 baddr,unsigned
int8 bdata)
{
       DISABLE_SPI;
       enable_air;
       while(input(MISO));
       ENABLE_SPI;
       data_out = baddr;
       xfer_data;

       status_byte = data_in;
       data_out = bdata;
       xfer_data;
       disable_air;
}
```

Note that along with writing the data, we simultaneously obtain a status byte from the AIR module. Anytime a header, data byte, or command strobe is sent on the SPI



■ PHOTO 5. The PIC18F47J53 module, the AIR PLANE, and its evaluation board cargo are all fitted on an EDTP plated-through perf board. The electrical connections are made with point-to-point solder techniques and wire wrap.

```
//*************************************************
//* AIR CONFIGURATION REGISTERS
//*************************************************
#define AIR_IOCFG2    0x00    // GDO2 output pin
                              // configuration
#define AIR_IOCFG1    0x01    // GDO1 output pin
                              // configuration
#define AIR_IOCFG0    0x02    // GDO0 output pin
                              // configuration
#define AIR_FIFOTHR   0x03    // RX FIFO and TX
                              // FIFO thresholds
#define AIR_SYNC1     0x04    // Sync word, high
                              // byte
#define AIR_SYNC0     0x05    // Sync word, low
                              // byte
#define AIR_PKTLEN    0x06    // Packet length
#define AIR_PKTCTRL1  0x07    // Packet automation
                              // control
#define AIR_PKTCTRL0  0x08    // Packet automation
                              // control
#define AIR_ADDR      0x09    // Device address
#define AIR_CHANNR    0x0A    // Channel number
#define AIR_FSCTRL1   0x0B    // Frequency
                            // synthesizer control
#define AIR_FSCTRL0   0x0C    // Frequency
                            // synthesizer control
#define AIR_FREQ2     0x0D    // Frequency control
                              // word, high byte
#define AIR_FREQ1     0x0E    // Frequency control
                              // word, middle byte
#define AIR_FREQ0     0x0F    // Frequency control
                              // word, low byte
#define AIR_MDMCFG4   0x10    // Modem
                              // configuration
#define AIR_MDMCFG3   0x11    // Modem
                              // configuration
#define AIR_MDMCFG2   0x12    // Modem
                              // configuration
#define AIR_MDMCFG1   0x13    // Modem
                              // configuration
#define AIR_MDMCFG0   0x14    // Modem
                              // configuration
#define AIR_DEVIATN   0x15    // Modem deviation
                              // setting
#define AIR_MCSM2     0x16    // Main Radio Cntrl
                           // State Machine config
#define AIR_MCSM1     0x17    // Main Radio Cntrl
                           // State Machine config
#define AIR_MCSM0     0x18    // Main Radio Cntrl
                           // State Machine config
#define AIR_FOCCFG    0x19    // Frequency Offset
                           // Compensation config
#define AIR_BSCFG     0x1A    // Bit
                      // Synchronization configuration
#define AIR_AGCCTRL2 0x1B     // AGC control
#define AIR_AGCCTRL1 0x1C     // AGC control
#define AIR_AGCCTRL0 0x1D     // AGC control
#define AIR_WOREVT1   0x1E    // High byte Event 0
                              // timeout
#define AIR_WOREVT0   0x1F    // Low byte Event 0
                              // timeout
#define AIR_WORCTRL   0x20    // Wake On Radio
                              // control
#define AIR_FREND1    0x21    // Front end RX
                              // configuration
#define AIR_FREND0    0x22    // Front end TX
                              // configuration
#define AIR_FSCAL3    0x23    // Frequency
                         // synthesizer calibration
#define AIR_FSCAL2    0x24    // Frequency
                         // synthesizer calibration
#define AIR_FSCAL1    0x25    // Frequency
                         // synthesizer calibration
#define AIR_FSCAL0    0x26    // Frequency
                         // synthesizer calibration
#define AIR_RCCTRL1   0x27    // RC oscillator
                              // configuration
#define AIR_RCCTRL0   0x28    // RC oscillator
                              // configuration
#define AIR_FSTEST    0x29    // Frequency
                         // synthesizer cal control
#define AIR_PTEST     0x2A    // Production test
#define AIR_AGCTEST   0x2B    // AGC test
#define AIR_TEST2     0x2C    // Various test
                              // settings
#define AIR_TEST1     0x2D    // Various test
                              // settings
#define AIR_TEST0     0x2E    // Various test
                              // settings
```

■ **LISTING 1.** These 47 registers can be considered as the AIR module's knobs. All of the AIR module's RF and data handling parameters are controlled by the values within these registers.

portal a status byte is returned by the AIR module on it's SO line. The layout of the status byte is laid out in **Figure 1**. Let's try out our new *write_air_reg* function:

```
init();
write_air_reg(AIR_PTEST,0x7F);
```

The *init* function has configured the PIC18F47J53 hardware, reset the AIR module, loaded the AIR module configuration registers and PATABLE, and placed the AIR module in the IDLE state. The transmit and receive FIFOs are also cleared during the initialization. The write to the AIR_PTEST register should return a status byte informing us that the CHIP_RDYn signal is logically low, the current AIR module state is IDLE, and there are more than 15 bytes free in the FIFOs. **Screenshot 1** captures the contents of an MPLAB Watch window that contains the returned status byte value. Using **Figure 1** to decode the value of the status byte verifies my predictions. We need not change the value of a register or issue a meaningful command strobe to obtain a status byte. We can also trigger the issuance of a status byte by issuing a NOP (No Operation) command strobe:

```
init();
send_strobe(AIR_SNOP);
```

Sending command strobes is also essential to getting on the AIR. Just like the read and write register functions, there's no rocket science behind the *send_strobe* function. Here's what the send_strobe function code looks like:

```
//*************************************************
//* SEND AIR STROBE
//*************************************************
void send_strobe(unsigned int8 bstrobe)
{
        DISABLE_SPI;
        enable_air;

        while(input(MISO));
        ENABLE_SPI;
        data_out = bstrobe;
        xfer_data;
        status_byte = data_in;
        disable_air;
}
```

The *xfer_data* macro — which gleans a status byte — is based on a built-in SPI function of the CCS C compiler.

AIR TRANSIT

I think we're ready to fly some bits around. Let's code up a transmitter. Here's what needs to happen:

```
#ifdef TRANSMITTER
            if(++usbcntr > 40000)
            {
                    AIR_idle_mode;
                    AIR_clear_tx_fifo;
                    build_tx_pkt(10,0x22);
                    send_pkt(payloadlen + 1);
```

```
//***********************************************
//* AIR STATUS REGISTERS
//***********************************************
#define AIR_PARTNUM   0x30    // Part number
#define AIR_VERSION   0x31    // Current version
                              // number
#define AIR_FREQEST   0x32    // Frequency offset
                              // estimate
#define AIR_LQI       0x33    // Demodulator
                     // estimate for link quality
#define AIR_RSSI      0x34    // Received signal
                     // strength indication
#define AIR_MARCSTATE0x35     // Control state
                              // machine state
#define AIR_WORTIME1  0x36    // High byte of WOR
                              // timer
#define AIR_WORTIME0  0x37    // Low byte of WOR
                              // timer
#define AIR_PKTSTATUS0x38     // Current GDOx
                      // status and packet status
#define AIR_VCO_VC_DAC       0x39    // Current
                     // setting from PLL cal module
#define AIR_TXBYTES   0x3A    // Underflow and #
                              // of bytes in TXFIFO
#define AIR_RXBYTES   0x3B    // Overflow and # of
                              // bytes in RXFIFO
//***********************************************
//* AIR PATABLE REGISTER
//***********************************************
#define PATABLE       0x3E
//***********************************************
//* AIR FIFO REGISTER
//***********************************************
#define TXFIFO        0x3F
#define RXFIFO        0x3F
```

■ **LISTING 2.** The AIR module status registers contain information on everything from the version of the core to the number of bytes in the receive and transmit queues. Oh yeah, the signal strength (RSSI) can also be found in the status register area.

```
                    output_toggle(BLED);
                    usbcntr = 0;
             }
#endif
```

The *init* function is identical for both the transmitter and receiver, and will not complete until the PIC18F47J53 is enumerated and goes online with the PC's USB portal. The CCS C compiler's *usbtask* function must be called periodically; the *usbtask* function call is included in the endless *do loop* that makes up the *main* function. I arbitrarily chose the name *usbcntr* for the 16-bit memory location that holds the number of cycles through the transmitter routine. The *usbcntr* value determines when to blink the blue LED. The self-explanatory AIR_idle_mode and AIR_clear_tx_fifo are command strobes in the form of macros. With the AIR module idling and the transmit FIFO cleared, we have indicated that we want to build a packet that is 10 bytes in length and send it to a receiver with the address of 0x22. To do this, some ground work must be laid first. In the *init* function, we loaded the AIR module's configuration registers with a modified template of values obtained from the Anaren website. The original set of AIR module configuration values called *original-config-values.h* is part of the article download package. If you compare the original set of configuration values with our modified configuration values in *air_rf_settings.h*, you'll see that we modified the AIR_PKTCTRL1 register. AIR_PKTCTRL1's original value was 0x04 which appends the RSSI and LQI bytes to the end of our packet. Changing the AIR_PKTCTRL1 value to 0x07 adds a packet address check to the packet's appended RSSI and LQI bytes. Now that address checking is in effect, we need to specify a receiver address in the AIR_ADDR configuration register. According to our *build_tx_pkt* function, the receiver's address should be 0x22 and that is reflected in the receiver's *air_rf_settings.h* file. Let's flesh out the *build_tx_pkt* function:

```
//***********************************************
//* BUILD TX PACKET
//***********************************************
void build_tx_pkt(unsigned int8 len,unsigned
int8 addr)
{
       unsigned int8 i;
       payloadlen = len;

       tx_buf[0] = payloadlen;
       tx_buf[1] = addr;
       for(i=2;i<payloadlen+1;++i)
       {

              tx_buf[i] = i;
       }
}
```

In variable length packet mode (AIR_PKCTRL least significant 2 bits = 01), the length of the packet is the first byte in the packet, and the address byte is next followed by the data. Once the packet is assembled, we can send it:

```
//***********************************************
//* SEND PACKET MANUAL
```

```
//***********************************************
//* AIR STROBES
//***********************************************
#define AIR_SRES      0x30     // Reset chip.
#define AIR_SFSTXON   0x31    // Enable and
       // calibrate frequency synthesizer (if
       // MCSM0.FS_AUTOCAL=1).
       // If in RX/TX: Go to a wait state where
       // only the synthesizer is running (for
       // quick RX / TX turnaround).
#define AIR_SXOFF     0x32    // Turn off crystal
                              // oscillator.
#define AIR_SCAL      0x33    // Calibrate
       // frequency synthesizer and turn it
       // off(enables quick start).
#define AIR_SRX       0x34    // Enable RX.
       // Perform calibration first if coming
       // from IDLE and MCSM0.FS_AUTOCAL=1.
#define AIR_STX       0x35     // In IDLE state:
       // Enable TX. Perform calibration first
       // if MCSM0.FS_AUTOCAL=1. If in RX state
       // and CCA is enabled: Only go to TX if
       // channel is clear.
#define AIR_SIDLE     0x36    // Exit RX / TX,
       // turn off frequency synthesizer and
       // exit Wake-On-Radio mode if applicable.
#define AIR_SAFC      0x37    // Perform AFC
       // adjustment of the frequency
       // synthesizer
#define AIR_SWOR      0x38    // Start automatic
       // RX polling sequence (Wake-on-Radio)
#define AIR_SPWD      0x39    // Enter power down
       // mode when CSn goes high.
#define AIR_SFRX      0x3A    // Flush the RX FIFO
                              // buffer.
#define AIR_SFTX      0x3B    // Flush the TX FIFO
                              // buffer.
#define AIR_SWORRST   0x3C    // Reset real time
                              // clock.
#define AIR_SNOP      0x3D    // No operation. May
       // be used to pad strobe commands to two
       // bytes for simpler software.
```

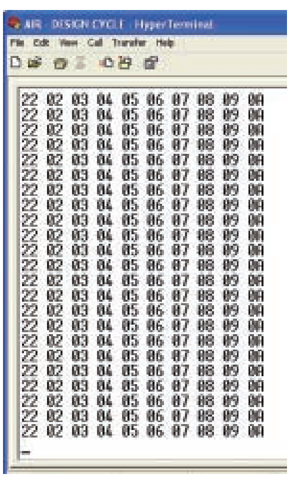
■ **LISTING 3.** Command strobes are issued by the host microcontroller to traverse the AIR module's internal state machine.

Bits	Name	Description		
7	CHIP_RDYn	Stays high until power and crystal have stabilized. Should always be low when using the SPI interface.		
6:4	STATE[2:0]	Indicates the current main state machine mode		
		Value	State	Description
		000	IDLE	Idle state (Also reported for some transitional states instead of SETTLING or CALIBRATE)
		001	RX	Receive mode
		010	TX	Transmit mode
		011	FSTXON	Frequency synthesizer is on, ready to start transmitting
		100	CALIBRATE	Frequency synthesizer calibration is running
		101	SETTLING	PLL is settling
		110	RXFIFO_OVERFLOW	RX FIFO has overflowed. Read out any useful data, then flush the FIFO with SFRX
		111	TXFIFO_UNDERFLOW	TX FIFO has underflowed. Acknowledge with SFTX
3:0	FIFO_BYTES_AVAILABLE[3:0]	The number of bytes available in the RX FIFO or free bytes in the TX FIFO		

■ FIGURE 1. The data contained within the status byte comes in handy when you need to quickly assess the status of the state machine and FIFOs.

