Practical Electronics

The UK's premier electronics and computing maker magazine

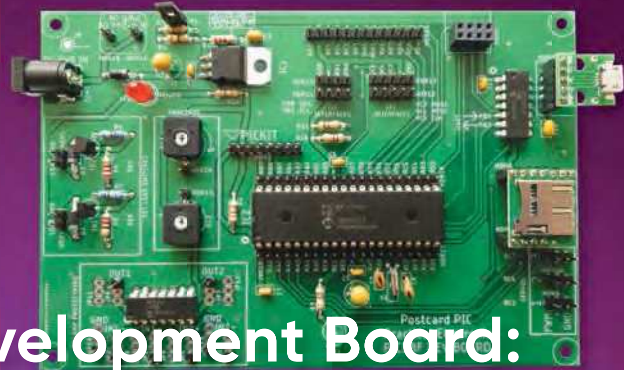
Circuit Surgery
Understanding Capacitor Dielectric Absorption

Audio Out
Introduction to the all-new PE Analogue Vocoder

Microcontrollers
Using I/O Expander Modules

Animating LED eyes for your robot!

WIN! Microchip MCP6C02 Evaluation Board WIN!



PIC18F Development Board: using Internet dashboards

Dual Battery Lifesaver

Building the all-new Colour Maximite 2 (Gen 2)

192kHz, 24-bit

SuperCodec: up and running

PLUS!

Techno Talk – On your marks...

Instrumentation – Thermometer Calibrator

Net Work – Supply chain woes and the endless rise of apps

Nov 2021 £5.49

MPLAB® Cloud Tools Ecosystem

Discover, Configure and Develop: An Ecosystem for All Your Ideas

MPLAB cloud tools ecosystem is a complete online solution for all skill-level users, to discover, configure, develop and debug embedded PIC® and AVR® microcontroller (MCU) applications.

- Intuitive entry into PIC and AVR MCU development with integrated MPLAB tools
- Quick prototyping with Microchip's PIC and AVR MCUs using Curiosity boards
- No software installation to start developing PIC and AVR MCU solutions

Designers who use PIC and AVR MCUs can visit the MPLAB Cloud Tools page and start developing based on their design phase.

- Search and Discovery: Access MPLAB Discover to find fully configured and complete source code projects
- Configure Code: Easily configure software applications with MPLAB Code Configuration
- Develop and Debug: Developing, debugging and deploying project applications directly from a preferred browser can be completed without any software installation with MPLAB Xpress IDE

MPLAB® DISCOVER

MPLAB® CODE CONFIGURATOR

MPLAB® XPRESS IDE

MICROCHIP

microchip.com/MPLABcloudtools

Practical Electronics

Volume 50. No. 11
November 2021
ISSN 2632 573X

Contents

Projects and Circuits

Series, Features and Columns

Regulars and Services

Made in the UK.

Written in Britain, Australia, the US and Ireland.

Read everywhere.

The December 2021 issue of *Practical Electronics* will be published on Thursday, 4 November 2021 – see page 72.

Practical Electronics

UK readers SAVE £1 on every issue

Take out a one-year subscription and save £10 over the year.
Even better – save £1 per issue if you subscribe for two years – a total saving of £24.

Overseas rates represent exceptional value

You also:
- **Avoid any cover price increase for the duration of your subscription**
- **Get your magazine delivered to your door each month**
- **Guarantee a copy, even if the newsagents sell out**

Order by:
- **Phone or post with a cheque, postal order or credit card**
- **Or order online at: www.electronpublishing.com**

Practical Electronics

SUBSCRIPTION PRICES

Subscriptions for delivery direct to any address in:

UK: 6-months £26.99, 12-months £49.85, 24-months £94.99

Europe Airmail: 6-months £30.99, 12-months £57.99, 24-months £109.99

Rest Of The World Airmail: 6-months £37.99, 12-months £70.99, 24-months £135.99

Cheques (in **£ sterling only**) payable to *Practical Electronics* and sent to: Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Tel: 01202 087631
Email: pesubs@selectps.com

Also via our online shop at**: www.electronpublishing.com**

Subscriptions start with the next available issue. We accept MasterCard or Visa.
(For past issues see the Back Issues page.)

DIGITAL SUBSCRIPTIONS

Electronic subscriptions are available from £24.99 for 12 months, more details at: **www.electronpublishing.com**

SUBSCRIPTION ORDER FORM

☐ **6 Months: UK £26.99, Europe £30.99 (Airmail), Rest of the World £37.99 (Airmail)**

☐ **1 Year: UK £49.85, Europe £57.99 (Airmail), Rest of the World £70.99 (Airmail)**

☐ **2 Years: UK £94.99, Europe £109.99 (Airmail), Rest of the World £135.99 (Airmail)**

To: **Practical Electronics Subscriptions**
PO Box 6337
Bournemouth BH1 9EH
United Kingdom

Tel: 01202 087631

Email: pesubs@selectps.com

I enclose payment of £ ....
(cheque/PO in £ sterling only)
payable to *Practical Electronics*

☐ Please charge my Visa/Mastercard

My card number is: ....
Please print clearly, and check that you have the number correct

Card Security Code .... Valid From Date.......................
(The last 3 digits on or just under the signature strip)

Card Ex. Date ....

Name ....

Address ....

Post code .... Tel. ....

Email ....

Subscriptions start with the next available issue.

Practical Electronics

Volume 50. No. 11
November 2021
ISSN 2632 573X

Editorial

Editorial offices

Practical Electronics
Electron Publishing Limited
1 Buckingham Road
Brighton
East Sussex BN1 3RA

Tel 01273 777619
Mob 07973 518682
Fax 01202 843233
Email pe@electronpublishing.com
Web www.electronpublishing.com

Advertisement offices

Practical Electronics Adverts
1 Buckingham Road
Brighton
East Sussex BN1 3RA

Tel 01273 777619
Mob 07973 518682
Email pe@electronpublishing.com

Editor Matt Pulzer
General Manager Louisa Pulzer
Digital subscriptions Stewart Kearn **Tel** 01202 880299
Online Editor Alan Winstanley
Web Systems Kris Thain
Publisher Matt Pulzer

Print subscriptions

Practical Electronics Subscriptions
PO Box 6337
Bournemouth BH1 9EH
United Kingdom

Tel 01202 087631
Email pesubs@selectps.com

Sourcing silicon

Over the last few years, in our October or November issues, we've often published a fun and light-hearted Christmas project. Something easy to make and which would brighten a Christmas tree or make a nice, inexpensive present. This year we planned to go a little further and offer eight little decorations based on PCBs cut into festive shapes – Father Christmas, a tree and so on – all covered with twinkling multi-coloured LEDs. We ordered the boards (hundreds of them) plus thousands of LEDs to make up kits for readers to assemble in time for Christmas. All was plain sailing until I performed a final check of the parts list. That's when the global shortage of ICs hit home. The very ordinary 50p 8-bit PIC that drives the LEDs was unobtainable. Likewise, a viable alternative was also reported MIA. Both chips were quoted by multiple major suppliers as being unavailable until June next year. These aren't esoteric, low-volume ICs – just cheap, bread-and-butter microcontrollers.

Perhaps, in retrospect, this is not so surprising. If industrial giants like VW are having trouble manufacturing cars because of global supply chain issues with silicon then it was bound to hit hobbyists too.

A little light at the end of the tunnel?

However, it's not all doom and gloom. First, most of you usually only need one or two of an item. Even if your preferred suppliers are out of stock then do have a hunt around on eBay or Amazon. You may well pay over the odds, but if it's just a case of paying a couple of pounds for a 25p device to complete a large project then while it's certainly irritating, it's not catastrophic.

Second, I've recently discussed the problem with my colleague Nicholas Vinen, the editor of Australia's *Silicon Chip* magazine. He buys a lot more parts than me, so he's generally better informed about the ups and downs of silicon supply chains for readers.

Nicholas thinks the situation has probably stabilised – ie, at least it's not getting worse. He reports that although the situation is still bad – with lots of basic items running low, especially MOSFETs – there are signs that components are now being restocked, albeit at a frustratingly slow rate.

I think the best advice we can give is do check that all parts are available before spending money on projects, and do be prepared to look further afield, even if you have to pay a little more.

Keep well everyone

Matt Pulzer
Publisher

The Fox Report

Barry Fox's technology column

VR/AR on tour in the UK

Will virtual reality (VR) and augmented reality (AR) change the face of entertainment or go the way of 3D? They all have in common the need to wear something on your head and over the eyes – which in the case of VR/AR is a bulky headset with built-in screen display, headphone audio and position-tracking electronics, plus a pair of hand controllers like gaming joysticks.

Gamers will have no problem with this. Non-gamers (like me) need a very good reason to wear any extra headgear. The best good reason will be superb, captivating audio-visual content.

Immersive arcade

I am assured by the UK government-backed venture, Digital Catapult, based at high tech offices in London near Euston Station, that the UK is a world leader in VR/AR content. So, I jumped at the chance of attending a preview of the VR/AR Immersive Arcade package which is now touring the UK to give public demonstrations.

The Arcade showcases – I quote – '12 of the most influential British Immersive productions from the last 20 years, from theatre to games to therapy (and)…is designed to inspire and delight virtual reality (VR) pros and novices alike.'

Anyone with their own headset can access the material online, or watch out for a physical demo in their area of the UK. For information, go to:

www.immersivearcade.uk
www.immersivearcade.uk/showcase

Demos are free and take between 5-40 minutes per person, depending on how busy the venue is. The tour covers much of the country (from Brighton to Dundee) – for more details of locations, go to:
www.immersivearcade.uk/tour

The Oculus Rift headset and a pair of hand controllers.

The organisers recommend starting with *Mimetic Starfish at the Millennium Dome*, immersive game *Can You See Me Now*, Punchdrunk's 2003 *Sleep No More* dark retelling of *Macbeth*, a VR *Space Walk* used to train astronauts, some *Ghost Hunts* and *Sky Slides*, refugee experiences with *We Wait*, and VR puzzle *Shadow Point*.

At the Digital Catapult offices, I donned an Oculus Rift headset and – after some time spent learning how to navigate through a virtual gallery maze of content options – experienced an odd CGI experience with flying birds, a far-too-long and rather low-res video journey round a decaying housing estate ahead of demolition and a very exciting simulation of flight with Leonardo, the Wright Brothers, Concorde and a future giant dreamliner.

A space walk for all budding International Space Station visitors.

The Digital Ghost Hunt: an educational, immersive experience that combines augmented reality, coding education and theatre. (Created by KIT Theatre / King's Digital Lab at the University of Sussex.)

Fun, but perhaps not for everyone

The Flight Sim was hugely impressive and great fun, as well it should be because it was made with a bag of British Airways' money for a flight training pod that was previously installed for demonstration at the prestigious Saatchi Gallery.

Gamers will enthuse and explore the 'arcade' for hours and hours. Others, like me, will view it as an interesting experience which is variously invigorating, exciting, fatiguing, vertigo-inducing, unsettling and, after half an hour or so, starting to cause dull headache discomfort.

Give me a 2D movie with a good script, believable acting and tight direction, any time – and no need to wear anything on my head.

Just like 3D?

Significantly, before I could experience the VR experience, I was obliged to sign several pages of legal waivers in case I suffered medical side effects such as epilepsy. This exactly parallels what the situation was with 3D, and reminds of another good reason (along with the inconvenience of wearing special spectacles and getting headaches) why 3D has so spectacularly failed every time anyone tries to make it go mainstream.

Take a vertigo-inducing trip round London's iconic Shard – you'll either love it, hate it... or possibly both!

Automation

Did you know that the father of automated mass production was British? During World II, John Sargrove worked on the mass-production of electronic valves. He was sure he could automate the production of complete radio sets.

In 1944 he set up Sargrove Electronics in a village hall at Effingham in Surrey. Within two years, ECME, his Electronic Circuit Making Equipment was working. A 20-metre production line produced 1500 two-valve radio circuits in eight hours. The circuitry was printed onto an insulating board, with resistors deposited by spraying. Each PCB had 15 fixed-value capacitors, two variable capacitors and two valve sockets. All were automatically soldered into position by mechanical hands controlled by electromagnetic relays and valve circuitry. Humans were needed only to plug in the valves and 'tweak' pre-sets.

Between 1947 and 1949, Sargrove's factory designed 'master brains' to automate other industrial processes. A 'magic eye' monitored the output of an electric sewing machine, and counted screws, pins and buttons, as they fell into cartons. Other master brains shut down machinery in a cotton mill when a thread broke, matched the colour of rosary beads, sorted good from bad coffee beans, and checked the size of dough lumps on a conveyor leading to a baker's oven.

In the early 1950s Sargrove tried to automate the production of TV receivers. 'Don't get the idea that we are out to rob people of their jobs,' he assured. 'Our task is to liberate men and women from being slaves of machines… automation means redeployment not unemployment. It relieves people of monotonous jobs so that they can do more interesting work'.

But the unions were wary and government aid dried up. There was always at least one faulty valve or relay holding up the line. The factory closed and Sargrove died in 1974 just as the Japanese started to make automation a way of life.

More technology stories and images at: **https://tekkiepix.com/stories**

Practical Electronics is delighted to be able to help promote Barry Fox's project to preserve the visual history of pre-Internet electronics.

Visit **www.tekkiepix.com** for fascinating stories and a chance to support this unique online collection.

On your marks...

Mark Nelson

Confused by the markings on some of the electronic gadgets and other stuff that you own? You won't be for long! More confusion: we've also got news of free energy that definitely isn't free – and sounds altogether rather far-fetched.

In last month's Net Work column Alan Winstanley did a great job of unravelling the connection between the well-known CE product mark and the new, less familiar UKCA symbol, also explaining the overlap period and the more complex situation that applies in Northern Ireland. It all made sense, but it also brought to mind the zany TV sitcom series from the late seventies and early eighties called 'Soap'. Episodes began with a garbled summary of the previous episode's storyline that ended with the words, 'Confused? You won't be after this episode of Soap'. Usually, viewers were left even more confused.

If CE and UKCA were all you had to contend with, life might be straightforward enough, but what about those other letters and hieroglyphs on electronic products that mean precious little to most people? You're bound to have seen them, probably without a clue as to their meaning. What are they trying to tell us? Do they even concern us? Now you can finally find out…

Global marketing, global marking

Nowadays, many products are sold all over the world, meaning that they need to demonstrate compliance and conformity in all of the markets they are sold in, either officially or else as 'grey imports'. Each of the symbols seen in the collage here (curated by the US company CUI) relates either to one of the national or regional safety and standards agencies around the world, or else to one of the test laboratories. Some of the latter have labs in more than one country and/or test products to the standards of more than one country or region. It's also worth mentioning that the 21 bodies mentioned in this article do not form an exhaustive list; other labs, bodies and conformance schemes exist.

Whether all this concerns us greatly is debatable, I imagine. Nevertheless, for possibly the first time in your life, you can now discover the approvals body or testing house indicated by each of these markings. And score points for each symbol you identify correctly! (sorry, no prizes!)

Top row
From left to right we have: Underwriters Laboratories (USA); the reversed UR symbol means UL Recognised; the Canadian Standards Association mark for products acceptable in either Canada and/or the USA; Normas Oficiales Mexicanas (Mexico); Inmetro (Brazil); the S mark of IRAM (Argentina); and our old friend CE (Europe).

Centre row
European Norm (Europe); TüV Rheinland (Germany); Nemko (Norway); Eurasian Conformity (Russia); Geprüfte Sicherheit/safety-tested; DIN (Deutsches Institüt für Normung) / German Institute for Standardisation; and Verband der Elektrotechnik Elektronik und Informationstechnik/Association for Electrical, Electronic and Information Technologies (Germany).

Bottom row
British Standards Institution kite mark (UK); DENAN PSE (Japan), KC (Korea); BSMI (Taiwan); CCC (China); PSB (Singapore); and the RCM Tick Mark (Australia).

Free energy, but at a price

As the clock counts down to 2050, when the UK is committed to be nett-neutral in terms of carbon-dioxide emissions, researchers are becoming ever more imaginative in their schemes for meeting this target. It is claimed, for instance, that it might be possible to transform the world's largest desert, the Sahara, into a giant solar farm, capable of meeting four times the world's current energy demand. Blueprints have been drawn up for projects in Tunisia and Morocco that would supply electricity for millions of households in Europe (let's hope they can compensate for the voltage drop in the cables). Whether or not Europe will allow itself to become dependent on electricity sources outside its political (military) control remains to be seen. (Many already doubt the wisdom of putting so many of their natural gas eggs in one Russian basket.)

A superior scheme for capturing the sun's energy would be to launch vast solar panels into space, convert sunlight into microwave energy and transmit this back to earth. In theory, this Space Solar Power (SSP) sounds like a very elegant solution, especially as the lifetime cost of microwave energy beamed from geostationary satellites could be half that of ground-based nuclear power. What the chief British protagonist of this solution fails to explain, however, is how the high-powered micro-wave beam can be constrained into a 98km^2 ground antenna without irradiating all life over a far wider area. Anyone who has worked with microwave beam transmissions knows that substantial overspill occurs that could be lethal. The minimalist web page on the subject (**www.fnc.co.uk/sbsp/**) overlooks this matter entirely, which rather casts doubt on the fundamental feasibility of the entire project. But now is far too early to prejudge the outcome, especially as the detailed report on this scheme has yet to be published.

Exclusive offer

Win a Microchip MCP6C02 Evaluation Board

Practical Electronics is offering its readers the chance to win a Microchip MCP6C02 Evaluation Board (ADM01104) – and even if you don't win, receive a 20%-off voucher, plus free shipping for one of these products.

This board demonstrates the performance of Microchip's MCP6C02 high-side current-sense amplifier.

The MCP6C02 amplifier is AEC-Q100 qualified and offered in both a Grade 1 6-pin SOT-23 package and Grade 0 8-pin 3x3 VDFN package. Delivering a maximum offset error of only 12µV, the VDFN package offers the lowest offset voltage for any Grade 0 high-side current-sense amplifier. Specified over a temperature range of –40°C to +150°C, its market-leading offset error allows the use of smaller-value shunt resistors while also maintaining a high measurement resolution. This enables a more accurate and energy efficient current measurement solution for those applications exposed to extreme temperatures, like the motor within a vehicle's water pump.

Worth $71.99 (approx £52.00) each

Microchip's MCP6C02 device also features an on-chip EMI filter and a zero-drift architecture. The EMI filter helps provide added protection against high-frequency electrical interference, such as wireless hotspots and radio frequencies, while the self-correcting architecture brings increased accuracy to current measurement. Together, these features enable developers to create higher performance solutions in a wide variety of applications, such as creating a current-controlled feedback loop for a power supply or motor, monitoring and charging batteries, or monitoring current levels for safety reasons.

A load is supplied on the board for a convenient demonstration of the evaluation board. Supplying power and a load to the evaluation board will produce a small, measurable voltage drop across a shunt resistor on the evaluation board. This voltage drop is then amplified by the MCP6C02 on the board and outputted differentially at the test points: VOUT+ and VOUT– (TP1 and TP2).

Free-to-enter competition

Microchip MCP6C02 Evaluation Board

How to enter

For your chance to win a Microchip MCP6C02 Evaluation Board or receive a 20%-off voucher, including free shipping, enter your details in the online entry form at:

https://page.microchip.com/PE-6C02.html

Closing date

The closing date for this offer is 31 October 2021.

July 2021 winner

Mark Peters

He won a Microchip MPLAB ICD 4 In-Circuit Debugger

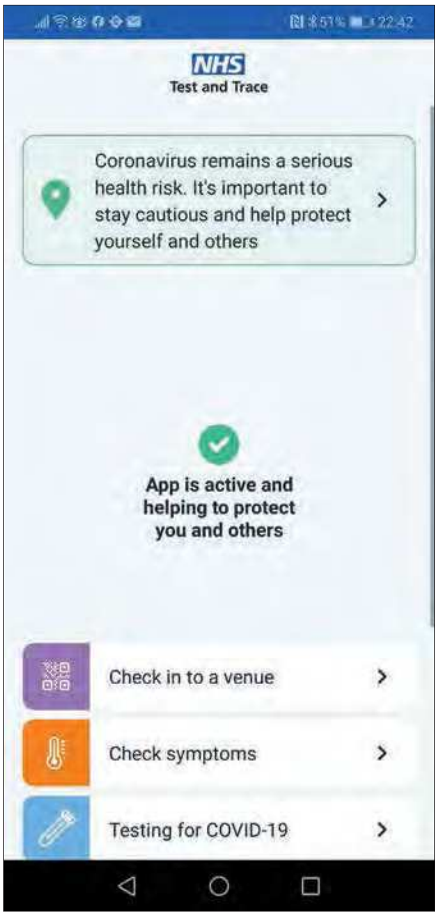
Net Work

Alan Winstanley

This month, *a* warning that multiple global supply chain issues mean that readers should plan ahead for Christmas. Plus, the unstoppable rise of apps and a handy gadget for visiting remote locations.

Although the Christmas festivities are still a little way off, there are some warning signs that this year's holiday season could be different from those we've had before. Apart from the disruption caused to families by social distancing, for more than a year many manufacturing industries have been in turmoil, caused by the global lack of semiconductor chips and a semi-absent ghost workforce operating on reduced hours, often due to the need to self-isolate. Factory layouts and production also had to be re-configured to cater for a spartan, socially distanced workforce.

In Britain, the term 'pingdemic' was coined to highlight the disruption caused when the NHS mobile phone app 'pinged' to say that the users must go into self-isolation, which resulted in terrible staff shortages that brought many businesses grinding to a near halt. The app has since been tweaked to make it more forgiving.



Pingdemic: the NHS app warns of Covid-19 risks and caused chaos among Britain's workforce before being tweaked.

The upheaval in industry caused by Covid-19 lockdowns has filtered through to the front lines and the laws of unintended consequences have taken hold. In both retail and industrial sectors, many supply lines have been starved of essential stocks and are living hand-to-mouth, with critical parts being in short supply. This hits just-in-time manufacturing hard. In early September, *Automotive News* reported that Stellantis – the global group behind car makers PSA (Peugeot Citroen), Vauxhall-Opel, Fiat, Chrysler, Alfa-Romeo, Lancia, Maserati, Jeep and Dodge – reported that some of its European factories were on short-time or stopping altogether, entirely because of semiconductor chip shortages. It's a knock-on effect from the surge in demand for IT gear, as more people went WFH (working from home) during the pandemic that broke out 18 months ago.

The shipping and freight-forwarding sectors have also been hit; vessels and containers are in the wrong place at the wrong time, making it much harder to ship goods from Asia to mainland Europe. Since the vast majority of merchandise is made in China, problems with stock shortages are storing up in the supply line and this may affect the Christmas rush drastically.