```
//*************************************************
void send_pkt(unsigned int8 pktsize)
{
        unsigned int8 i;

        DISABLE_SPI;
        enable_air;
        while(input(MISO));
        ENABLE_SPI;
        data_out = TXFIFO + 0x40;
        write_data;
        for(i=0;i<pktsize;i++)
        {
                spi_write(tx_buf[i]);

        }
        disable_air;
        AIR_transmit_mode;
        while(input(GDO0)==0);
        while(input(GDO0));
        AIR_idle_mode;
}
```

The 0x40 added to the TXFIFO address allows us to burst write to the transmit FIFO. Burst writing/reading allows us to drop the CSn signal logically low and continually stream data until we raise the CSn signal which signals the end of the data burst. We can apply the same TXFIFO bursting logic to RXFIFO bursting. Take a look at the code that makes up a bursting RXFIFO read:



■ SCREENSHOT 2. A 10-byte packet delivered as ordered to the receiver at address 0x22. What you don't see here is the length byte we read and didn't write to the receive buffer, and a CRC at the end of the packet.

```
//*************************
//* READ RXFIFO
//*************************

void read_rxfifo(unsigned
int8 *buf,unsigned int8
len)
{
unsigned int i;
DISABLE_SPI;
enable_air;

while(input(MISO));
ENABLE_SPI;
data_out = RXFIFO | 0xC0;
write_data;
for(i = 0;i < len;i++)
{
        buf[i] = spi_read(0);
}
disable_air;
}
```

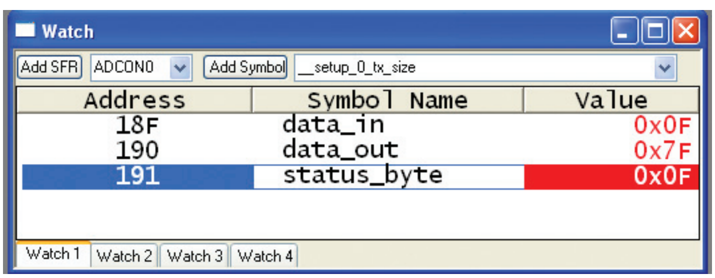
The 0xC0 that is ORed to the RXFIFO address tells the module to burst the read operation. Data is streamed from the RXFIFO to the PIC18F47J53 until the CSn line is returned logically high. Here's my idea of how to receive data sent by our TRANSMITTER code:

```
#ifdef RECEIVER
        AIR_receive_mode;     //enter receive mode
        while(input(GDO0)==0);
        while(input(GDO0));
        AIR_idle_mode;        //packet received