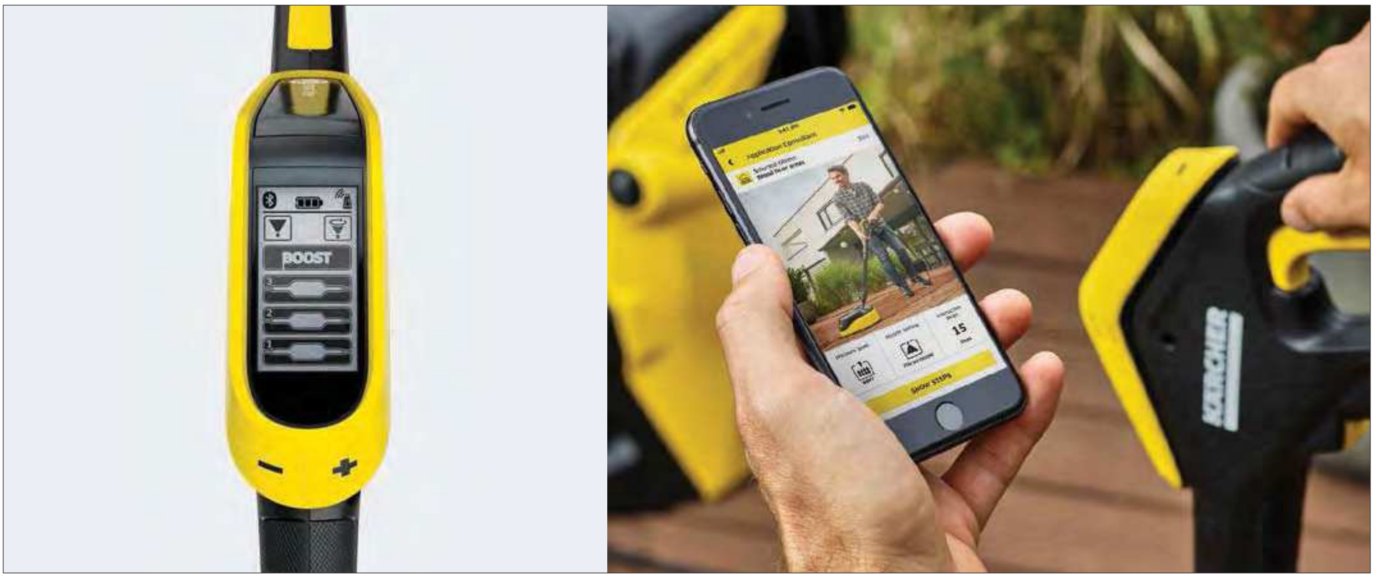
Lightweight but bulky items that take up lots of shipping volume are also proving more expensive to ship, which impacts the price and suddenly makes them uncompetitive. Even timber is in short supply: garden sheds, once a traditional haunt of many electronic hobbyists, have risen 50% in price. Corrugated cartons are in short supply too, with industry blaming consumers for stockpiling cardboard boxes in their garages instead of recycling them. Not to mention, of course, a European shortage of truck drivers to deliver the goods anyway, occasionally resulting in empty shelves in some supermarkets. There will undoubtedly be more to come before industry supply lines stabilise again next year. With Christmas in mind, the motto is 'buy early, or go without', or be ready to choose 'old stock' rather than the latest 2022 models.

A dromedarian money saver

Many will be sourcing their needs on Amazon, but it's best to 'tread carefully' and keep an eye on price variations rather than jumping in without hesitation. A smart buyer gets a feel for prices and lead times beforehand: Amazon's pricing varies substantially at times, especially for more expensive items. A Panasonic Hi-Fi sourced by the author jumped from £229 to £299 and back again: I just checked and it's back up at £299 again (out of stock!), a £70 premium. Fortunately, I'd waited until it was £229. A Bosch electric 'lawnraker' cost anything from £80 to £145 while another model was £221, then £115 in the space of a few hours before rising back to £200. Sometimes the same product appears multiple times online, but at different prices, so it's sensible to search around for the best offer, and maybe revisit the product page a day or two later.

As regular readers will know, I'm a big fan of the Cameliser price tracker plug-in which sends alerts when they drop to a desired level. The Cameliser should be a critical part of any Amazon shopper's armoury, so head over to **https://camelcamelcamel.com/** and set up an account if you haven't already done so – the savings can be substantial. Don't be swayed by 'recommended retail prices' either, as these are totally worthless; improbable 'discounts off RRP' feed the greed of buyers and urge them into snapping up a 'bargain'. For example, the Oral B Genius X twin toothbrush has an 'RRP' of £449.99 according to Amazon, but is actually selling for £175. Seasoned shoppers will get a feel for true 'street prices' before committing.

Many buyers also take time out to pore over the glowing product reviews left by supposedly happy customers.



Karcher's K7 Smart Control app adds more control and monitoring to their top-of-the-range pressure washers.

Starting last year, Britain's Competition and Markets Authority (CMA) opened an investigation into the use of fake reviews appearing on both Amazon and Google, being careful not to accuse them directly of potentially breaching UK consumer protection laws. There are entire dark web sectors dedicated to posting paid-for fake reviews into product pages. Sometimes they stick out like a sore thumb: odd linguistics, a strange non-English choice of words or near-identical wording paraphrased under different account names give us clues as to their authenticity. One Chinese IP camera vendor had nothing but 5* reviews on its website, all somehow re-quoted mysteriously from Amazon, while other Chinese vendors will offer buyers tempting discounts, an Amazon voucher or free products afterwards in return for leaving a glowing review.

One honest British reviewer slashed his rating of a Chinese vacuum to one star after claiming he got an email from a seller directed to his personal email address, asking him to remove his review or change it to 5 stars in exchange for a £40 refund. It does pay to groom the reviews, at least the more recent ones, checking for similar ones or positive ones all posted at around the same time.

Bluetooth earbuds have built-in rechargeable batteries and microphone, and cost around £20-£30 (Image: Amazon).

Some reviews are entertaining if not plain nutty: a buyer of some Covid-19 masks gave a 5* review even though the masks had not arrived: he'd received nothing but a future promise, and was 'happy with the outcome once I receive my items many thanks [sic]'. Another 5* review said, 'Prompt reply from the seller when item did not arrive on time, with a detailed explanation and new delivery date. Also offered discount/refund if not with me by 30 August. Very satisfied.' We will never know if it actually arrived. There's a 5* review for spray paint: 'Cannot review as not used yet, bought as stocking filler for Xmas,' while a 4* review of it opines, 'Seems good, but I have not used it yet'.

As many of you will know, it's possible to 'Ask a Question' about a product, which Amazon will then forward to existing customers. This can be a handy way of gleaning more information, especially when the product description is woolly or ambiguous. Not every existing customer realises that their answer will be published online regardless, though. Comically, one exasperated buyer of a health monitor replied 'I am absolutely fed up with would-be customers asking me questions about this product. Do as l did and buy one if you think you need one and give me a break please, l cannot be the only person ever to have purchased this item. Thank you.' I'll leave it there!

More App-lications

Smartphones apps are here to stay and have gradually become the focus of many an Internet user's concentration and lifestyle. As a sign of how our culture is being further skewed against civility, it was recently found that younger users are now so absorbed with their apps that the sound of a mobile phone's ringtone was considered rude, distracting and intrusive: they use social media instead and pick and choose what to say and when (and to whom). Not for them, the intrusive noise of a phone ringing, or actually talking to someone in person.

Consumer product manufacturers have been bitten by the app bug as well: many Braun electric toothbrushes now have Bluetooth and apps display your brushing regimen on a mobile screen, while the Braun iCheck 7 and some Omron blood pressure monitors also capture data onto a mobile phone.

Apps are becoming a common way of tuning in to streaming radio broadcasts, available with a voice command to a Google home hub or Amazon Echo: just ask Alexa to play your favourite channel. One wonders whether traditional radio-frequency broadcasts will eventually come under threat. Some portable radios have NFC (Near Field Communication) built in to enable the radio to act as a hands-free Bluetooth speaker: simply set up an audio stream on an app on your NFC-enabled phone, tap the radio with it and audio plays seamlessly on the radio speaker.

It's not necessary to spend hundreds on Bluetooth earbuds; some obscure Chinese-made earbuds offer remarkable value for under £20 and are worth a gamble. Their base holder unit recharges its internal battery, powerbank-style, and allows the earbuds'

The SPOT Gen4 is a portable messenger / SOS pager that summons help via the Globalstar LEO satellite network. Flexible tariffs are available for users.

own internal cells to be topped up when on the go. They may also have an internal microphone for mobile phone users. Many such earbuds are sold on Amazon or eBay under obscure brands that come and go overnight.

Other handy gadgets recently sourced online include a thermocouple with LCD readout and Bluetooth, which tells me when my oven has reached the desired temperature, ready to put the roast in, or I could measure how well a roast joint was cooking: search eBay for 'BBQ thermometer' – just the thing for a Christmas turkey. Some apps are optimised for phones rather than tablets or may not work with some versions of Bluetooth, and the old problem still exists of apps becoming incompatible in the future and rendering the gadget unusable/pointless.

There are plenty of bizarre apps too: Germany's Karcher pressure washer manufacturer has – you guessed – a mobile app to accompany its K7 Smart Control pressure washer. Fine if you don't mind the risk of getting your phone drenched or dropped on the concrete while you wash the car. An unlikely looking YouTube video of the app in action is at: **https://youtu.be/s3l0Ip36qmI**

Phone home

The new iPhone 13 is out. One interesting pre-launch rumour – that ultimately proved to be unfounded – was that it could incorporate satellite communication technology. The idea was that it could be used where 4G or 5G telecoms was unavailable, perhaps in emergency situations, sources suggested. It didn't happen this time – but maybe the satellite link-up will arrive with iPhone 14!

On a similar theme, but much closer to home (the author's anyway), it was reported in July that the Yorkshire Wildlife Trust is supplying SPOT Gen4 satellite GPS messenger devices to its teams of surveyors carrying out conservation work on Yorkshire's beautiful but remote peaty moorlands. In the event of an emergency in a remote area, the SPOT Gen4 can transmit an SOS message up to Globalstar's LEO satellites with just one button press. This alerts SPOT's 24/7 worldwide search and rescue service and sends GPS coordinates to summon help from local emergency services. More details at: **https://bit.ly/pe-nov21-sg4**

Other news

Hard on the heels of the early UK trials of hydrogen as a fuel source for heating homes in Spadeadam, Cumbria (see last month) comes news of a larger trial at the village of Winlaton near Newcastle in northern England. The village's 668 homes, church, primary school and several small businesses are in the first community to receive via the public network a mix of natural (methane) gas blended with up to 20% hydrogen. The trial started in August 2021 and will last around 10 months.