        read_air_reg(RXFIFO);//get length byte
        rx_len = data_in;
        read_rxfifo(rx_buf,rx_len);
        //read address byte and data
        read_rxfifo(rf_stats,2);
        //read LQI and RSSI
        for(i=0;i<rx_len;i++)
        {
           printf(usb_cdc_putc, "%X
           \r\n",rx_buf[i]);
        }
        output_toggle(BLED);
#endif
```

After entering receive mode and receiving a packet — which is signaled by the toggling of the AIR module's GDO0 I/O pin — we place the AIR module in IDLE mode and read the length byte of the received packet. The amount of data specified by the length byte is burst read into the receive buffer rx_buf. The appended RSSI and LQI bytes are bursted into the rf_stats array. If all went as planned, we should be able to transmit the contents of the receive buffer to a HyperTerminal session via the PIC18F47J53's USB portal.

A BREATH OF FRESH AIR

I'll leave you with **Screenshot 2**. The AIR Tools source code is included in its entirety within the download package that accompanies this edition of Design Cycle. I'll also include the AIR PLANE ExpressPCB layout file for those of you that want to scratch-build your own AIR craft. You're allowed to wear that pointy hat adorned with stars and moons as AIR is now in your Design Cycle. **NV**



■ SCREENSHOT 1. A read operation returns the number of free bytes in the receive FIFO. Conversely, a write operation returns the number of free bytes in the transmit FIFO. When the FIFO_BYTES_AVAILABLE is equal to 15, 15 or more bytes are available.

Anaren
AIR A2500R24x Modules
AIR A2500R24A-EM1
www.anaren.com

Custom Computer Services
CCS C Compiler
www.ccsinfo.com

Microchip
PIC18F47J53 Microcontroller
www.microchip.com

■ BY LOUIS E. FRENZEL W5LEF

BLUETOOTH: FORGOTTEN WIRELESS TECHNOLOGY OR JUST TAKEN FOR GRANTED?

New Versions and Applications Keep This Legendary Technology Alive and Well

You have no doubt heard of Bluetooth — the short-range wireless technology. It has been around for over a decade now. Yet it seems to be in the background these days with all the attention focused on the 3G/4G cellular technologies like Long Term Evolution (LTE) and, of course, the ever present Wi-Fi. Yet, Bluetooth is alive and well, and continuous progress is being made in new standards and products. Here is an update on what may really be the most widely used wireless technique today.

BLUETOOTH BASICS: IN CASE YOU FORGOT

Bluetooth is a short-range radio technology that operates in the unlicensed 2.4 GHz industrial-scientific-medical (ISM) band. It has to co-exist with Wi-Fi LANs, ZigBee networks, cordless phones, and microwave ovens to mention just a few services that also use this band. Bluetooth does pretty well with a range up to about 30 feet. The basic output power is 1 mW (0 dBm) but you can use two other power levels for longer ranges: 2.5 mW (4 dBm) and 100 mW (20 dBm). The 4 dBm version is the most popular, but the higher power version can reach out to about 100 meters under line-of-sight conditions.

Bluetooth has a raw data rate of 1 Mb/s, but some of that is overhead that goes to headers and error correction so the net data rate is 723 kb/s. The modulation is Gaussian FSK in a frequency hopping spread spectrum (FHSS) scheme. The carrier hops from one of the 79 channels to another in a random sequence. The hop rate is 1,600 hops per second for a dwell interval of 625 microseconds. During each hop interval, some data is transmitted. The FHSS method makes the data very secure. A more recent upgrade to Bluetooth version 2.1 is called Enhanced Data Rate (EDR) and uses π/4-DQPSK modulation that gives a data rate to 2.1 Mb/s. With 8DPSK modulation, the data rate tops out at 3 Mb/s. Version 3.0 (known as Bluetooth High Speed) uses the same transmission protocol but employs a Wi-Fi 802.11 radio link to get a data rate up to 24 Mb/s.

Version 3.0 was created as a way to let the Bluetooth protocol operate over a Wi-Fi connection if available. Wi-Fi is the wireless local area network (WLAN) standard of the IEEE, also known as 802.11. It is available in several versions (a, b, g and n). Bluetooth radios incorporate what is called a generic alternate MAC/PHY. MAC means media access control and PHY means physical layer. The MAC part of the radio defines how the data is packaged and transmitted, while the PHY defines the radio interface like RF or modulation. This alternate MAC/PHY is able to dynamically select either the Bluetooth radio or the Wi-Fi radio based on the need for faster transmission or not. This version does not seem to be very widely used.

A while back, an attempt was made to make ultra-wideband (UWB) an alternate PHY for Bluetooth. That would provide a high speed data option with a rate to 480 Mb/s over distances of less than 10 meters. The UWB

■ FIGURE 1. The Quatech Bluetooth wireless serial adapters. (A) The CS0400-479 is for RS-232 cable replacement. Just plug it into the DB9 connector. The antenna is built in. (B) The SS-BLT-400 can be used for RS-232/422/485 interfaces in industrial applications. Transmit range is up to 300 feet under clear path conditions for both units.

wireless standard was developed by the WiMedia Alliance and uses orthogonal frequency division multiplexing (OFDM) in the 3.1 to 10.7 GHz range. Its main use so far has been a very successful wireless USB cable replacement. The UWB project was never implemented, leaving Bluetooth without a high speed option. However, there does not seem to be any great pressure for it. Most Bluetooth applications are low speed anyway. There has also been some talk of using the emerging 60 GHz ISM band to provide an option to get data rates up to and over 1 Gb/s. There is not much call for it now. One of the more recent additions to Bluetooth is Version 4.0. It features lower power consumption and encrypted connections. A special version of Ver. 4.0 is called Bluetooth low energy. These Bluetooth radios use very little power and can run for months — even years — on a single coin cell. They will be great in portable applications.

PROFILES

The Bluetooth standard defines not only the radio interface described above but also a comprehensive data transmission protocol. The protocol details how the data is packaged into packets, transmitted, and received. It is a flexible protocol that can be adapted to almost any type of data. The protocol is implemented in a software stack of sequential procedures. While designers are free to create their own application on top of the Bluetooth protocol, it is a difficult process. To make Bluetooth more attractive to a wider range of applications, the Bluetooth Special Interest Group (SIG) has developed a wide range of special software packages called profiles that are designed to implement the most popular applications. The profiles define the applications.

All of these standards are created and managed by the Bluetooth SIG — an organization that develops the technology, establishes the standard, tests for interoperability and certification, and promotes the brand. It is a consortium of over 14,000 companies and their work is on-going. For more details on this, go to **www.bluetooth.org** or **www.bluetooth.com**. Bluetooth specifications and features are continuously updated.

BLUETOOTH HEADSETS

Among the profiles available are ones for cell phone headsets, wireless headphones, wireless printer to PC connections, human interface devices (keyboards, mouse, hand controllers, etc.), automotive hands-free kits, cordless phones, fax machine links, wireless speakers, and a whole bunch of others. The profiles speed up and simplify the application development and help ensure compatibility of similar products. One popular application is the use of Bluetooth in the hand controllers for the Nintendo Wii game. By far, the most widespread use of Bluetooth is in cell phone headsets and hands-free installations in cars and trucks. When I get in my car, my Apple iPhone automatically pairs (hooks up) with the hands-free stuff in my car. All I need to do to answer a call is press the button on the steering wheel and talk or listen. Very handy. No texting is allowed, of course.

Bluetooth headsets are also popular. A Bluetooth chip in the ear piece communicates with a Bluetooth chip in the cell phone. It works great. Some say it is the answer to the

WHAT'S NEXT IN *BLUETOOTH* TECHNOLOGY?

BLUETOOTH V4.0 PROFILE	PRODUCT AND USE ENABLED	EXPECTED PUBLISH DATE
Proximity Profile	Enables a device to lock/unlock a device (such as a PC) depending on the path loss to a peer device (typically a key fob) and enables devices to sound an alarm if the path loss between devices increase beyond a threshold or the devices move out of range of each other	21-Jun-11
Find Me Profile	Enables a device to locate a peer device within range	21-Jun-11
Link Loss Service	Enables a client device to configure an alarm that is issued when devices are out of range)	21-Jun-11
Immediate Alert Service	Enables a client device to activate notifications	21-Jun-11
Tx Power Service	Provides the transmit power level between devices	21-Jun-11
Time Profile	Enables devices to receive time, date and time zone information from another device (typically a phone or GPS device that has accurate time information)	26-Jul-11
Current Time Service	Enables devices to share time, date and time zone information with a client device	26-Jul-11
Next DST Change Service	Enables devices to share the time when next DST change occurs with a client device	26-Jul-11
Reference Time Update Service	Enables a device to request updated time information from another device	26-Jul-11
Phone Alert Status Profile and Service	Enables a watch or similar device to notify the user that the mobile phone is ringing, vibrating or displaying a message	26-Jul-11
Blood Pressure Profile and Service	Enables a blood pressure monitor to send measurement data to a mobile phone or computing device in an internationally recognized data format usable by caregivers and for health care records	26-Jul-11

■ FIGURE 2. The Bluetooth SIG plan for what is to come with the new low power version 4.0 standard. *Courtesy Bluetooth SIG.*

recent scare that cell phones give you brain cancer. There is no absolute proof yet, but many believe that holding a high power microwave device by your head for a while everyday can cause damage. In the meantime, if you are paranoid, just use a Bluetooth headset. No problem.

CABLE REPLACEMENT

One of the best applications for Bluetooth is cable replacement. A good example is replacing serial data connection cables in systems that use the popular RS-232 or RS-485 protocols. These low speed interfaces are still widely deployed in industrial and some commercial applications. Connections to older printers or industrial automated machines like programmable logic controllers are examples. You can buy Bluetooth radios that just plug right into the standard connectors and link up automatically. Products from Quatech, Inc., are good examples. **Figure 1A** shows their CS0400-479 serial adapter. It plugs right into a standard RS-232 nine-pin DB9 connector and gets its power from the system or an external supply if needed. It can implement any of the standard data speeds from 1,200 bps to 230 Kbps. The antenna is internal. **Figure 1B** shows the SS-BLT-400 serial adapter. It works on RS-232 or RS-485 interfaces. It uses a DB9 connector for RS-232 or screw terminals for the RS-422/485 connections. Data rates can be in the 1,200 bps to 115.2 kbps range. The antenna is external as you can see. Both of these devices are easy to use; just plug them in. The range can be up to 300 feet under clear path conditions.

LOW POWER BLUETOOTH

The newest Bluetooth version 4.0 or low power (LP) version targets anything mobile or portable including people. The goal is to create wearable wireless devices in watches, running shoes, and medical monitoring devices. Many of the newer Bluetooth chips contain the standard Bluetooth transceiver in addition to a low power transceiver. Two new LP profiles were recently announced: the heart rate profile and the temperature profile. Both are designed to enable the wireless monitoring of body functions.

As an example, a person can wear a temperature sensing patch that sends the signal to the LP Bluetooth radio. The Bluetooth radio then periodically (like every hour) transmits the data to a Bluetooth enabled cell phone and the data is sent to a doctor or someone monitoring the temperature. Dayton Industrial — a manufacturer of wireless monitors for sports and fitness — recently introduced a heart rate chest belt. It contains Nordic Semiconductor's μBlue nRF8001 low power Bluetooth chip in it. It allows for the setting up of new ways to collect, interpret, and display heart rate data for training purposes. Other similar applications are on the way.

Aside from the health and fitness areas, the Bluetooth 4.0 version will also find use in PC peripherals like mice and keyboards, or smart home monitoring and control. The table in **Figure 2** shows some of the forthcoming profiles.

BLUETOOTH DESERVES MORE RESPECT

According to the Bluetooth SIG, there are over three billion Bluetooth devices currently in use. In 2010 alone, 1.7 billion Bluetooth enabled devices were commercialized. How's that for being ubiquitous?

Bluetooth may have reached that point where it is just taken for granted. Certainly Bluetooth has to be the most widely deployed short-range wireless technology in the world. Bluetooth continues to age well and has lots of life left in it despite the lack of a really fast version.

Continuing evolution is ensured as the Bluetooth SIG keeps the new applications and updates coming. **NV**

ELECTRONET

For the ElectroNet online, go to **www.nutsvolts.com** click ***Electro-Net***

#37 SMILEY'S WORKSHOP

C PROGRAMMING - HARDWARE - PROJECTS

by Joe Pardue

Follow along with this series! Joe's book & kits are available at www.nutsvolts.com

avrtoolbox USART

Recap

We continue to build tools to go into our avrtoolbox. Last month, we looked at buffering data. This month, we'll use those buffers for serial communications while learning about the AVR USART. We will use the software engineering tools we've been discussing throughout the avrtoolbox series to design and build a USART library. If you ran the Serial Tester in Workshop 35 and the Ring Buffer Tester in Workshop 36, you've already used earlier versions of the USART library and didn't even know it. AND folks, that is the main reason why libraries are such a powerful software engineering tool — you can use them and not even know it!

Introduction

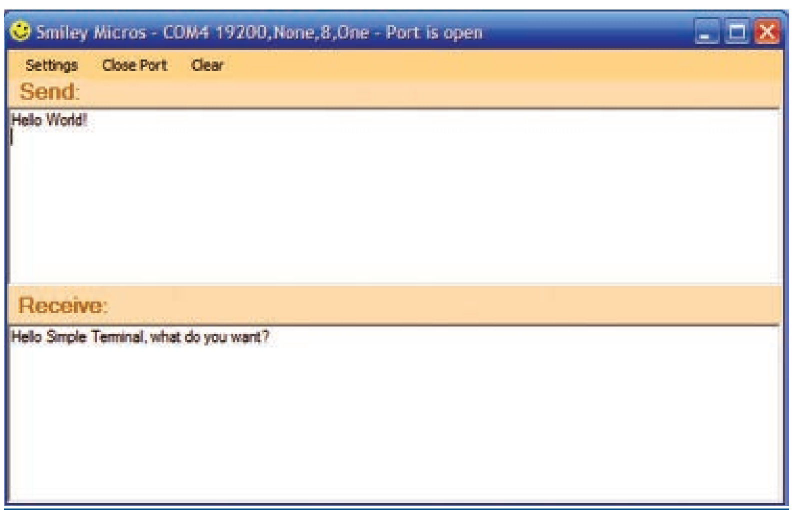
Much of what we've done in these Workshops takes for granted that we have functioning serial communications as a tool for doing our development work — whether we are blinking LEDs or developing a robotic coyote that will leap over our back fence and eat our neighbor's, yappy little poofy dogettes (okay, that was unkind dream fulfillment on my part). Anyway, somewhere in the background we've got a serial link between our AVR embedded system and a Windows PC, easing both development and use of our systems.

Ultimately, for an embedded system to be 'embedded' we will cut that link. However, for learning purposes, the serial port is our greatest friend. I personally consider the serial port to be the most powerful embedded systems development tool you can use. The communication channel between a PC and an embedded system can allow you to rapidly upload test programs via an AVR bootloader and you can sprinkle serial statements throughout your code with little messages like "Got to button_reader()" and "Leaving button_reader()." So, if you are running your code with a terminal program you might see "Got to button_reader()" but not "Leaving button_reader()." You'll then know that your program crapped out in the button_reader() function between the first and second statements.

I now do virtually all my debugging this way — bracketing suspicious blocks of code with messages until I've rooted out the guilty party.

Debugging without the serial port working properly is a bit like doing surgery in the dark. Oh sure, you can spend money and use simulators or emulators, or go really cheap and have the code blink LEDs, but if the design is going to have a serial port built into it anyway, why not use it for development? The only real drawback is that the code to use the serial port has to work properly. So, how do you debug serial code?

Well, I used the serial port to debug itself giving me the classic chicken or egg problem. I can't tell you the times I've sat cussing and looking at a blank terminal trying to get my serial code working properly. I must admit that I had to occasionally resort to blinking an LED to get things going. But the advantage you have is that I've got things going for you, so you don't have to re-plow that field. Let's learn some more about the USART that makes



■ FIGURE 1. A Simple Terminal.

■ FIGURE 2. An Arduino voltmeter.

these serial tools available.

We looked at the PC side of the two-sided serial link way back in Workshops 18, 19, and 20 (these are available at **www.nutsvolts.com**). In those articles, we learned how to write PC serial code in C# and Visual Basic .NET. We created the Simple Terminal shown in **Figure 1** and we even learned how to write a GUI for an Arduino-based voltmeter with a read-out on the PC as shown in **Figure 2**.

The PC at one time had a UART to do serial communications, but that is long gone in favor of USB. So now, we have to use a virtual serial port on the PC side of things. In fact, those three articles were excerpts from my book *Virtual Serial Port Cookbook* where you can learn how to use the FTDI FT232R to ease your communications between a PC and a microcontroller. (If you find this intriguing and want to help support your favorite technical magazine and writer, you can get the book and an educational parts kit from the *Nuts & Volts* website.)

A Very Brief History of Serial Communications

There are some seriously oddball things about USARTs and serial ports, mostly due to accidents of the history and the evolution of serial communications, so let's first take a look at that history and get a feel for the origins of some of the weirdness.

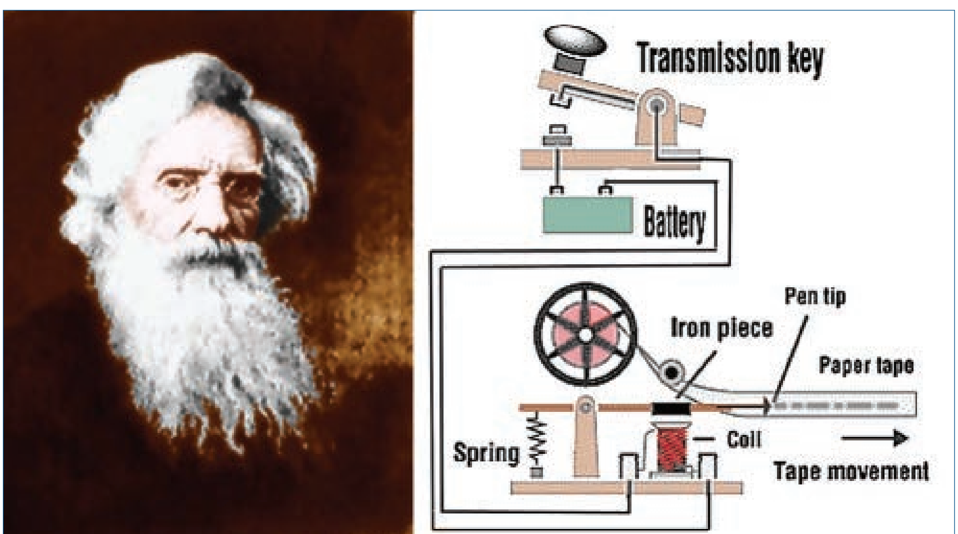
Why do we call the communication speed 'baud rate?' Why does a serial connector have a ring indicator pin? Why does a PC keyboard have a 'Ctrl' (control) key?

Frankly, a lot of the hardware and software terminology for serial communications seems weird when seen out of the context of how we arrived at today's serial communication techniques. This next section skims the surface of a large topic primarily to show how some of the terms we will be using came to be.

Samuel F.B. Morse

Morse (**Figure 3**) patented the telegraph in 1840. The name comes from the Greek tele = far away, and graphos = writing. True to its name, the original telegraph machine 'wrote' with dots and dashes on paper.

Morse's real invention was not the transmitter or receiver — which were based on devices that were being played with in electric laboratories of the time. His contribution was a binary code (**Figure 4**) that allowed characters to be sent as a serial stream of electric signals. He made the most commonly used characters (such as A, E, and T) into the simpler codes and the less commonly used (such as Q, X, and Z) into the more complex code.



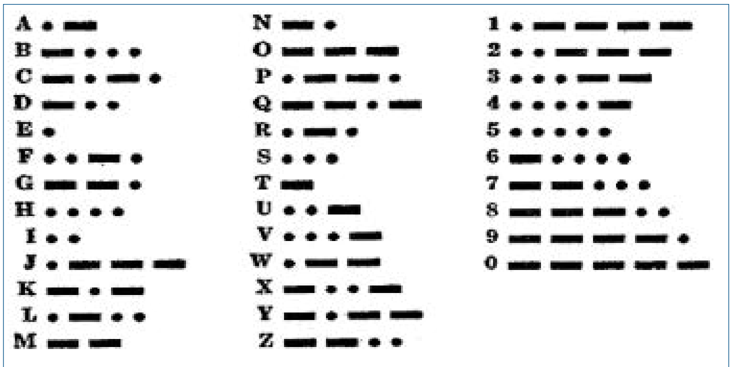
■ FIGURE 3. Morse self-portrait (yes, he was an artist) and his telegraph machine.

Emile Baudot

In 1874, Emile Baudot (**Figure 5**) invented a five-bit binary code that used a five-key transmitter. By using mechanical clockworks, the five bits were shifted out onto a single wire and used by the receiving station to print a character on paper. Five bits can uniquely encode 32 characters. Later modifications to Baudot's code changed the code to 26 character codes and 6 control codes. Two of the control codes were used to select either a 26 letter code or a 26 number/punctuation code table. The remaining four control codes were used for mechanical instructions to control the remote printer. With this new code, an operator could cause the remote printer to print 52 characters and could also control where on the paper the characters were printed.

If you wonder why this matters, look at your computer's keyboard and note the Ctrl key. It is used to alter the meaning of the rest of the keyboard in much the same way as was done by Baudot's apparatus, which is a direct ancestor of your keyboard. Also, when we transmit something we will use the ASCII code. We will see that there are a lot of atavistic printer control codes such as CR (Carriage Return), LF (Line Feed), and BEL (for bell, as in dingaling). We might wonder why we need such 'characters' in our attempts to send data between a PC and a microcontroller since neither has a carriage, a roller, or a dingalinger. Now you know.

Baud rate refers to the number of unique symbols that can be transmitted per second — the actual physical ability of the system to change states each second. There is often some confusion in the use of Bd (Baud) and bps (bits per second). The bps refers to the amount of information that can be transmitted each second. If each physical state change represents a bit of information, then Bd = bps. While this often is not the case, in our use where one state change represents one bit of data, we will use them interchangeably.



■ FIGURE 4. Morse's eponymous code.

Teletype Machines

Baudot's invention evolved into the teletype machine — an electro-mechanical typewriter that could act as both a transmitter and receiver of text messages over long distances.

In early computers, a teletype machine (**Figure 6**) was used to enter characters that were punched into cards or paper tape for loading programs into computers. [During WWII, the Colossus computer at Bletchley Park was used to crack encrypted German teletype messages.] Eventually, direct connections were developed to allow the teletype to function much like a PC keyboard.

Figure 7 shows Dennis Ritchie (inventor of the C programming language) standing next to Ken Thompson (inventor of Unix), designing the original Unix operating system at Bell Labs on a PDP-11 using a teletype machine to 'talk' to the computer.

RS-232 and Modems

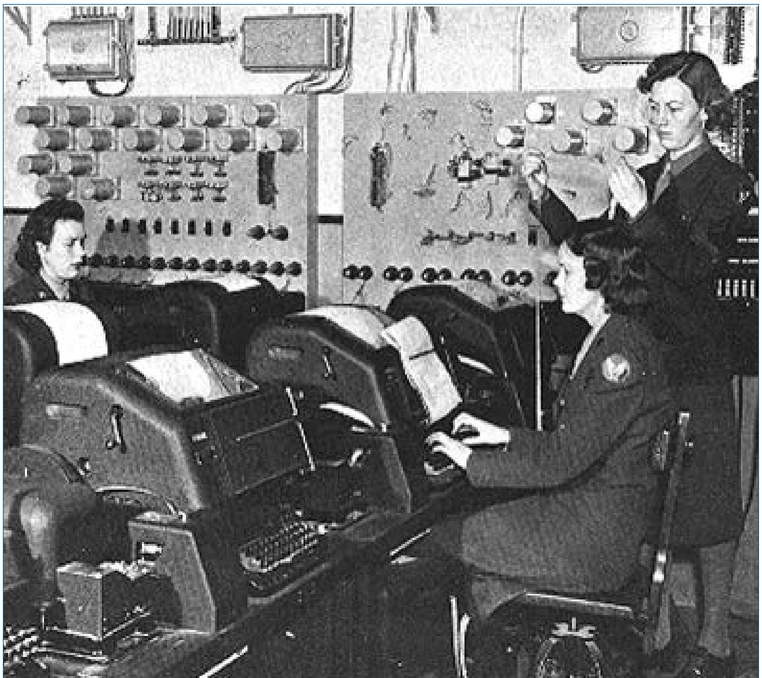
The telegraph and teletype machines used a binary (on or off) signal to transmit and receive data. However, the world was wired not for binary signals, but for telephones that sent analog signals (voice — 300 to 3,400 Hz). During the 1950s, folks figured out how to allow binary signals to be sent over these plain old telephone lines by acoustic frequency MOdulating and DEModulating of the signal. Thus, the name modem.

The RS-232 standard was written to allow teletype machines — which were referred to as DTE (Data Terminal Equipment) — to link to a modem — referred to as DCE (Data Communication Equipment) — that could then transmit the binary data from a teletype machine over a phone line to a distant computer. There were several iterations of this standard, but by 1969 RS-232C became *the* standard that would eventually be adopted (sort of) by Microsoft for the PC serial port. The 'sort of' is necessary since the PC isn't exactly 100% pure, but close enough that the PC serial port often became known as the RS-232 port.

The port has nine pins. Data is transmitted on the TxD pin and received on the RxD pin. Six additional pins are used to control the communications between the PC and the modem. They are the DCD, DSR, DTR, RTS, CTS, and RI, which we'll learn a lot more about later. The serial port was originally intended to be used with modems, but designers found that it could also be used to communicate with other peripheral devices such as mice, drawing pads, oscilloscopes, etc., thus leading to our more generic use for communicating with microcontrollers.

So now you know that we call the communication speed 'baud rate' after Emile Baudot; that a serial connector has a ring indicator pin because the original connector was meant to attach to a modem and use a

■ FIGURE 5. Emile Baudot pictured on a French phone card.



■ FIGURE 6. WAC teletype operators during World War II.

telephone line which 'rings' when a call comes in; that a PC keyboard has a 'Ctrl' (control) key because it evolved from Baudot's original keyboard; and that any other weird term you come across probably has a historic reason for being used. Just as a fun factoid of interest, the Arduino uses the DTR pin of the modem to reset the AVR so that you can use the bootloader to upload applications.

So, what does this have to do with USART?

Early in the development of digital systems, all the above arcania was put into a single IC peripheral that came to be known as a **UART**: **U**niversal **A**synchronous **R**eceiver **T**ransimitter. Like many useful peripheral ICs (memory, ADC, timers, etc.) the UART was soon put on the same silicon as a microprocessor, like the AVR core. The old-fashioned UART only did asynchronous (not occurring at predetermined or regular intervals – no clock) communication, but Atmel added some features that let theirs do some synchronous (does occur at predetermined regular intervals – clocked) things. They also added an S to the name that has confused folks mightily. We won't be using the S other than in the name. So, just get used to the fact that when we say USART we are actually dealing with the venerable UART. However, since this is an Atmel peripheral, we'll use their name for it (grudgingly). Clear? I didn't think so.

AVR USART Hardware

I suggest that you get the Atmel ATmega328 datasheet and take a look at the section on the USART. Yup, this beast is like most peripherals. It can do so many different things that the real hard part is figuring out the limited subset of things that you want it to do. We won't be using 90% of what you see in the datasheet, but the trick — as usual — is figuring out which 10% we want to use. There are things that it can do that I have never seen done in any system I've used — like allowing five, six, seven, eight, or nine data bits. I've only ever used eight bits and have only seen seven and nine on other systems (never five or six). Guess somebody somewhere in some historical context used these so they are there if you need them.

To simplify things, let's keep this as simple as possible by using only the most common modem parameters: eight data bits, no parity, one stop bit, and no handshaking. Yes, so the datasheet is hard. So let's try to simplify it a bit.

Set Up the USART

Using Aliases for Comprehension and Portability

We have already discussed a bit about using aliases for register and bit acronyms in earlier Workshops, but let's risk some repetition and revisit the concepts (anyway, repetition is supposed to help learning). We have two main problems with registers and the bits in them. First, there are hundreds of them, and second, they are named using acronyms making for a double whammy of "wazzat!?" when you look at their names. Quick — what is UCSR0C or UPM01?

You are likely to see both of these in some AVR source code, so you might expect that you are supposed to know what they mean. The UCSR0C is the USART Control and Status Register C for USART0, and the UPM01 is USART parity mode bit 1 for USART0. Now don't you feel like a real dummy for not knowing that? Well, I don't and personally I think folks who use the raw acronyms in their code are foolish. Wouldn't it be better if they had used an alias in the first place that spelled it out? For example say they used:

```
#define USART_CONTROL_STATUS_REG_C          UCSR0C
#define USART_PARITY_MODE_BIT_1             UPM01
```

Then, when they look at their code and see USART_CONTROL_STATUS_REG_C or USART_PARITY_MODE_BIT_1 they will have a much better chance of knowing what is going on without having to dig out the datasheet to find the acronyms. The compiler doesn't care because it doesn't see the long name anyway since the preprocessor substitutes the acronym. Of course, you do have the bother of all the extra typing and I do hate typing the '_' character, but I've found in the long run doing things this way is just adding more documentation for your code. But that is generally a good thing — especially for educational code — so that's what we will do.

Another example of why this is useful is that not all the AVRs use the same acronyms for the same register, and some have more than one USART. Using defines helps ease reusing code for multiple devices as shown here:

```
// Remove comment from only one device
//#define USART_ATMEGA169
#define USART0_ATMEGA328
//#define USART1_ATMEGA328
     …more devices…

#if defined(USART_ATMEGA169)
   #define USART_CONTROL_STATUS_REG_C     UCSRC
   #define USART_PARITY_MODE_BIT_1        UPM1
     …more aliases…
#elif defined(USART0_ATMEGA328)
   #define USART_CONTROL_STATUS_REG_C     UCSR0C
   #define USART_PARITY_MODE_BIT_1        UPM01
     …more aliases…
#elif defined(USART1_ATMEGA328)
   #define USART_CONTROL_STATUS_REG_C     UCSR1C
   #define USART_PARITY_MODE_BIT_1        UPM11
     …more aliases…
```

This allows you to use the same source code for each device, only changing one line to select the device. So, the code can have USART_CONTROL_STATUS_REG_C and the preprocessor gets to decide if that is UCSR0C or UCSR1C, depending on the #define you selected. Do this and give your library an indicative name like libavr_USART0_atmega328.a and away you go.

You'll find something like this in the avrtoolbox USART directory where we have libraries for the AVR Butterfly (ATmega169), the Arduino board (ATmega328), and the BeAVR (ATmega644).

Which Registers and Bits?

While I promised that we would only use a small subset of what the USART can do, we still have to set a bunch of registers and bits. To further simplify our life, let's just look at what we have aliased for the ATmega169 (AVR Butterfly) in USART.h:

```
// Registers
#define UART_BAUD_RATE_HIGH          UBRRH
#define UART_BAUD_RATE_LOW           UBRRL
#define UART_CONTROL_STATUS_REG_A    UCSRA
#define UART_CONTROL_STATUS_REG_B    UCSRB
#define UART_CONTROL_STATUS_REG_C    UCSRC

// Bits
#define UART_ENABLE_TRANSMITTER      TXEN
#define UART_ENABLE_RECEIVER         RXEN
#define UART_READY_TO_TRANSMIT       UDRE
#define UART_TRANSMIT_COMPLETE       TXC
#define UART_RECEIVE_COMPLETE        RXC
#define UART_DATA_REG                UDR
#define UART_STOP_BIT_SELECT         USBS
#define UART_CHARACTER_SIZE_0        UCSZ0
#define UART_CHARACTER_SIZE_1        UCSZ1
#define UART_CHARACTER_SIZE_2        UCSZ2
#define UART_MODE_SELECT             UMSEL
#define UART_DOUBLE_SPEED            U2X
#define UART_FRAME_ERROR             FE
#define UART_DATA_OVER_RUN           DOR
#define UART_PARITY_ERROR            UPE
#define UART_PARITY_MODE_0           UPM0
#define UART_PARITY_MODE_1           UPM1
#define UART_MULTI_PROCESSOR_
COMMUNICATION_MODE                   MPCM
#define UART_TX_COMPLETE_INTERRUPT_ENABLE TXCIE
#define UART_RX_COMPLETE_INTERRUPT_ENABLE RXCIE
   #define
UART_DATA_REGISTER_EMPTY_INTERRUPT_ENABLE UDRIE
#define UART_RX_DATA_BIT_8                RXB8
#define UART_TX_DATA_BIT_8                TXB8
```

That is five registers and 23 bits! The long form of the names are somewhat explanatory, but you may want to look at the datasheet to get the details if you are excessively curious or masochistic. You can see why we want to set this up one time, get it working, put it in a black box with lots of duct tape, and never look at it again.

How Do We Set These Registers and Bits?

We have two USART initialization functions: one initializes everything and the other relies on defaults for all but the baud rate. The comprehensive function has a list of parameters that it sets:

```
void USART0_init(uint32_t baud,\
 uint32_t freq_cpu,\
 USART_mode_t mode,\
 USART_databits_t databits,\
 USART_stopbits_t stopbits,\
 USART_parity_t parity)
```

In the function, we call various other functions or macros to set the parameters, such as:

```
USART0_mode(mode);
```

which is a macro:

```
#define USART0_mode(x) bit_write(x,
UART_CONTROL_STATUS_REG_C, BIT(UART_MODE_SELECT))
```

If you are interested in the details, you can view the rest in the avrtoolbox project source code by clicking: Source – Browse – Trunk – libavr – source – usart – usart0_init.c. This is complex and you can easily see why we want to set this up one time, get it working, put it in a black box with lots of duct tape, and never look at it again.

■ FIGURE 7. C being invented on a more recent teletype machine.

The Rest is in avrtoolbox

To conform to our avrtoolbox process, we need a Functional Requirements Specification, an Application Programmer Interface, a tester program, the source code, libraries for the AVR Butterfly (ATmega169), Arduino board (ATmega328), and BeAVR (ATmega328 on a breadboard). You will find all that in various states of completion in the avrtoolbox open source project at **http://code.google.com/p/avrtoolbox**. Just click on Source, then Browse | trunk | libavr.

You will see four subdirectories: doc, librarian, source, and testers. I'll leave it up to you to figure out what is in each.

Please be aware that this is a work in progress. Not everything is finished and none of it is perfect. In fact, there are typos and bugs everywhere. If you expect perfection, stay away. If, however, you want to be helpful, when you find a problem or have a question please put on your biohazard suit and start a thread on **www.avrfreaks.net** with the word 'avrtoolbox' in the title; I probably will see it. First, read my blog entry that will tell you why you need the biohazard suit: **http://smileymicros.com/blog/2011/01/24/using-an-internet-forum**.

Next month, we are going to take a look at the C standard library and apply what we've learned about ring buffers and USART libraries to get the venerable C printf() function working, along with some string manipulation functions and a command-line interpreter that will soon become one of your favorite development tools. **NV**

For a complete product detail visit our webstore!!

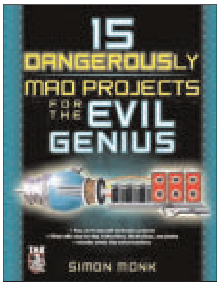
The Nuts & Volts WEBSTORE

ELECTRONICS

15 Dangerously Mad Projects for the Evil Genius
by Simon Monk

NEW!

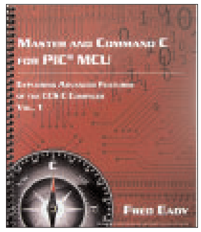
UNLEASH YOUR INNER MAD SCIENTIST!



This wickedly inventive guide explains how to design and build 15 fiendishly fun electronics projects. Filled with photos and illustrations, *15 Dangerously Mad Projects for the Evil Genius* includes step-by-step directions, as well as a construction primer for those who are new to electronics projects

$24.95

Master and Command C for PIC MCUs
by Fred Eady

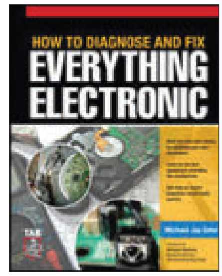


Master and Command C for PIC MCU, Volume 1 aims to help readers get the most out of the Custom Computer Services (CCS) C compiler for PIC microcontrollers. The author describes some basic compiler operations that will help programmers — particularly those new to the craft — create solid code that lends itself to easy debugging and testing. As Eady notes in his preface, a single built-in CCS compiler call (output_bit) can serve as a basic aid to let programmers know about the "health" of their PIC code.

$14.95

How to Diagnose and Fix Everything Electronic
by Michael Jay Geier

Master the Art of Electronics Repair!



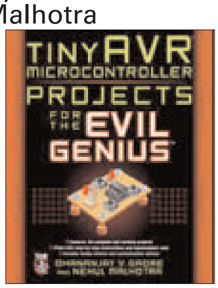
In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.

About the Author

Michael Jay Geier began operating a neighborhood electronics repair service at age eight that was profiled in *The Miami News*.

$24.95

tinyAVR Microcontroller Projects for the Evil Genius
by Dhananjay Gadre and Nehul Malhotra

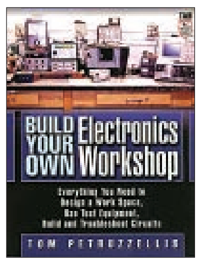


Using easy-to-find components and equipment, this hands-on guide helps you build a solid foundation in electronics and embedded programming while accomplishing useful — and slightly twisted — projects. Most of the projects have fascinating visual appeal in the form of large LED-based displays, and others feature a voice playback mechanism. Full source code and circuit files for each project are available for download.

$24.95

Build Your Own Electronics Workshop
by Thomas Petruzzellis

BUILD YOUR OWN DREAM ELECTRONICS LAB!



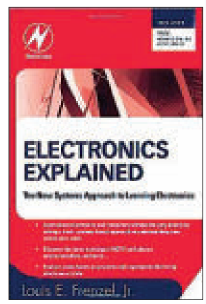
This value-packed resource provides everything needed to put together a fully functioning home electronics workshop! From finding space to stocking it with components to putting the shop into action -- building, testing, and troubleshooting systems. This great book has it all! And the best part is, it shows you how to build many pieces of equipment yourself and save money, big time!

Reg Price $29.95

Sale Price $24.95

Electronics Explained
by Louis Frenzel

The New Systems Approach to Learning Electronics



Don't spend time reading about theory, components, and old ham radios - that's history! Industry veteran, Louis Frenzel, gives you the real scoop on electronic product fundamentals as they are today. Rather than tearing electronics apart and looking at every little piece, the author takes a systems-level view. For example, you will not learn how to make a circuit but how a signal flows from one integrated circuit (IC) to the next, and so on to the ultimate goal.

$29.95

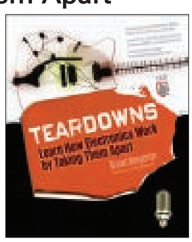
Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists
by Dustyn Roberts

In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95*

TEARDOWNS
by Bryan Bergeron

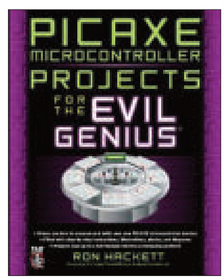
Learn How Electronics Work by Taking Them Apart



Amp up your knowledge of electronics by deconstructing common devices and analyzing the revealed components and circuitry. *Teardowns: Learn How Electronics Work by Taking Them Apart* contains 14 projects that expose the inner workings of household appliances, workbench measuring instruments, and musical equipment. Discover how resistors, capacitors, sensors, transducers, and transistors function in real circuitry.

Reg $24.95 **Sale Price $21.95**

PICAXE Microcontroller Projects for the Evil Genius
by Ron Hackett

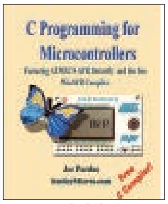


This wickedly inventive guide shows you how to program, build, and debug a variety of PICAXE microcontroller projects. *PICAXE Microcontroller Projects for the Evil Genius* gets you started with programming and I/O interfacing right away, and then shows you how to develop a master processor circuit.

$24.95

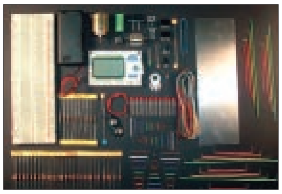
Order online @ www.nutsvolts.com
Or CALL 1-800-783-4624 today!

BOOK & KIT COMBOS



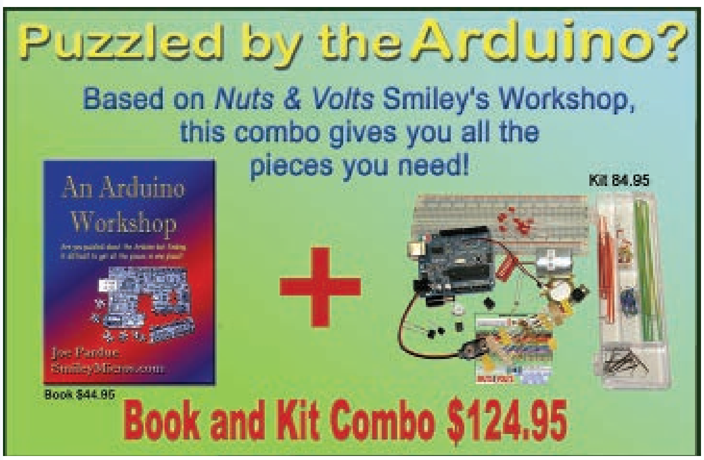
Book **$44.95**

From the Smiley Workshop C Programming for Microcontrollers
by Joe Pardue

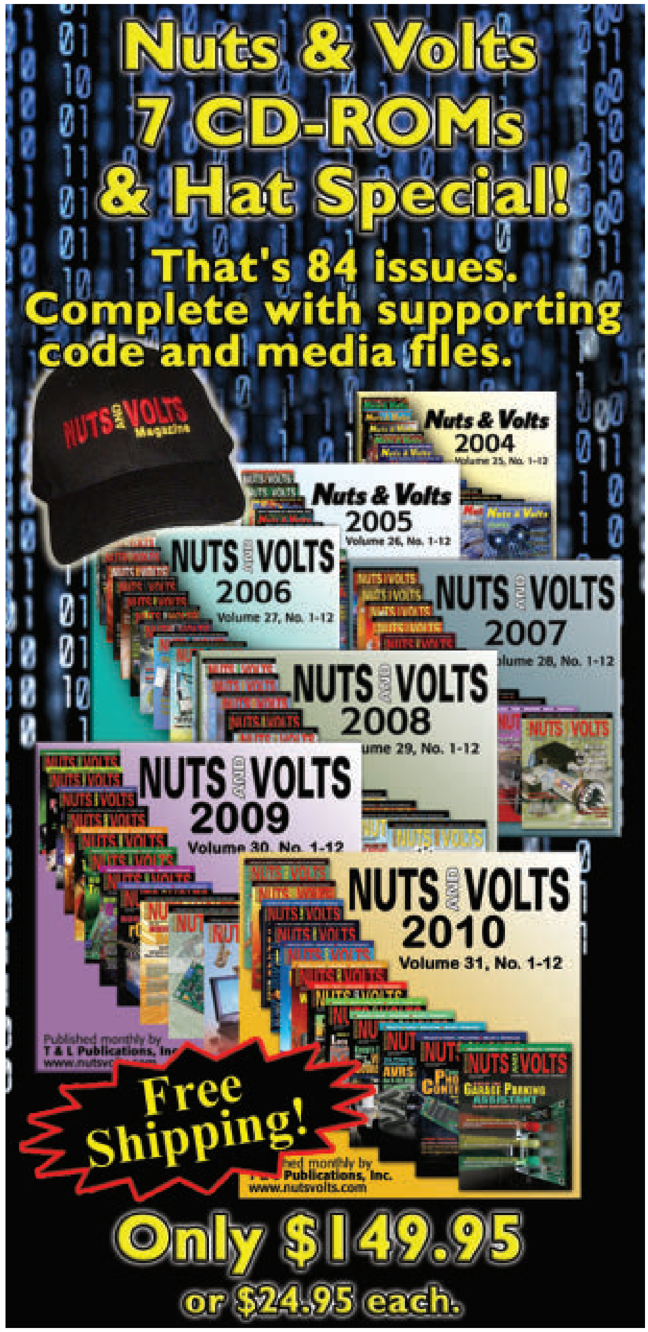


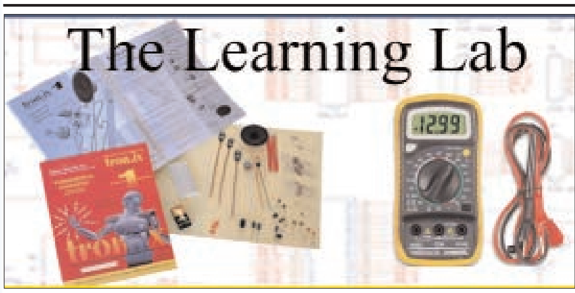
Kit **$66.95**

Do you want a low cost way to learn C programming for microcontrollers? This 300 page book and software CD show you how to use ATMEL's AVR Butterfly board and the FREE WinAVR C compiler to make a very inexpensive system for using C to develop microcontroller projects.
Combo Price $99.95 Plus S/H



CD-ROM SPECIALS





This lab — from the good people at GSS Tech Ed — will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

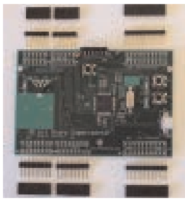
Reg Price $79.95 **Subscriber's Price $75.95**

This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants, and conversions. AND it still fits in your shirt pocket!

CALL 1-800-783-4624 today!
Or Order online @ www.nutsvolts.com

PROJECTS

32-Bit Micro Experimenter Board

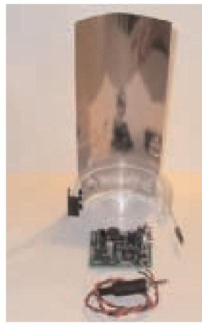


The new 32-Bit Micro Experimenter is the fastest way to learn 32-bit microcontrollers.

The kit includes onboard 46 programmable I/O and USB, free software, carefully documented step-by-step experiments for USB, embedded web server, graphics and audio, wireless, RTOS, and file I/O. User pushbuttons, LEDs, and 32 kHz clock crystal. Can be used in solderless breadboard environment or stand-alone. Also supports Arduino compatible interface.

Subscriber's Price **$89.95**
Non-Subscriber's Price **$93.95**

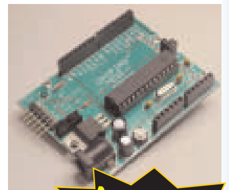
Piezoelectric Film Speaker Kit



As seen in the November 2010 issue, here is a great project to amaze your friends and to demonstrate a unique way of producing sound. Kit contains one piece of piezoelectric film, speaker film stand, PCB, components, audio input cable, and construction manual. All you'll need to add is a battery and a sound source.
For more info, please visit our website.

Subscriber's Price **$69.95**
Non-Subscriber's Price **$74.95**

CHIPINO Kit



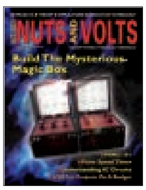
Assembled units $29.95!

The CHIPINO module is an electronic prototyping platform that is used in a series of articles starting with the March 2011 issue of *Nuts & Volts Magazine*.

Developed by the CHIPAXE Team as a bridge between PICs and Arduinos. The module was designed specifically to match the board outline, mounting holes, connector spacing, and most of the microcontroller I/O functions found on the popular Arduino module.

Subscriber's Price **$18.95**
Non-Subscriber's Price **$19.95**

Magic Box Kit

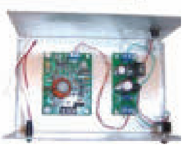


As seen on the April 2007 cover

This unique DIY construction project blends electronics technology with carefully planned handcraftsmanship. This clever trick has the observer remove one of six pawns while you are out of the room and upon re-entering you indicate the missing pawn without ever opening the box.
Includes an article reprint.

Subscriber's Price **$39.95**
Non-Subscriber's Price **$45.95**

Mini-Bench Supply Complete Kit

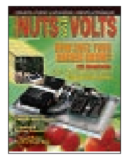


A small power supply with +5V, +12V, and -12V outputs is a handy thing to have around when you're breadboarding circuits with both op-amps and digital ICs.
Kit includes: Enclosure box, accessories, DC-to-DC converter kit, switching regulator kit, and article reprint.
For more information, please see the "feature article section" on the of the *Nuts & Volts* website.

Subscriber's Price **$76.95**
Non-Subscriber's Price **$83.95**

Sorting Counter Kit

NEW!



As seen in the July 2011 issue.

Sorting counters have many uses — keeping score, counting parts, counting people — it is just a handy gadget to have around. This is a very simple project for those who want to learn to solder or are interested in using microprocessors and how they function. No special tools are needed, just a small tip soldering iron. It has no box as it stands alone, therefore there is no drilling.

Subscriber's Price **$33.95**
Non-Subscriber's Price **$39.95**

16-Bit Experimenter Handbook

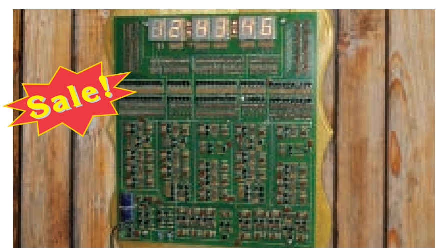
NEW!

From the series of articles by Tom Kibalo, The Complete 16-Bit Experimenter Handbook on CD-ROM!

This CD-ROM handbook contains: 200 pages of detailed instructions and 41 experiments with 'C' software examples. Also includes the experiments for the dsPIC33.
For a complete list of kits and combos, please visit our webstore.

Subscriber's Price **$64.95**
Non-Subscriber's Price **$69.95**

Transistor Clock Kit



Sale!

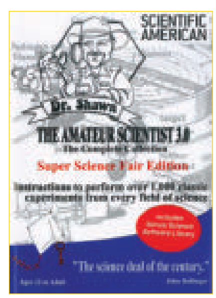
If you like electronic puzzles, then this kit is for you! There are no integrated circuits; all functionality is achieved using discrete transistor-diode logic. The PCB is 10"x11" and harbors more than 1,250 components! For more info, see the November 2009 issue.

Reg $225.95
Sale Price $199.95

PCBs can be bought separately.

The Amateur Scientist 4.0 The Complete Collection

by Bright Science, LLC