Project managers HyDeploy reminds us that hydrogen used to be a major component in 'town gas', gas created from coal and used widely throughout Britain before

Cooking on gas: the town of Winlaton is trialling the use of natural gas blended with up to 20% hydrogen. (Image: YouTube/HyDeploy)

'cleaner' North Sea natural gas was adopted in the 1960s (see: **https://en.wikipedia.org/wiki/Coal_gas**). Up to 60% of the gas (by volume) being used by consumers was hydrogen. More details available at: **https://hydeploy.co.uk/winlaton/**

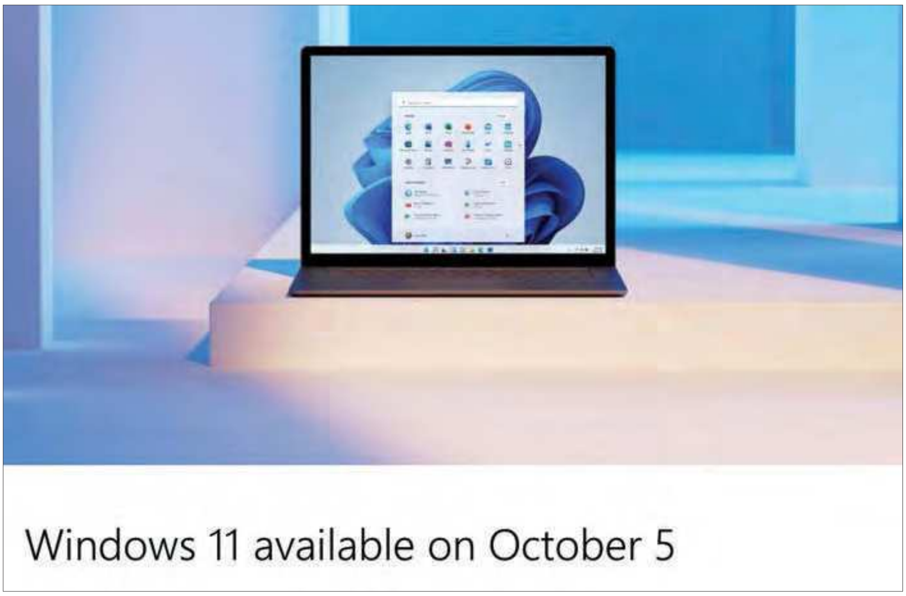
Britain's Lotus sports car maker has started work on building a new research facility for producing new electric vehicles. Four new models, including a sports car are promised in the next five years. As British petrolheads know, Lotus cars are renowned for going like a 'bat out of hell', so it was with a seemingly straight face that the location of the new site was announced as Wuhan, China.

Microsoft has announced the official launch date for Windows 11. Upgrades will be released gradually in a rolling programme, starting on 5 October. In the September issue I highlighted that Windows PCs would need a TPM 2.0 (Trusted Platform Module) fitted at motherboard level before Windows 11 could be installed. Many systems are already suitable, but it's best to consult your motherboard maker's website for news about compatibility. More news from Microsoft is at: **https://bit.ly/pe-nov21-w11**

Last month, I highlighted the new UKCA mark that is starting to appear on goods alongside the CE mark. The original deadline for implementing it was 1 January 2022 (see page 13, October issue) but this deadline has now been extended to 2023. A lack of testing capacity has been cited as the reason. All goods that require such a conformity mark before going on sale in the UK will now have an extra year to meet this requirement, but the UKCA mark is already starting to appear on equipment alongside other compliance logos.

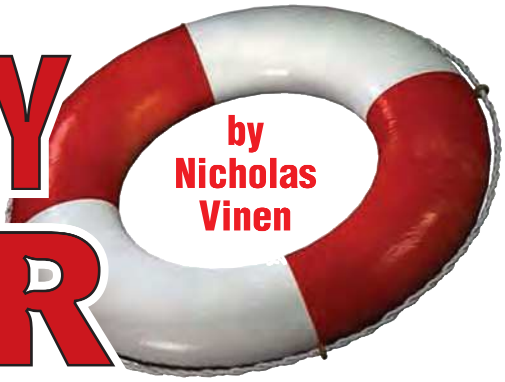
See you next month for more news and updates from *Net Work!*

The author can be reached at:
alan@epemag.net



Windows 11 – coming soon to a PC near you – but do check for motherboard compatibility.

Dual Battery Lifesaver



by Nicholas Vinen

This small board provides an easy way to protect rechargeable batteries from being completely drained if a device is accidentally left switched on. It can work with devices that run from a single battery, or two separate batteries. Both thresholds are fully adjustable, and it can handle several amps per battery, drawing just a few microamps when off.

Early next year we will be publishing a *Battery Vintage Radio Power Supply* project. One of the authors suggested that the low-battery cut-out section of the circuit could be useful on its own, and we had to agree with him.

So, ahead of the full project we have produced a separate PCB which contains just that portion of the circuitry.

It can be used with just about any device powered by 3.6-15V DC at up to 5A per output. Typically, it is configured so that both outputs are cut off if either falls below its individual voltage threshold.

However, it can also be reconfigured only to cut the outputs off if both fall below the threshold, or you can build a slightly simpler version for use with a single battery.

No heatsinking is necessary as the MOSFETs used for switching have minimal dissipation, around 100mW at 5A.

It has provision for an optional onboard power indicator LED, and also provides for an SPST (or similar) switch to disable the outputs, so that you can use a small, low-current switch as a power switch.

We previously published a very small single-battery *Lifesaver* in the September 2014 issue, which has been quite popular. Besides being small, its other advantage is that it can handle quite a bit of current; 20A or more.

However, it used quite a few SMDs and was a bit tricky to build, tricky to set up and had a limited adjustment range once built.

This version uses all through-hole parts and so is nice and easy to build, and not all that much bigger despite being able to handle two batteries. This one is also straightforward to set up, with a single trimpot allowing the cut-out voltage to be adjusted over a wide range for each channel.

Circuit description

MOSFETs Q1 (and Q2, if fitted) connect the supplies at CON1 and CON2 to the outputs at CON3 and CON4 when switched on. They are switched off, disconnecting the outputs, if either (or both) supply voltages are below defined thresholds.

When switched off, either via the switch S1 or due to a low battery voltage, the circuit only draws about 10µA from the higher voltage battery and about 2µA from the other.

Presumably, you would notice the device has switched off and either recharge the cells or swap them for fresh ones.

But if for some reason you forget and leave the device switched on, it would be several months before this minimal current drain could damage the cells. That's why this circuit was designed with a low quiescent current in mind.

When the power switch (S1) is closed, current can flow from whichever battery has a higher voltage, through small signal diodes D1 and D2 and then switch S1, into the input of REG1.

This is an ultra-low-quiescent-current, low-dropout 3.3V linear regulator. It powers micropower dual comparator IC1 and also serves as a voltage reference.

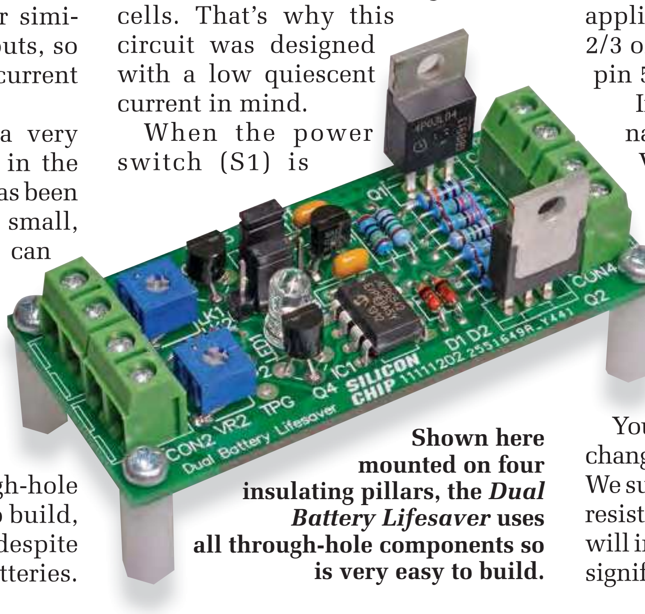
A fraction of this 3.3V reference is fed to the two inverting inputs of the comparators, at pins 2 and 6 of IC1. The fraction that is applied to those pins depends on the rotation of trimpots VR1 and VR2. These set the low-battery cut-out voltages, and they can vary the voltage at those inputs over the full range of 0-3.3V.

The actual battery voltages are applied to the non-inverting inputs, pins 3 and 5, after passing through fixed resistive dividers. While these two dividers use the same resistor values, they are in different orders. So around 1/3 of the CON1 voltage is applied to pin 3 of IC1a, while about 2/3 of the CON2 voltage is applied to pin 5 of IC1b.

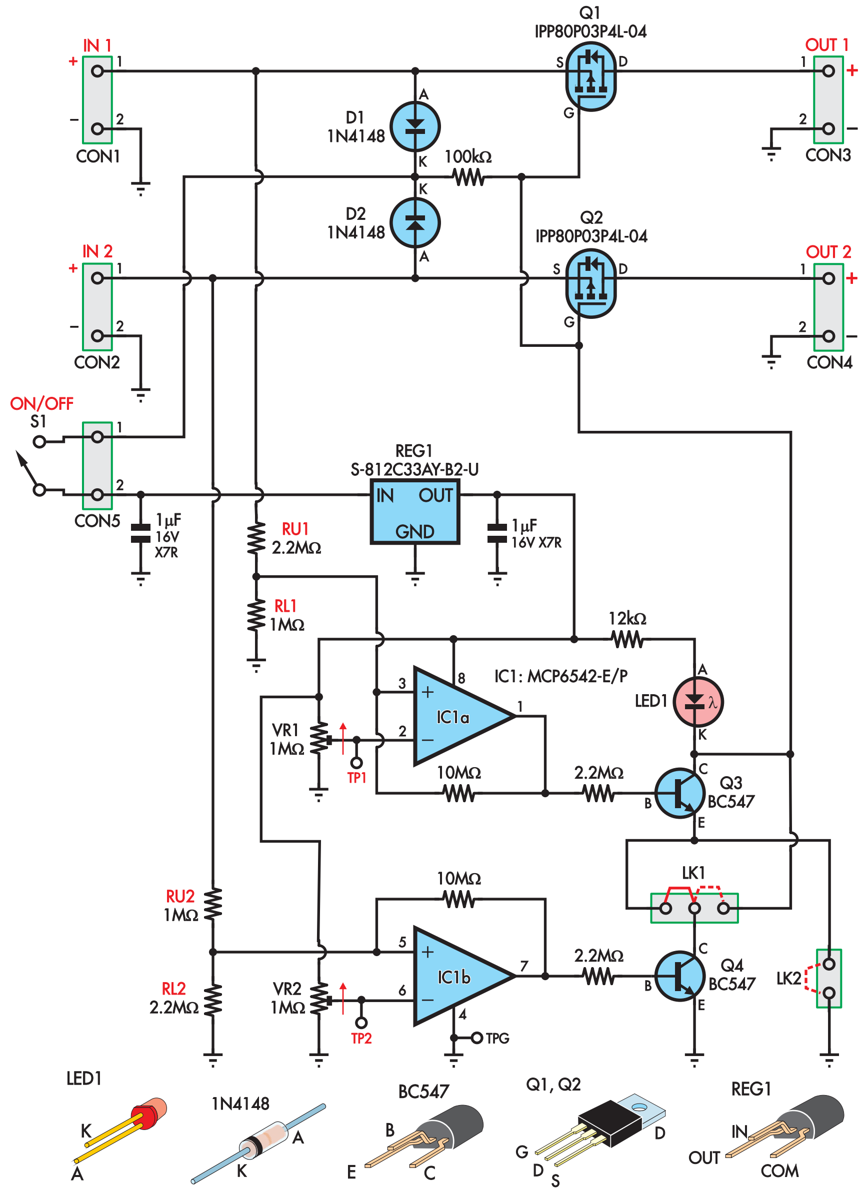
In combination with the nominally 3.3V reference and trimpots VR1 and VR2, you can set the switch-on voltage thresholds to anywhere from 0-10V for the CON1 battery, and 0-4.5V for the CON2 battery.

Those ranges suit Li-ion, LiPo or $LiFePO_4$ batteries with one or two cells in series, respectively.

You can easily change these ranges by changing the dividing resistor values. We suggest that you try to keep the total resistance around 3.3MΩ; lower values will increase the quiescent current, and significantly different values will alter



Shown here mounted on four insulating pillars, the *Dual Battery Lifesaver* uses all through-hole components so is very easy to build.



Dual Battery Lifesaver

Fig.1: the *Dual Battery Lifesaver* is built around micropower comparator IC1 and micropower regulator REG1, which supplies IC1 and also acts as the voltage reference. IC1 compares fixed fractions of the battery voltage(s) with the voltages at the pot wipers, and if the battery voltages are high enough, it switches on transistors Q3 and Q4, which in turn switch on MOSFETs Q1 and Q2.

the hysteresis percentage (as described below). Table 1 shows some possible combinations for other voltage ranges.

Hysteresis is provided by 10MΩ feedback resistors between the comparator outputs and non-inverting inputs. This has been arranged so that the hysteresis is a fixed percentage of the voltage.

The source impedance for the non-inverting inputs is 687.5kΩ in both cases (1MΩ||2.2MΩ). This forms a divider with the 10MΩ feedback resistor, giving a hysteresis percentage of 687.5kΩ ÷ 10MΩ = 6.875%.

So for low-battery cut-out voltages of, say, 3.3V and 6.6V, that would give you switch-on voltages 6.875% higher, or 3.525V and 7.05V respectively. The resulting hysteresis voltages are around 0.23V and 0.45V.

When both batteries are above their switch-on voltages, output pins 1 and 7 of IC1 are high, at 3.3V. Therefore, the base-emitter junctions of NPN transistors are forward-biased and so both conduct, pulling the gates of MOSFETs Q1 and/or Q2 low and lighting LED1 (as long as LK1 is in the position shown).

If either battery falls below its switch-off voltage, the corresponding transistor switches off and thus Q1 and Q2 switch off.

The high base resistors for Q3 and Q4 (2.2MΩ) are chosen because if one battery voltage is low but the other is high, current will still flow from the corresponding comparator output and this will increase the current drawn from the higher voltage battery (usually the one connected to CON1).

The 2.2MΩ base resistors are the highest practical values to minimise this, and determine the minimum value for LED's current-limiting resistor as 12kΩ. That means that LED1 has to be a high-brightness type.

If LK1 is moved to the alternative position and LK2 is fitted, rather than being connected collector-to-emitter, Q3 and Q4 are in parallel, collector-to-collector. In that case, if either battery voltage is above the defined threshold, the associated NPN transistor will pull the MOSFET gates low, and so both outputs will be connected to the inputs.

On/off switch

If you don't need a power switch on the supply, you can simply place a shorting block on CON5. CON5 is provided as a convenient way to switch power on and off, and you only need an SPST switch that hardly has to handle any current.

But with S1 off, there will still be a small quiescent current drawn from the two batteries due to the resistive dividers which remain connected. This is around 1µA for every 3.3V. That should mean the batteries last for around a year with the device switched off via S1.

If you need to reduce the battery drain further when off, you will instead need to use a DPST or DPDT

FEATURES & SPECIFICATIONS

- Two input/output pairs
- Individual low-battery cut-out voltage settings
- Passes through 3.6-15V at up to 5A per output
- Both outputs switch off if either (or optionally both) voltage falls below its threshold
- Fixed 6.875% hysteresis
- Quiescent current when off: around 10µA from the higher voltage battery and 2µA from the other

Parts list – Dual Battery Lifesaver

1 double-sided PCB coded 11111202, 70 x 32mm, available from the PE PCB Service
4 2-way terminal blocks, 5.08mm pitch (CON1-CON4)
1 2-pin header or polarised header (CON5)
1 4-pin header (LK1,LK2)
3 shorting blocks/jumper shunts (CON5,LK1,LK2)
1 SPST panel-mount switch (S1; optional)
4 tapped spacers (for mounting the board)
8 M3 x 6mm panhead machine screws (for mounting the board)

Semiconductors
1 MCP6542-E/P dual micropower comparator, DIP-8 (IC1)
[element14, RS, Digi-Key, Mouser]
1 S-812C33AY-B2-U micropower low-dropout regulator, TO-92 (REG1)
[Digi-Key, Mouser]
2 IPP80P03P4L04 P-channel logic-level MOSFETs, TO-220 (Q1,Q2)
[Silicon Chip Online Shop Cat SC4318 or element14, RS, Digi-Key, Mouser]
2 BC547 100mA NPN transistors, TO-92 (Q3,Q4)
1 high-brightness LED (LED1)
2 1N4148 small signal diodes (D1,D2)

Capacitors
2 1µF 50V multi-layer ceramic

Resistors (all 1/4W 1% metal film, unless otherwise indicated)
2 10MΩ 4 2.2MΩ 2 1MΩ 1 100kΩ 1 12kΩ
2 1MΩ mini horizontal trimpots (VR1,VR2) [eg, element14 108244]

switch to cut the battery connections to CON1 and CON2. That switch will need to handle the full load current for each battery.

Note that the batteries may still suffer from a small amount of self-discharge, so it's still a good idea to check and charge them every six months or so.

Construction

The *Dual Battery Lifesaver* is built on a double-sided PCB coded 11111202 . It measures 70 × 32mm and is available from the *PE PCB Service*. Refer now to Fig.2, the PCB overlay diagram, which shows where all the parts go.

As you read the following instructions, keep in mind that if you are using the device with a single battery, you can omit D1, D2, Q2, CON2, CON4, VR2 and some of the resistors – see Fig.3. You will need to add a couple of wire links, shown in red, which you might be able to make from component lead off-cuts.

Start by fitting all the resistors. While you can determine the value of a resistor by reading its colour bands, it's best to use a DMM set to measure ohms to verify this, as some colours can look like other colours under certain types of light.

If you are happy with the 0-10V adjustment range for the battery connected to CON1 and 0-4.5V for CON2, use 2.2MΩ resistors for RU1 and RL2, and 1MΩ resistors for RL1 and RU2, as shown in Fig.1. Otherwise, refer to Table 1 to determine the best resistor values to use.

With all the resistors in place, follow with the two small diodes, D1 and D2. These must be oriented with their cathode stripes facing as shown in Fig.2. Then fit comparator IC1. Make sure its pin 1 notch and dot go towards the top of the board, as shown. We don't recommend that you use a socket for reliability reasons, although you could if you wanted to.

Next, fit switch header CON5. You can use a regular or polarised header, or just solder a couple of wires to the PCB. If you want the supply always to be on, you can either place a shorting block on CON5 or solder a small wire link in its place.

The next step is to fit small signal transistors Q3 and Q4. They are the same type; ensure their flat faces lie as shown in the overlay diagram, and bend their leads out gently to fit the pad patterns. Follow with regulator REG1, which is in a similar package to those transistors, then install the two ceramic capacitors where shown.

Now mount the two trimpots, which are the same value. Follow with the four terminal blocks. Make sure that their wire entry holes face towards the outside of the module, and note that the side-by-side blocks are spaced apart and so should not be dovetailed; mount them individually.

Next, fit the two TO-220 devices, which mount vertically. Ensure that their metal tabs are oriented as shown. You could crank their leads so that their tabs are flush with the PCB edges, allowing heatsinks to be fitted later, but their dissipation should be low enough that heatsinks are not necessary.

All that's left is to solder the four-pin header shared by links LK1 and LK2 in place, followed by LED1. How you do this depends on what your plans are.

If you don't need an external power-on LED indicator, you can simply push it right down (with its longer lead on the side marked 'A', opposite the flat on the lens) and solder it in place.

If you want it to be externally visible, depending on how you will be mounting the board, you may be able to mount it on long leads and have it project out the lid of the device.

Or you could chassis-mount the LED using a bezel. You could then either solder flying leads from its leads to the PCB pads, or solder a 2-pin header (regular or polarised) onto the PCB and then solder leads to the LED with a plug or plugs at the other end.

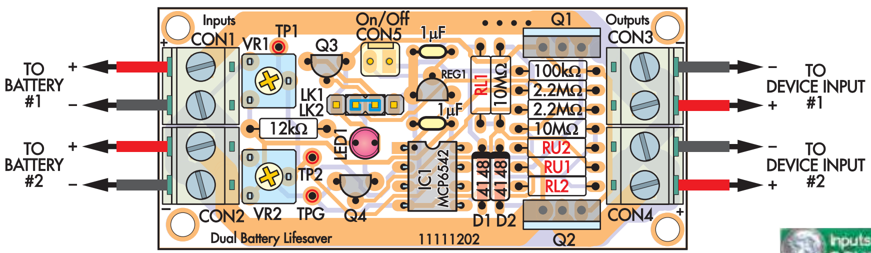


Fig.2: the PCB has been kept as small as possible while still being easy to build, handling a decent amount of current and providing for easy wire attachment and mounting. Assembly is straightforward but make sure that the IC, terminal blocks, MOSFETs, diodes and LED are correctly orientated. Use the component overlay above in conjunction with the same-size photo at right to assist you in component placement. Note that the values of RL1, RL2, RU1 and RU2 need to be chosen from the table overleaf.

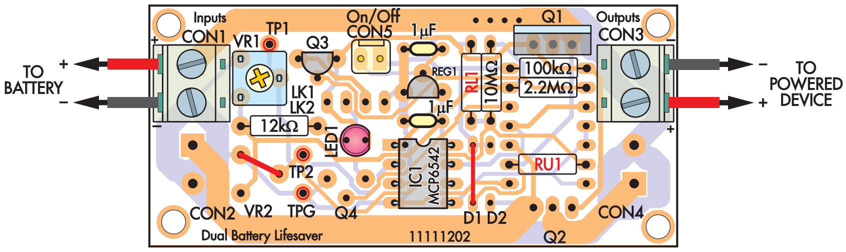


Fig.3: the same PCB can be fitted with fewer components if you only have one battery to protect, as above. Again, the two resistors shown in red need to be selected from Table 1. You will also need to add two wire links, shown in red.

Voltage range	Upper resistor	Lower resistor
0-4.5V	1.0MΩ	2.2MΩ
0-5.25V	1.2MΩ	1.8MΩ
0-6.3V	1.5MΩ	1.5MΩ
0-7.8V	1.8MΩ	1.2MΩ
0-10V	2.2MΩ	1.0MΩ
0-12.3V	2.4MΩ	820kΩ
0-15V	2.7MΩ	680kΩ

Table 1 – suggested resistor pairs for various cut-out voltage ranges.

Testing and adjustment

It's best to test and adjust the *Dual Battery Lifesaver* using a variable DC bench supply; ideally one with current limiting. The following instructions assume that you used the resistor values shown in Fig.1. If you changed them, you might need to alter the suggested voltages.

Place one shorting block on CON5 and another across the middle two pins of LK1/LK2.

Start by setting VR1 and VR2 at their maximum settings. If you've built the two-battery version, bridge the positive inputs together (you don't need to bridge the negative terminals as they are connected on the PCB). Now set your bench supply to around 4V and the current limit to a low value, then switch it off and wire up either input (CON1 or CON2) to the supply.

Switch the supply on and watch LED1. It should not light yet, and the current drawn from the supply should be low (under 1mA). If it's significantly higher than that, you could have a board fault, so switch off and check for short circuits and incorrectly located or oriented components.

If all is well, wind the voltage up to about 8V, then rotate VR1 anticlockwise until LED1 lights up. Then reduce the supply voltage slightly and check that LED1 switches off.