There are 1,000 projects on this **CD**, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do their first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!

Reg $26.95 Sale Price $23.95

WE ACCEPT VISA, MC, AMEX, and DISCOVER. Prices do not include shipping and may be subject to change.

FUNDAMENTALS FOR THE BEGINNER

Build a DC to DC Power Supply

This is an excellent circuit that you can use to power projects that require from zero to something less than nine volts.

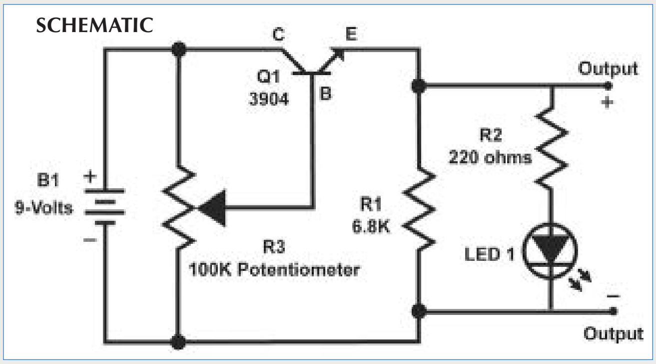
1. Build the Circuit.

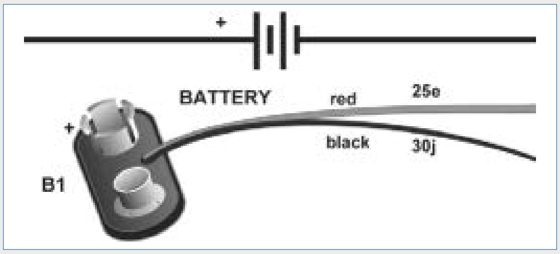
Using the schematic along with the pictorial diagram, place the components on a solderless breadboard as shown. Verify that your wiring is correct.

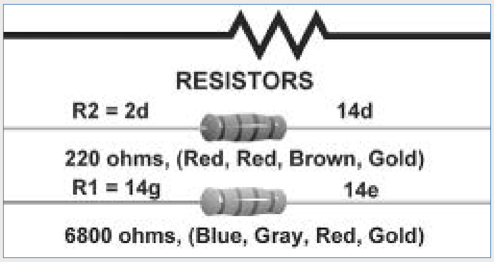
2. Do the Experiment.

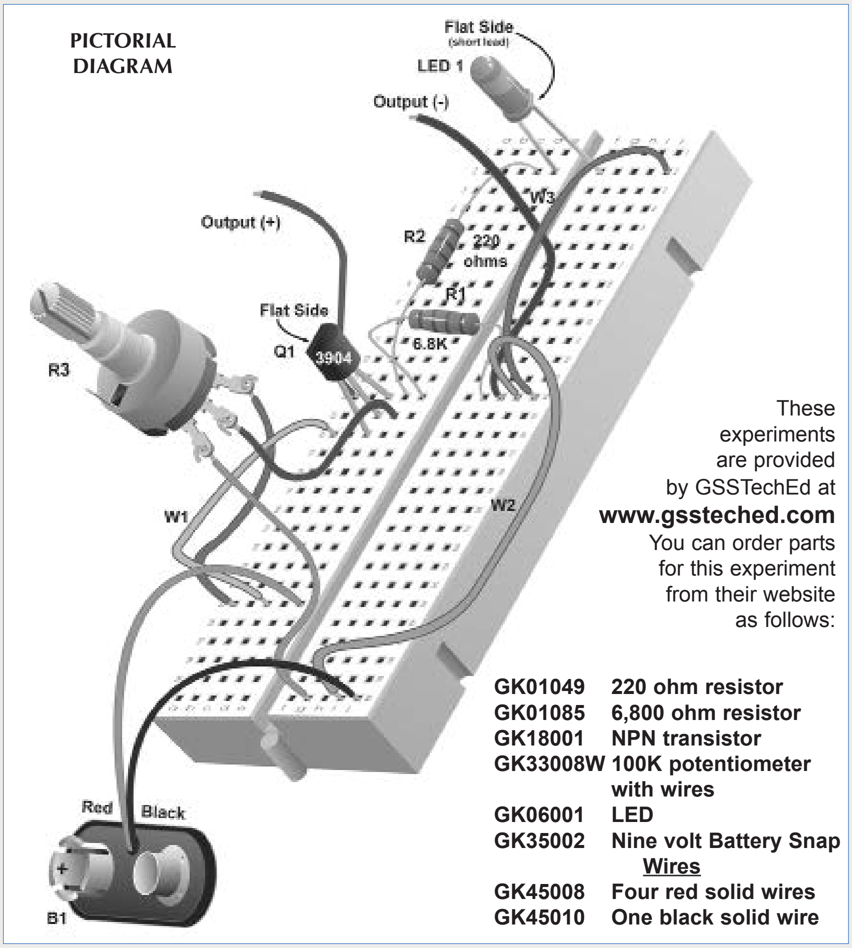
Theory: This circuit is a transistor with a potentiometer in place to adjust the bias on the base/collector circuit. By adjusting the bias with the potentiometer, you are able to get an output from zero volts DC to nine volts DC. The LED across the output lets you know that there is a voltage. The resistance across transistor Q1 is varied with the potentiometer. When the resistance across Q1 is high, there is no voltage at the output. As you lower the resistance across Q1 with the potentiometer, the voltage increases at the output of the power supply. This is therefore a DC to DC power supply.

Procedure: Connect a nine-volt battery to the battery snap and you should see the LED light up. By using a multimeter at the output, you should be able to adjust it from zero to nine volts.



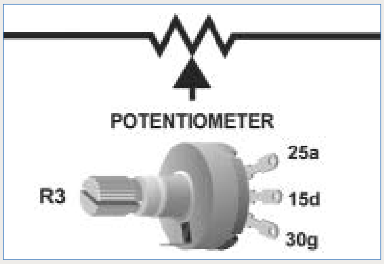


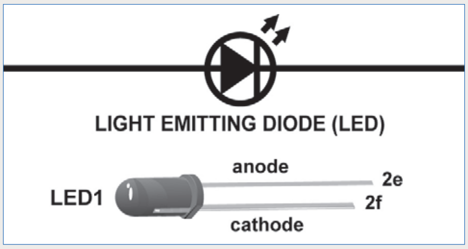


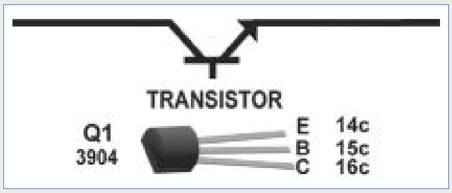


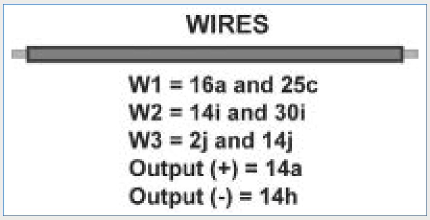
These experiments are provided by GSSTechEd at **www.gssteched.com** You can order parts for this experiment from their website as follows:

GK01049	220 ohm resistor
GK01085	6,800 ohm resistor
GK18001	NPN transistor
GK33008W	100K potentiometer with wires
GK06001	LED
GK35002	Nine volt Battery Snap
	Wires
GK45008	Four red solid wires
GK45010	One black solid wire









CLASSIFIEDS

KITS/PLANS

ROBOTICS

HARDWARE WANTED

SURPLUS

COMPONENTS

PCB SOFTWARE

ENERGY SOLUTIONS

DESIGN/ENG SERVICES

OSCILLOSCOPES

TECH FORUM
READER-TO-READER

Send all questions and answers by email to **forum@nutsvolts.com**
*Check at **www.nutsvolts.com** for tips and info on submitting to the forum.*

>>> QUESTIONS

Backup Alarm

I would like to install a backup alarm on my '07 Mazda Miata with the Mazda jingle (ZOOM ZOOM). I can get the jingle from YouTube but I don't know if it's possible to erase the sounds on a current backup alarm and install the jingle, or build an entirely new alarm. Are there any plans, diagrams, etc., available?

#8111 **John Mierta**
Las Vegas, NV

Recycle Old Answering Machines

Is there anything I can do with my old answering machines? Newer electronic machines are available at low prices and must have important parts or uses!

#8112 **James Muller**
Elyria, OH

Analog S Meter

I have a vintage Realistic DX-400 shortwave radio. The signal meter finally gave out after all these years. I would like a small circuit to fit in a small confined space, using a 10 bargraph LED. The bargraph should fit perfect in the small window where the needle meter once was.

#8113 **Frank S Barren Jr**
Berwick, PA

SMT Solution

I am moving from through hole PCBs to surface-mount technology and am looking for a nominally cost-effective way of doing reflow soldering. I would prefer a solution better than a toaster oven. Are there inexpensive reflow ovens (under $1,000) that you can suggest or recommend? I imagine this is something many *Nuts & Volts* readers would love to know!

#8114 **Joe Menard**
via email

Optical Speed Measurement

I want a camera-like device that can measure the speed of a surface that is passing by. The passing surface is moving perpendicular to the focus plane at some modest speed TBD by sensor element speeds, I suppose. For example, this little camera looks at a piece spinning on the lathe and reports "five meters/second surface speed." To do this, the device has to be able to detect the speed of a moving image pattern falling on the sensor array. If we can detect that, we can calculate the speed of the actual object using focal length and distance of the surface. However, the light readings from the sensor array would need clever measurements to determine the speed.

#8115 **Dion Johnson**
Scotts Valley, CA

>>> ANSWERS

[#4115 - April 2011]

LED/LCD Digital Counter

I am looking for a fairly large (6-8 or 10-12 inch high) LED or LCD, two digit, digital display which could be changed by using perhaps one button to increase (by one at a time) and one button to decrease the displayed value, i.e., 00 to 99.

It also would be good to just use a numbered keypad to enter the value.

Another useful feature would be to turn on and off (flashing at a chosen rate) the display to indicate urgency.

This is to be in an office area where they are serving the public. Most of you have been where there is a service counter and you need to "take a number" to secure your place in line. Often, they will have a "Number Being Served" display to show you where you are in line.

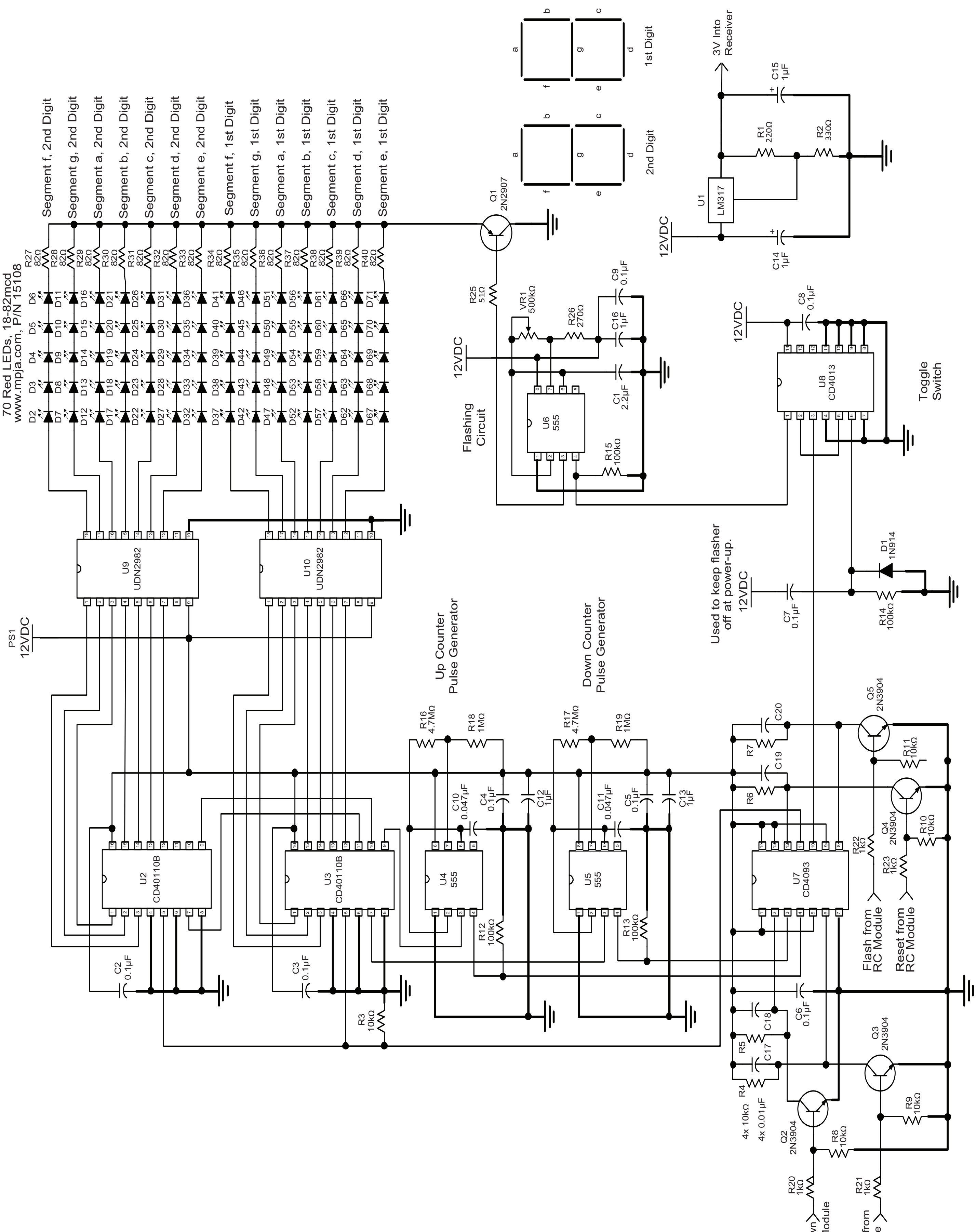
Well, my application would be to enable the employees (they can see the display from a distance at their desks) to view how many people are currently waiting to be seen. If there are not so many, the employee can afford to give the customers a bit more time if needed or if very busy, will have to keep the visit as brief as possible.

Right now, they are using an old mechanical flip-number type display which is a rather awkward (and not too state-of-the-art method) way to do it. An employee is continually jumping up from their desk (interrupting their work and time) to change the sign.

I would like to learn what the circuit design could be and what components are needed so I could construct it myself.

Here is a circuit (**Figure 1**) that will help prevent an employee from getting up to push buttons on the display or running several feet of wire to a button controller. The circuit has been designed and tested to work with a $10 remote control car. Remove the receiver from the car and disconnect the motors and battery pack. Wire the motor connections from the receiver to the circuit. Each motor connection has two wires and each wire supplies a signal to the circuit. I used the forward/back motor connection for the up/down counters and the right/left connection for the reset/flash functions to make the remote easier to understand and operate. Supply power to the receiver with U1.

VR1 allows you to control how fast you want the display to blink. The



70 Red LEDs, 18-82mcd
www.mpja.com, P/N 15108
Segment f, 2nd Digit
Segment g, 2nd Digit
Segment a, 2nd Digit
Segment b, 2nd Digit
Segment c, 2nd Digit
Segment d, 2nd Digit
Segment e, 2nd Digit
Segment f, 1st Digit
Segment g, 1st Digit
Segment a, 1st Digit
Segment b, 1st Digit
Segment c, 1st Digit
Segment d, 1st Digit
Segment e, 1st Digit
Q1 2N2907
R25 51Ω
1st Digit
2nd Digit
3V Into Receiver
C15 1µF
R1 220Ω
R2 330Ω
U1 LM317
12VDC
C14 1µF
VR1 500kΩ
R26 270Ω
C9 0.1µF
C16 1µF
C1 2.2µF
R15 100kΩ
U6 555
Flashing Circuit
C8 0.1µF
U8 CD4013
Toggle Switch
U9 UDN2982
U10 UDN2982
PS1 12VDC
Used to keep flasher off at power-up.
C7 0.1µF
R14 100kΩ
D1 1N914
Up Counter Pulse Generator
Down Counter Pulse Generator
R16 4.7MΩ
R18 1MΩ
R17 4.7MΩ
R19 1MΩ
C10 0.047µF
C4 0.1µF
C12 1µF
C11 0.047µF
C5 0.1µF
C13 1µF
U2 CD40110B
U3 CD40110B
U4 555
U5 555
U7 CD4093
R12 100kΩ
R13 100kΩ
C2 0.1µF
C3 0.1µF
R3 10kΩ
C6 0.1µF
Q5 2N3904
Q4 2N3904
Q3 2N3904
Q2 2N3904
R11 10kΩ
R10 10kΩ
R9 10kΩ
R8 10kΩ
R22 1kΩ
R23 1kΩ
R20 1kΩ
R21 1kΩ
R4
R5
R6
R7
C17
C18
C19
C20
4x 10kΩ
4x 0.01µF
Flash from RC Module
Reset from RC Module
Count Down from RC Module
Count Up from RC Module

Figure 1

reset button sets the display to 00. Pushing either counter button once will cause the display to count up or down once. Holding down either counter button will cause the display to continuously count up or down. Pushing the flash button will toggle the display to either flash or stay on.

Make your own display by mounting LEDs onto a piece of 12" square, 1/8" thick plastic such as McMaster 8747K112. To make the display easier to see, mount a clear, gray-tinted plastic of the same size over the LEDs, such as McMaster 85805K26.

Each segment is made up of five LEDs spaced 0.8" to 0.9" apart, center-to-center yielding 8" to 9" high digits. LCDs consume far less current, but finding the sizes you've requested can be difficult and pricey. Using LEDs lowers the cost considerably and gives you more options such as size, colors, etc. For an indoor environment, use only diffused LEDs rated about 200 mcd or less.

[#6117 - June 2011]

AC Resistance Needed

How can I find the AC resistance of the emitter diode of a power transistor?

AC resistance of the emitter diode of a transistor (re) can be calculated by:

$$re = .026v / Ie$$

where Ie is the DC emitter current. This is the solution for the resistance at room temperature of 25 degrees Celsius. You can find the emitter current by measuring it directly (open the emitter lead and insert the milliammeter), indirectly, (if there is a single external emitter resistor, measure the voltage drop on the emitter resistor and use Ohms Law to find the current), or by simulation.

To simulate, download LTspice (free) and input the transistor circuit you want to use. Place your cursor over and around the emitter until you see the clamp-on ammeter icon, then left-click. The emitter current will be plotted.

Bill Bradrick
Pittsburg, KS

Best Selling Kits for Electronic Enthusiasts

KIT OF THE MONTH

Ultrasonic Antifouling Kit for Boats

KC-5498 $155.75 plus postage & packing

Marine growth electronic antifouling systems can cost thousands. This project uses the same ultrasonic waveforms and virtually identical ultrasonic transducers mounted in sturdy polyurethane housings. By building yourself (which includes some potting) you save a fortune! Standard unit consists of control electronic kit and case, ultrasonic transducer, potting and gluing components and housings. The single transducer design of this kit is suitable for boats up to 32ft (10m) boats longer than about 14m will need two transducers and drivers. Basically all parts supplied in the project kit including wiring.
(Price includes epoxies).

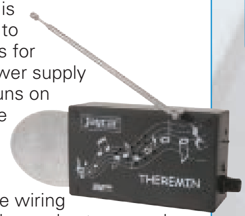
- 12VDC
- Suitable for power or sail
- Could be powered by a solar panel/ wind generator
- PCB: 78 x 104mm

Best Seller

Theremin Synthesiser Kit MkII

KC-5475 $54.00 plus postage & packing

The ever-popular Theremin is better than ever. It's easier to set up with extra test points for volume adjustment and power supply measurement and it now runs on AC to avoid the interference switchmode plugpacks can cause. It's also easier to build with PCB-mounted switches and pots to reduce wiring to just the hand plate, speaker and antenna and has the addition of a skew control to vary the audio tone from distorted to clean.



- Complete kit contains PCB with overlay, pre-machined case & all specified components
- PCB Dimensions: 85 x 145mm

SLA Battery Health Checker Kit

KC-5482 $57.75 plus postage & packing

The first versions of the battery zapper included a checker circuit. The Mk III battery zapper (KC-5479) has a separate checker circuit - and this is it. It checks the health of SLA batteries prior to charging or zapping with a simple LED condition indication of fair, poor, good etc.

- Overlay PCB and electronic components
- Case with machined and silk-screened front panel
- PCB Dimensions: 185 x 101mm

Don't just sit there BUILD SOMETHING!

SD/MMC Card Webserver Kit in a Box

KC-5489 $65.00 plus postage & packing

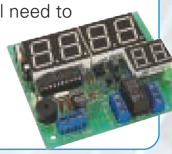
Host your own website on a common SD/MMC card with this compact Web server In a Box (WIB). Connecting to the Internet via your modem/router, it features inbuilt HTTP server, FTP server, SMTP email client, dynamic DNS client, RS232 serial port, four digital outputs and four analogue inputs. Requires a SD memory card, some SMD soldering and a 6-9VDC adaptor. Kit includes PCB, case and electronic components.

- PCB Dimensions: 123 x 74mm

Remote Control Digital Timer Kit

KC-5496 $29.00 plus postage & packing

Remote-controlled digital timer with a bright 20mm-high 7-segment red LED display. It can count up or down from one second to 100 hours in 1-second increments. Its timing period can either be set and controlled using the remote control or it can be automatically controlled via external trigger/reset inputs. An internal relay and buzzer activate when the unit times out. The relay contacts can be used to switch devices rated up to 30VDC or 24VAC and the project can be powered from a plugpack or a battery. Short form kit only - you'll need to add your own universal remote, power supply and enclosure.

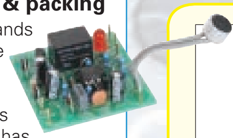


- 9-12VDC @300mA
- PCB and components included
- PCB Dimensions: 80 x 80mm

"Minivox" Voice Operated Relay Kit

KC-5172 $12.00 plus postage & packing

Voice operated relays are used for 'hands free' radio communications and some PA applications etc. Instead of pushing a button, this device is activated by the sound of a voice. This tiny kit fits in the tightest spaces and has almost no turn-on delay. 12VDC @ 35mA required. Kit is supplied with PCB electret mic, and all specified components.



- PCB Dimensions: 47 x 44mm

Audio Amplifier Kits

"The Champ" Audio Amplifier Kit

KC-5152 $6.00 plus postage & packing

This tiny module uses the LM386 audio IC, and will deliver 0.5W into 8 ohms from a 9 volt supply making it ideal for all those basic audio projects. It features variable gain, will happily run from 4-12VDC and is smaller than a 9 volt battery, allowing it to fit into the tightest of spaces. PCB & all electronic components included.

- PCB Dimensions: 46 x 26mm

"Pre-Champ" Versatile Preamplifier

KC-5166 $6.50 plus postage & packing

Designed to be used with the 'Champ' amplifier KC-5152. Unless you have a signal of sufficient amplitude the 'Champ' will not produce its maximum power output. The 'Pre-Champ' is the answer with a gain in excess of 40dB, which is more than enough for most applications. You can vary the gain by changing a resistor and there is even provision on the PCB for an electret microphone.

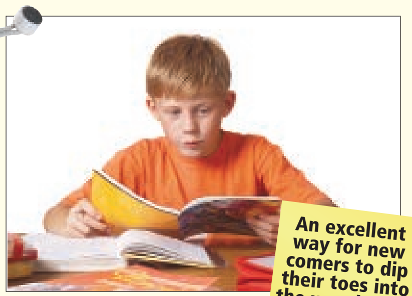
- Power requirement 6 - 12VDC
- Kit includes PCB & electronic components
- PCB Dimensions: 46 x 36mm

50 Watt Amplifier Module

KC-5150 $22.00 plus postage & packing

This 50 watt unit uses a single chip module and provides 50WRMS @ 8 ohms with very low distortion. PC Board and all electronic components supplied. Requires heatsink. See website for full specs.

- PCB Dimensions: 84 x 58mm
- Heatsink to suit HH-8590 $12.50



An excellent way for new comers to dip their toes into the wonders of electronics!

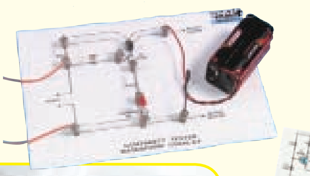
Teach Your Kids Electronics!

Short Circuits Volume 1, will teach you everything you need to get started in electronics and is suitable for ages 8+. We give you the option of buying the book on its own, or together with the accompanying kit that contains the components for each of the 20-odd projects described in the book. Some of the exciting projects include a Police Siren, Electronic Organ, Sound Effects Unit, Light Chaser and many, many more! The full color 96 page book, is lavishly illustrated with over 100 drawings and diagrams. No prior knowledge of electronics is needed, projects are fun and totally safe to build.

Short Circuits Book
BJ-8502 $7.25

Short Circuits Project Kit
KJ-8504 $24.50

Short Circuits Book and Project Kit
KJ-8502 $32.50



MORE INFO AVAILABLE ONLINE www

AGES 8+

Post & Packing Charges

Order Value	Cost
$25 - $49.99	$7.50
$50 - $99.99	$20
$100 - $199.99	$40
$200 - $499.99	$60
$500+	$75

Max weight 12lb (5kg).
Heavier parcels POA.
Minimum order $25.

Note: Products are dispatched from Australia, so local customs duty & taxes may apply.
Prices valid until 31/8/2011

HOW TO ORDER

WEB: www.jaycar.com
PHONE: 1-800-784-0263*
FAX: +61 2 8832 3118*
EMAIL: techstore@jaycar.com
POST: P.O. Box 107, Rydalmere NSW 2116 Australia

• ALL PRICING IN US DOLLARS • MINIMUM ORDER ONLY $25

*Australian Eastern Standard Time
(Monday - Friday 09.00 to 17.30 GMT + 10 hours)
Note: 10-14 days for air parcel delivery

Order online: www.jaycar.com

Jaycar Electronics

BlackJack SolderWerks Inteligent Preheating System

Note: Hot Air gun shown but not included.

The BK7000 & BK7050 from *BlackJack SolderWerks use a highly efficient quartz infrared heater system that provide fast heat up and utilizes a feedback control circuit to optimize the PCB pre-heat operation.

They are designed to aid the reworking operation for large and multilayer PCBs (such as game consoles and motherboards) that use either traditional or lead free solder.

Both systems have three modes of operation.

Once again Circuit Specialists brings you a quality product at a great price!

Type 0: for single temperature heating using an internal temperature sensor.

Type 1: for more precise temperature control using the supplied external temperature probes.

Type 2: for a fully automated time and temperature based profile mode.

Item# **BK7000** **$288.00**

250mm x 200mm (9.8"x7.8") Pre-Heat area

www.CircuitSpecialists.com/BK7000

Item# **BK7050** SALE **$339.00**

300mm x 300mm (11.8"x11.8") Pre-Heat Area

www.CircuitSpecialists.com/BK7050

*BlackJack SolderWerks is a Circuit Specialists, Inc. brand.

3 Channel Programmable Regulated DC Power Supplies

Check out our new programmable hi preformance 3 channel power supplies. Featuring both USB & RS232 interfaces, Overload Protection, Auto Fan Control, and Series or Parallel Operation. Both units feature a Large LCD display panel with simultaneous output and parameter view and a keypad for control. They are ideal for applications requiring high resolution, multiple output, and automated operation such as in production testing. There are both fine and coarse controls via the shuttle knob and 90 memory settings. Software included.

CSIPPS33T 0~32V / 0~3A **$299.00**

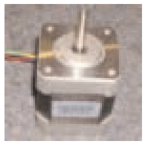
www.CircuitSpecialists.com/CSIPPS33T

CSIPPS55T 0~32V / 0~5A **$379.00**

www.CircuitSpecialists.com/CSIPPS55T

Stepper Motors and Controllers

Circuit Specialists carries a large selection of stepper motors and controllers



42BYGH207
NEMA 17
3.0 kg.cm Torque
$14.90 ea.

42BYGH404
NEMA 17
3.4 kg.cm / 47 oz.in Torque
$14.90 ea.

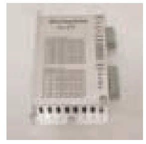
57BYGH207
NEMA 23
8.0kg.cm / 111 oz.in Torque
$19.50 ea.

57BYGH303
NEMA 23
15 kg.cm / 208 oz.in Torque
$23.95 ea.

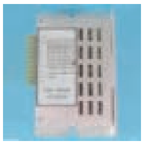
57BYGH405A
NEMA 23
20 kg.cm / 277 oz.in Torque
$27.50 ea.

85BYGH450B-03
NEMA 34
48 kg.cm / 665 oz.in Torque
$68.00 ea.

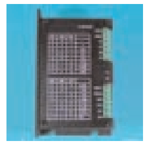
85BYGH450C-03
NEMA 34
63 kg.cm / 870 oz.in Torque
$89.00 ea.



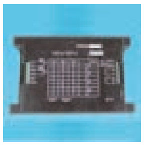
QJ-215
Microstepping Stepper Motor Driver for our 42 Series Stepper Motors.
$54.90 ea.



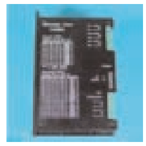
XCW-220
2 Phase Microstepping Stepper Motor Driver for our 42 & 57 series stepper motors.
$39.95 ea.



CW230
2 Phase Microstepping Stepper Motor Driver for our 42 & 57 series stepper motors.
$48.50 ea.



CW250
2 Phase Microstepping Stepper Motor Driver for our 57 & 85 series stepper motors.
$54.95 ea.



CW860
2 Phase Microstepping Stepper Motor Driver for our 57 & 85 series stepper motors.
$96.00 ea.

http://www.CircuitSpecialists.com/stepper-motors-and-controllers

60 Watt Digital Soldering Station

For use with traditional or Lead Free Soldering

This is a terrific soldering station featuring a microprocessor controlled temperature set up system with great versatility. The unit includes 3 preset temperature settings that are user definable so you can turn on the system & push one button to go to the temperature range you desire. Specific system temps can also be set with an easy to use push button up/down button AND when you turn off this station, the unit keeps the last used temperature in memory & automaticly returns to that setting the next time the user turns the system on. The temperature display can be set to display in Celsius or Fahrenheit scale.

The CSI-Station-3DLF is a powerful 60 watt soldering system. The fast heat recovery provided by a 60 watt system like this allows the user to solder both traditional solder and lead free solder. This system features a grounded tip to protect delicate circuits from static charge. Also included is a seperate iron holder. Circuit Specialists stocks a large supply of tips for this station.

Features:

* 60 watt dual core ceramic heater
* 150 to 450 degree Celsius Temperature range
* Versatile easy to read liquid crystal display
* 3 preset & user definable temperature settings.
* Automaticly remembers previous temperature setting
* Display in Celsius or Fahrenheit scale
* 3 foot cord length from station to iron tip
* Broad selection of replacement tips available

Item# **CSI-Station-3DLF** **$49.00**

www.CircuitSpecialists.com/CSI-Station-3DLF

Circuit Specialists carries a wide selection of traditional & lead-free soldering equipment, including soldering irons, soldering stations, hot-air rework stations, focused infrared systems, reflow ovens, pre-heating systems, regular and lead-free solder, solder flux, and soldering aids for both thru-hole and surface-mount soldering.

See our complete line of soldering equipment at:

www.CircuitSpecialists.com/soldering-equipment

Back to School

Want to start building your own inventions? Get with the Stamps in Class program. Learn electronics, programming, robotics and sensor management with the easy-to-use BASIC Stamp microcontroller. Free software, tutorials, and sample code downloads are available online. Start with either kit below, and then explore all ten books in the series. Our kits work well for the individual learner, or an entire classroom STEM program.

BASIC Stamp Discovery Kit (includes *What's a Microcontroller?* tutorial):

- Serial Version w/USB adapter & cable - #27207; $159.99
- USB Version - #27807; $159.99

Boe-Bot Robot Kit (includes *Robotics with the Boe-Bot* tutorial):

- Serial Version w/USB adapter & cable - #28132; $159.99
- USB Version - #28832; $159.99

Ready for more? Explore multicore microcontroller technology with the Propeller Education Kit and Labs. With 8 processors that run simultaneously, the Propeller is an ideal controller for quickly integrating all the pieces of your embedded system design project. As always, the manual, labs, software, and sample code are available as free downloads.

Propeller Education Kit:

- 40-Pin DIP Version - #32305; $99.99
- PropStick USB Version - #32306; $99.99

Learn more about the **Parallax Education** program at www.parallax.com/education or call toll-free at 888-512-1024 (M-F, 7AM-5PM, PDT).

Friendly microcontrollers, legendary resources.™

PARALLAX INC

www.parallax.com

"ParallaxInc" on Twitter, Facebook, and YouTube