Now rotate VR1 and VR2 fully anti-clockwise, set the supply voltage to your desired cut-out voltage for whichever of the two is lower, then rotate either VR1 or VR2 clockwise slowly until LED1 switches off. Then increase the supply voltage to your other desired cut-out voltage; LED1 should switch back on. Rotate the other trimpot slowly clockwise until the unit switches off.

You have now set both battery cutout thresholds. If you want both outputs to switch off whenever either battery voltage drops below the threshold you've set, the unit is now complete.

If you only want it to switch off when both batteries fall below their respective limits, remove the jumper from LK1/LK2 and insert two jumpers on the 4-pin header side-by-side.

Reproduced by arrangement with
SILICON CHIP magazine 2021.
www.siliconchip.com.au

USB SuperCodec

Part 3

By Phil Prosser

Over the last two issues, we've introduced our new USB Sound Card, which we've dubbed the *SuperCodec*, and described its performance and operation in some detail. You would agree it offers extremely high recording and playback performance – so much so that our Audio Precision system can barely even measure its distortion! Now it's time to put it all together, and get it up and running.

It's best to build the SuperCodec in stages, checking after each stage is complete that everything you have just assembled is working properly.

Before starting, check that the PCB slides neatly into the case. This board is specifically made to fit a Hammond 1455N2201 case, which is sold by both Altronics and Mouser, as stated in the parts list published previously.

The part codes given are for the case with black end panels, as we have used, but note that Mouser stocks it in several other colours too. Now let's move on to mounting the components on the PCB.

Mounting the pre-regulators

Loading this section is pretty straight forward, as it is all through-hole. The PCB has a section marked to indicate this part of the circuit.

Referring to the PCB overlay diagram, Fig.17 and the photograph alongside (which you should do throughout the construction process), this section is at lower right.

Start by fitting the six resistors in this section, in the positions shown in Fig.17. Follow with the three diodes, D1 (1N4004) and D2-D3 (1N5822).

Note that they are not all aligned in the same direction. They have been oriented to minimise path length and radiation loops, so double-check that your diode cathode stripe is aligned as shown in the overlay diagram and on the PCB, before soldering each.

Next, install the seven MKT capacitors, which are not polarised, followed by the DC input barrel connector and the fuse clips, marked F1. Then you can fit the eight electrolytic capacitors; these are polarised, so their longer (positive) leads need to go into the pads nearest the + marks on the PCB and in Fig.17.

Make sure that the 2200µF 10V capacitor goes to the right, as shown by the smaller circle, with the larger 2200µF 25V type to its left. Also ensure that the two 470µF capacitors fitted in this section are rated at 25V; the 470µF 6.3V capacitor goes elsewhere.

You can then solder LED2 in place. For now, mount it vertically, with the base of its lens 10mm above the top surface of the PCB. Make sure its longer anode lead goes into the pad marked 'A'.

It's then time to solder switchmode regulators REG1 and REG2 in place. They have five pins; if yours are all in a row, crank them out with needle-nose pliers to fit the pad pattern on the PCB. They don't need heatsinks.

Now solder the inductors to the board. L1 and L3 are both bulky toroidal types while L2 and L4 are smaller bobbin types. Put a dab of RTV or neutral cure silicone sealant under each inductor to help hold it into place, and prevent vibration, as shown in the photo overleaf.

Oh no! I've put an IC in the wrong way around!

Everybody makes mistakes! So what to do if you got a part the wrong way around or in the wrong spot?

For through-hole parts, there are two ways to proceed. For electrolytic capacitors, you are best off using a solder sucker to get as much of the solder from the holes as you can, then judiciously heating one pin and 'pushing' the capacitor to lever up the component on the hole you have hot. Be careful and make sure that the leads are straight and will not tear the through-hole plating out as they go.

For op amps, resistors and diodes, the easiest and safest way by far is to cut the component from its leads, then remove the leads individually from the board and clean up the holes. It sounds wasteful, but this could save you tearing a track from your PCB, a lot of frustration and many naughty words.

Surface-mount parts are much easier to remove with a hot air gun. Set it to about 300°C, heat the part until all the leads come loose and use tweezers to lift it free of the PCB before the solder solidifies – job done. If you don't have one, you can alternatively heat each side of the part until it comes loose. If it's an IC, this is easiest to do if you join all the pins on each side with one big blob of solder. It's easy enough to clean up afterwards.

If you won't pay what your local electronics shop is asking for a hot air station, look on eBay; there are 'decent' hot air guns available at giveaway prices. Search for 'hot air SMD rework'; some are well under $100. These are brilliant for heatshrink work too. Note that it's best to keep these switched off when not in use!

Finally, add the 0Ω link; we used a length of 0.7mm tinned copper wire bent to form an Earth connection point, but you can also use a zero-ohm resistor, as shown on the PCB overlay diagram.

Testing the pre-regulators

Connect a voltmeter from ground (eg, either end of the 0Ω link) to the near end of FB12's pad. This is a convenient point to measure the –12V rail, as marked on the PCB.

Connect your 12V DC plugpack to CON1. The specified plugpack is a switch-mode unit capable of delivering at least 1.5A continuously. Switch on the power and look for the –12V rail coming up. Check that it is between –11 and –13V. Ours measured –11.5V.

Then move the red probe to the near end of FB8 (another empty pad) and check that the +6.5V rail measures 6.0-7.5V. Ours was close to 7V.

Finally, move the probe to the near end of FB11 and check that the +12V rail is OK. It will possibly be close to 11V due to the forward voltage drop of diode D1.

You can then disconnect the plugpack and proceed with the construction. If any of the readings are off, look for short circuits or bad solder joints. Also make sure that your plugpack has the current capacity to kick that negative regulator into operation.

Mounting the linear regulators

This section is positioned in the middle of the board and includes regulators REG3, REG4, REG6-REG8 and the surrounding components.

Start by loading all the ferrite beads in this section, FB8 through FB13. These can be any small ferrite that fits; they are there to offer a high impedance at high frequencies to keep the noise on the rails down.

If your beads came loose (as they often do), feed component lead off-cuts from the previous section through each one before soldering, or sections of tinned copper wire cut to length.

When soldering them, try to ensure they are held tightly to the board to prevent rattling. Dabs of RTV or neutral cure silicone under each one should help in that regard.

Next, fit REG7, the sole SMD regulator, while there is plenty of room around it. Follow with the ten resistors in this section, each being near one of the regulators. Then fit 1N4004 diodes D22-D29. As before, watch their orientations.

Then install the six MKT capacitors, followed by the 12 polarised electrolytics. As usual, make sure their longer leads go into the pads marked +. Keep in mind that they are not all oriented the same way. Again, with the two 470µF capacitors, they must be 25V-rated types, not 6.3V.

Finally, fit TO-220 package regulators REG3, REG4, REG6 and REG8. Three of these (REG3, REG4 and REG6) are mounted on small heatsinks. In each case, place a lockwasher over a 6-10mm M3 machine screw shaft, followed by a flat washer. Insert an insulating bush into the hole on the regulator tab, then feed the machine screw through this.

Slide a TO-220 insulating washer over the screw shaft, then feed the screw into the tapped hole on the heatsink. Do the screw up loosely, then drop the regulator leads into the PCB pads, while also lining up the heatsink posts with their mounting holes.

Make sure the heatsink is pushed down fully and solder its posts to their pads. You will need a hot iron to do this, and it also helps to add a little flux paste to the area around the bottom of the posts. Then hold the regulator vertical and do up the machine screw tightly before soldering and trimming the regulator leads.

Note that if you are using the recommended NE5532 op amps, in theory, you could leave off the heatsinks for REG3 and REG4. But they would run hotter. We recommend that you fit all three, just to be safe.

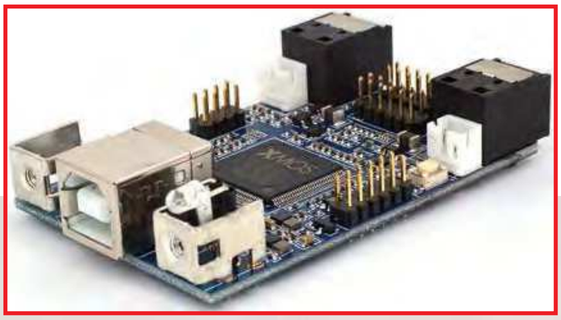
Testing the linear regulators

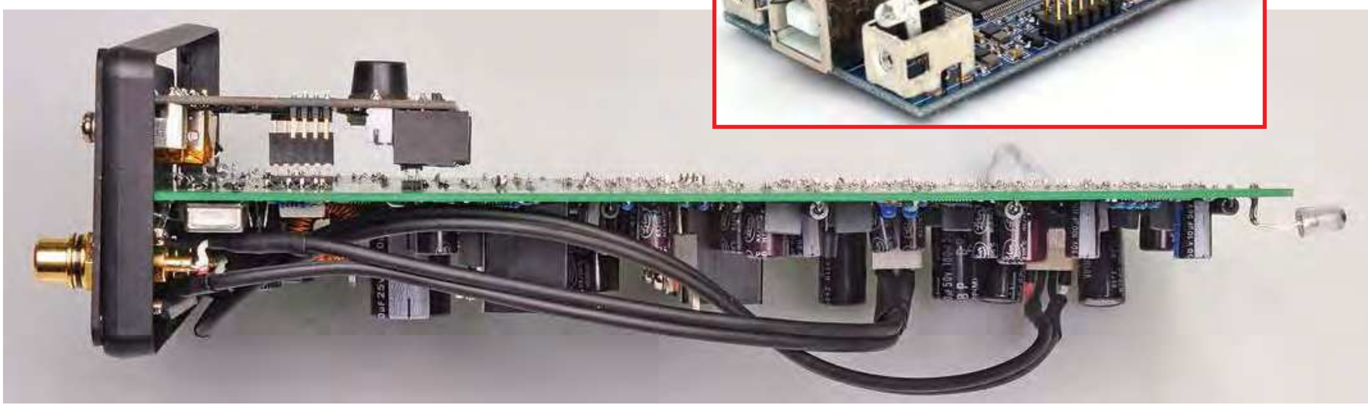
Reconnect the plugpack and measure the voltage at either end of FB9, on the left side of the PCB. You should get a reading in the range of 3.2-3.4V. Ours measured a touch over 3.4V – this is OK since the rail is currently unloaded.

Measure either end of FB7 for +5V; this should read between 4.75 and 5.25V. Then measure the voltage on the tab

Soldering tips

- Use a very fine tip on your soldering iron, the finest solder you have, with gel or liquid flux and a magnifying lens.
- Stay calm. Remember that if you only solder down one pin of each device at the start, you can easily melt this and move things around to get better alignment.
- Then by soldering a second pin, you can lock the part in place. Go easy on the solder and remember you can reflow one pin if you need to nudge the part a bit.
- Use less solder than you think you need. You will be surprised!





The completed project, albeit upside down! The main *SuperCodec* PCB 'hangs' off the rear panel with no connection at all to the front panel – even the power LED shines through a hole in the panel. The daughter board (at left of main pic and inset above) is the MCHStreamer USB to I²S interface which plugs into the two 12-pin sockets on the underside of the main PCB.

of REG6, which is the +2.5V rail. This should give a reading between 2.3V and 2.7V.

Next, check the voltages on the right-hand pads for the two 10Ω resistors in the upper-right corner of the board. The pad nearest the top edge of the board should be −9V (−8V to −10.5V) while the one immediately below should be +9V (+8V to +10.5V).

If there are any problems, check the plugpack output voltage – is it working OK, or has it overloaded and shut down? If it shut down, look for a short circuit on the board. If you have not used the specified plugpack, is that negative regulator overloading it on startup? Try a beefier supply.

Also check that all the diodes and capacitors are the right way around and all solder joints are good.

Once the power supplies are all up and running, you are well on the way. We can now mount the remaining SMDs without fear of damaging them.

Galvanic isolator and ASRCs

This section is in the lower left-hand corner of the PCB (Fig.17). Start by loading all the surface-mount capacitors in this section, then all the SMD resistors. The capacitors will be unmarked; while the resistors will be marked with codes indicating their values; you will need a magnifier to read them. In all cases, it's easiest to rely on what's written on the packaging, and fit one set of values at a time.

Adding a little flux paste (or liquid flux) on each SMD pad before placing the component will make soldering easier.

With the capacitors and resistors in place, proceed to solder IC6, IC7 and IC12. Note that pin 1 faces towards the bottom of the board in each case. Check and double-check the pin 1 marking on top of the IC package before soldering them, as they are difficult to remove. Again, flux paste will make soldering these parts much easier.

Given the proximity of the pins on these ICs, it's best not to worry about bridging pins when soldering them. Instead, check carefully after soldering using a magnifier, and use a dab of flux paste and some solder wick to clean up any bridges which have formed.

If you are lucky, you will have a microscope; if not, you can use a smartphone camera to zoom in close to the soldered pins and take a photo. This is a good way to check for hidden bridges between pins.

Next, mount the 4N28 and associated through-hole resistors, plus transistor Q1 and reset chip IC13.

Finally, install the headers for the MiniDSP MCHStreamer which go on the back of the board. These should be ESQT-106-03-F-D-360 elevated headers providing 10mm clearance, to ensure the MCHStreamer fits.

Testing this section

Follow the instructions in the text box below to install the driver and get the MCHStreamer running. Once you've connected it to your computer, check that it has been detected by clicking on the volume control and checking that it comes up with Speakers (MCHStreamer Multi Channels), as shown in that panel. Operating systems other than Windows will use a different method.

Once you've verified that it has been detected, unplug it from the computer and then fit it into the two matching sockets on the underside of the PCB. It should seat firmly onto the connectors.

Power the sound card back up and connect the USB socket to your computer. You then need to make sure that the MCHStreamer is selected as the current sound output.

To do this in Windows 10, left-click on the sound icon, and you will get a pop-up window as shown in the panel.

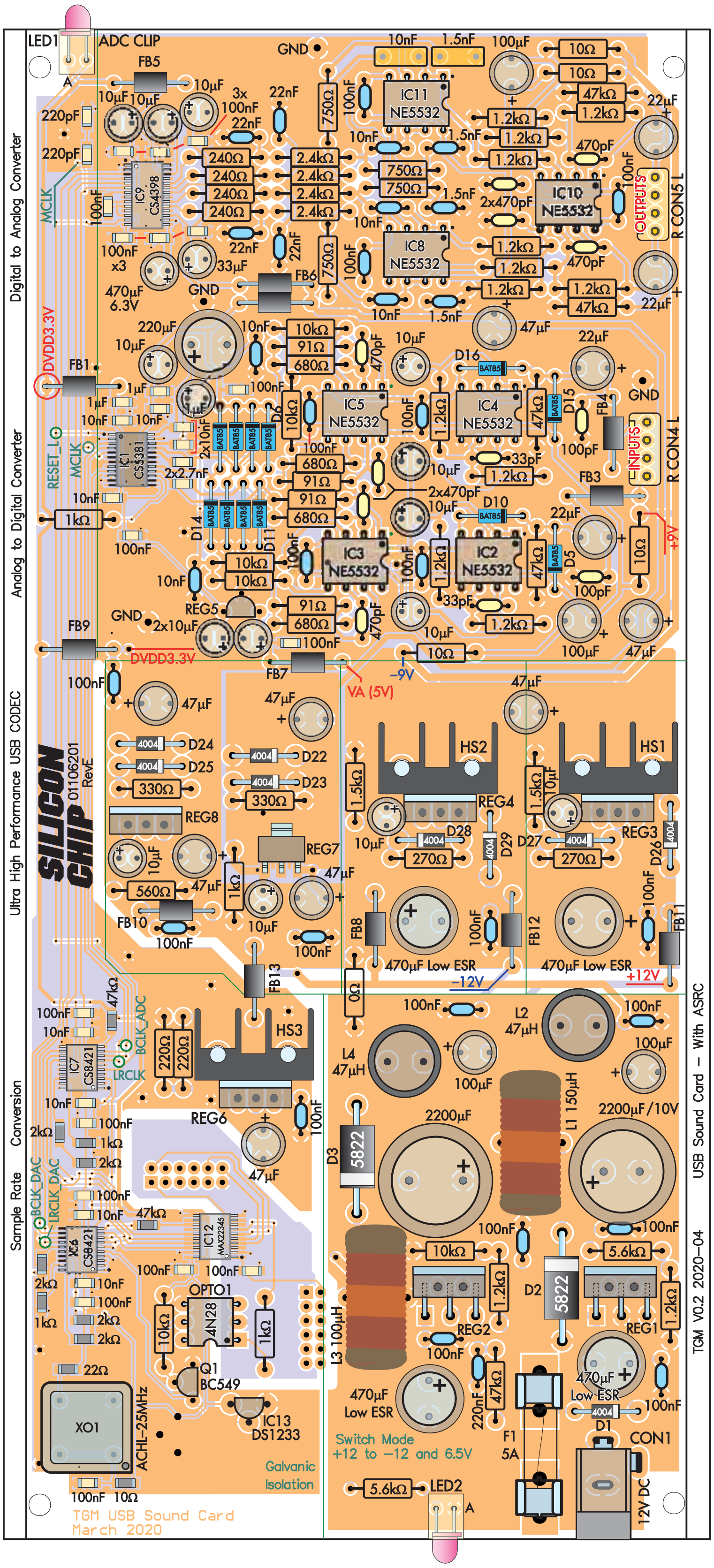


Fig.17: the PCB overlay for the *SuperCodec* shows all components in place. However, as discussed in the text, it's best to assemble the board section-by-section, allowing you to test each on completion and if necessary, fix any errors as you go. This overlay does not show the MCH daughter board, which plugs into the two header sockets (bottom left) on the underside of the main board.

If the MCHStreamer is already selected, then you're all set. Otherwise, left-click on the caret ('^') to get a list of available sound devices. You can then switch to the MCHStreamer.

Now play some music or another audio file. It does not matter what you choose, as we just want data to come out of the MCHStreamer.

Here's the matching PCB photo – there's a mix of through-hole and SMD components. You shouldn't have a great deal of drama with the resistors and capacitors, but some of the SMD ICs have quite fine pin spacing so you'll need to take your time with these. Any solder bridges between pins must, of course, be removed.

Check that the collector of Q1 (the pin towards the bottom of the PCB) goes high. Check for fixed-frequency square waves on the test points labelled on the PCB: MCLK (25MHz), BCLK_DAC (12.5MHz) and LRCLK_DAC (195.3125kHz).

If you have trouble, check the power supplies. Anything odd here needs to be tracked down. The individual power supplies will assist you in isolating power-related problems to a small group of components. Also check for solder bridges, bad solder joints (especially on SMD IC pins) and check those capacitors.

Loading the DAC and ADC sections

These sections are in the top half of the board and include all the remaining components. Start by fitting all the remaining surface-mount capacitors. Make sure that the two 2.7nF (2700pF) caps go where indicated as these are critical to good performance.

There is also one SMD resistor remaining (220Ω) so install that now. Then solder the ADC and DAC chips, IC1 and IC2. Orient both with pin 1 towards the top of the board, with the power supplies are at the bottom. Use lots of flux paste, thin solder wire and tack down one corner to allow you to align the IC before soldering the remaining pins.

Check there are no missed SMDs now, as after we load the through-hole parts, it is harder to get the soldering iron in there.

Now mount REG5, the LP2950-3.3V in a TO-92 package. Follow with the seven ferrite beads left, FB1-FB7, then all the rest of the through-hole resistors and diodes.

The diodes left are all BAT85s, but they don't all face in the same direction, so check the Fig.17 to make sure they're all installed with the correct orientation.

The seven op amps are next. They are all oriented with pin 1 towards the upper right-hand corner of the board. You can either solder sockets and then plug the ICs in, or solder the ICs directly to the board (which will give better reliability, but make it harder to swap them later).

Follow with all the MKT and ceramic capacitors, then the electrolytics. As usual, be careful to insert the longer leads of the latter into the pad nearest the + symbol, which varies in orientation for each capacitor.

The 470µF capacitor below IC9 is the 6.3V-rated type, to allow it to be closer to the chip, while the four 22µF capacitors are non-polarised types. (You could use 47µF or 100µF NP capacitors, as we did in our prototype, although we didn't find this to give any benefits.)

Now fit LED1, again with its lens 10mm above the PCB and with its anode to the pad marked 'A'. Then fit polarised headers CON4 and CON5, and the PCB assembly is complete.

Testing this section

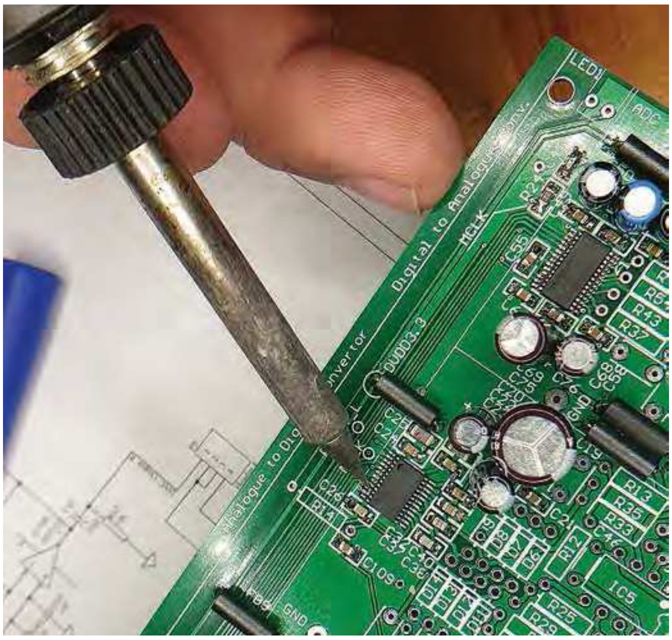
Check that there are no missing parts on the board. If there are, look them up and fit them. Also check your soldering to make sure it's all good, especially on the SMD ICs. It's best to clean flux residue off so you can get a good look at the solder joints.

Now apply power, without the sound card plugged into a PC. It is not even necessary that the MCHStreamer is plugged in, but this does not matter as it is isolated from the rest of the board!

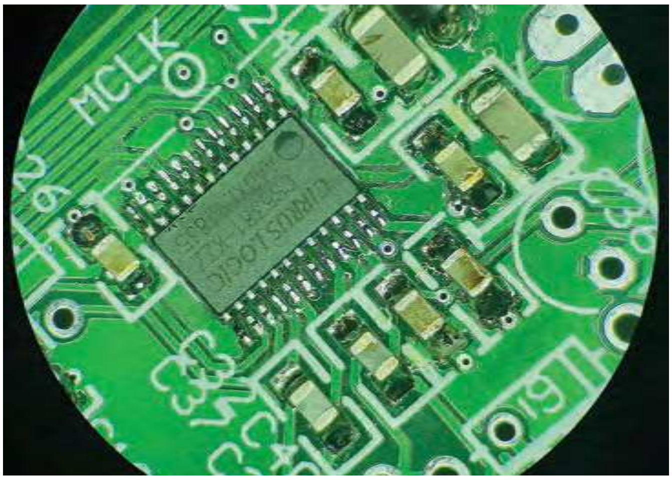
Connect the ground of your DVM to a convenient ground point. We soldered a PCB pin to a few of the larger GND vias; there is a convenient one just above the 3.3V regulator. But you can also just hold the black probe in one of those holes.

Apply power and board and re-check the 3.3V rail, the +5VA rail, the +2.5V rail and the ±9V rails, as before. This is to make sure you haven't introduced any short circuits across any of the rails.

Assuming these are OK, and there is no part emitting smoke or getting hot, we can proceed. If something is wrong, follow the usual checks for solder bridges, especially on the ADC and DAC where the pins are close to one another. Also check the component orientations.



You can solder fine-pitched SMDs with a standard iron.



If you are lucky enough to own one, a PCB microscope can help identify problems in soldering – or alternatively, confirm you've done a great job! If you don't own one, you could try using the camera in your smartphone to take close-up shots which you could then enlarge via your photo-editing software to help you spot any 'oopses'. Don't have photo-editing software? Try downloading GIMP (it's free!).

Now it is time to get into some of the fun tests. Switch the power off, plug the MCHStreamer into the sound card and the PC, then plug its outputs into some sort of amplifier. Power it back up and play some sound (eg, music). Then you can check that you get appropriate sounds from the amp!

Alternatively, you may choose to put a scope on the output(s) and look for the audio. Assuming that works, connect a stereo RCA-RCA cable from the outputs to the inputs, play some audio and then simultaneously make a recording. Check that the recorded sound file matches the playback audio.

If any of these tests fail, check the data paths from the MCHStreamer to the DAC and ADC chips. This is ideally done using a scope with its timebase set to 50ns/division. Check the MCLK, LRCLK, SDATA, BCLK and RESET lines.

If the RESET line is not high, the MCHStreamer is probably not connected properly. Is its light on? Why not?

Check the clock and data lines on the USB card side of the galvanic isolators – they should be there if they are on the PC side. If not, why not?

Metalwork

If you are using the recommended case, the Hammond 1455N2201, there is refreshingly little metalwork to do. Cut and drill the front and rear panels as shown in Fig.18.

The front panel has a single hole for the ADC Clip LED. The rear panel has cutouts for the USB input, power input, power LED and four RCA connectors.

Rectangular holes are always a nuisance to cut. As these are small, we recommend marking the outlines on the panel, then drilling a series of small holes around the inside perimeter with a 1.5-2.5mm drill bit. Keep the holes close together and err on the side of drilling well inside the marked square, rather than touching the outline.

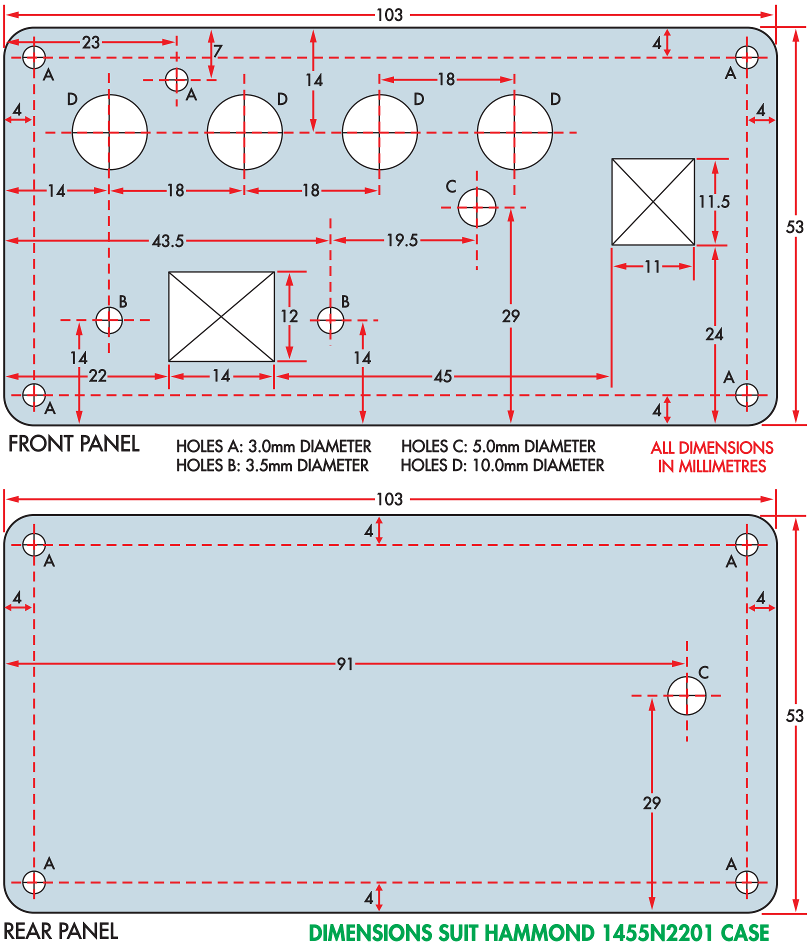
Once you have broken free the tab of aluminium from the middle of the hole, use a square or triangular file to neaten the holes to the required square. Touch up the edges with black paint or at a pinch, a marker, to make this neat.

To finish the front panel, stick a small rubber stopper on the front panel in a location that will ensure that the *SuperCodec* is held tightly against the rear panel. This will minimise strain on the MCHStreamer connectors when power is being plugged in and out. If you have foam tape, a thick layer of this along the edge of the PCB would also work fine.

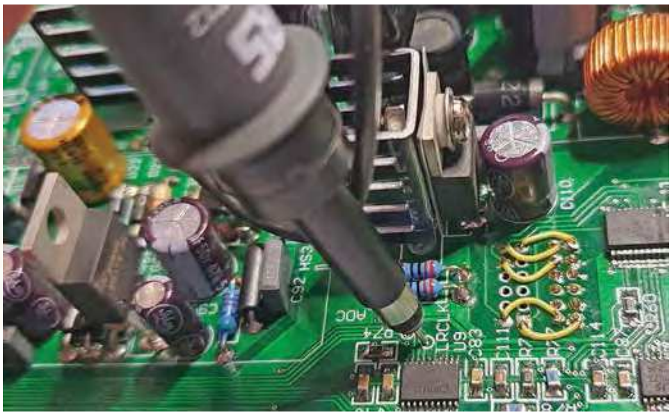
The *SuperCodec* slides into slots in the case and is held tight by the rubber stopper at the front, and the MiniDSP MCHStreamer, which is attached to the rear panel.

Final assembly

You need to make up some cables using the two polarised header plugs and matching pins, two 30cm lengths of



A view of the board with the power supply sections completely assemble and nothing else. This way, we can check that all the supply rails are correct without risking any damage to the expensive chips they will be powering.



Test points are provided to help you verify correct operation.

figure-8 screened cable, the four RCA panel-mount connectors and some heatshrink tubing. The result is two cables, each with two RCA connectors at one end and a four-pin header plug at the other.

At the header ends, start by separating the two channels of coax, then stripping 25mm of the outer sheath of each, exposing the shield braid. Tease the inner conductor from the braid, and strip the end by 5mm. Twist the braid wires together into a neat bundle.

Next, cut two 20mm lengths of heatshrink, one around 3mm diameter and one 5mm. Slide the 5mm piece over both the shield braid and central conductor. Do not shrink this yet.

Slide the 3mm heatshrink over the braid; there ought to be 4-5mm of wire protruding. Shrink this down. Slide the 5mm heatshrink sleeve to cover about 3mm of the junction where the braid and inner core separate, then shrink it down.

Present the bare wires to a crimp pin. You need to trim off excess braid wire, so that the strain relief crimp (at the back of the pin) will go over the braid, with about 3mm of wire in the main crimp as shown.

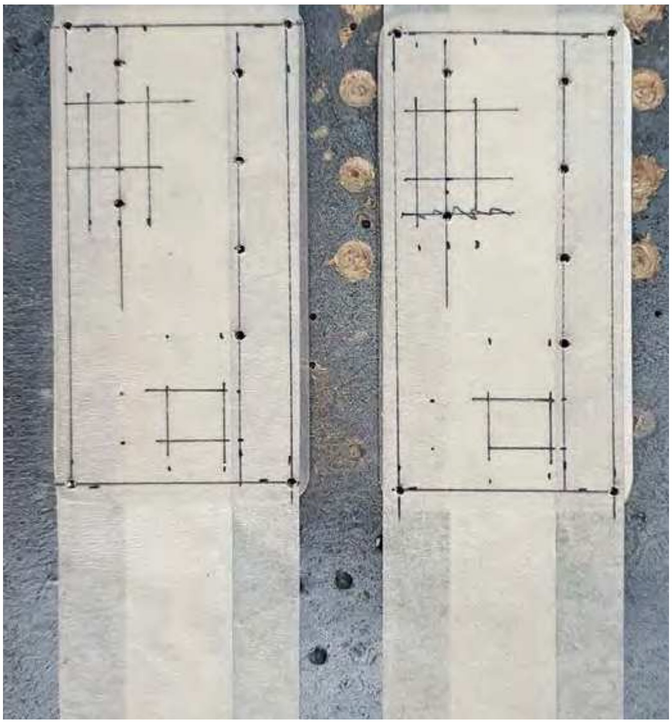


Fig.18 (opposite): drilling/cutting diagram for the rear and front panels (most holes are on the rear panel with only one LED hole requrired on the front) – it is available for download from the November 2021 page of the *PE* website. Above are the rear panels (yes, we made two prototypes!) with masking tape holding down the panels and also providing a handy means of marking out the holes required.

Crimp the middle section using sharp-nosed pliers. Make sure the crimping doesn't cause the pin to splay out so wide that it will no longer fit into the plastic block. Then add a tiny amount of solder to the crimp, being careful not to allow it to wick down to the connector spring. Then crimp the strain relief onto the heatshrink around the braid.

Next, strip back 3mm from each of the inner conductors and crimp and solder to another pin as above.

Now push the pins into the header plug. The shield braids go into the middle two pins, with the left and right signals on the outside. You will feel and/or hear a click when they seat properly.

Then take the two pairs of RCA socket and mount them to the rear panel using the supplied plastic insulating washers, to isolate them entirely from the back panel.

As before, separate the twin coax cables into left and right wires, and strip back 25mm of the outside insulation. Cut two more pieces of 5mm and 3mm heatshrink and twist the braids, insulate them and then shrink the braid and overall sleeving, as with the header end. You can then solder the input and output wires to the RCA connectors, as shown in the photo above.

The two things to check for are that the input pair and output pair are wired to the same cables and that the left (white/black) and right (red) sockets are wired to the appropriate pins on those headers – see Fig.17. Check the orientation of your polarised headers to determine which pin will go to the left signals on the board, and which goes to the right.

You can make these checks most easily by plugging the cables into the sockets on the board and then using a DMM set to continuity mode. Probe from the centre of each RCA connector to the pins on the headers (through the slots in the plastic housing), to verify that each one goes where it should.

Mounting the USBStreamer

The USBStreamer needs to be isolated from the case of the *SuperCodec*. This optimises the effectiveness of the galvanic isolation and improves hum rejection.

This is achieved by using TO-220 bushes on the M3 machine screws that attach the USBStreamer through the rear panel, and placing fibre washers on the inside of the rear panel, between it and the USB Streamer brackets.

See the photo overleaf, where you can see the insulating washers under the screw heads on the rear panel.

This is required to prevent ground noise from the USB-Streamer card being conducted through the case and injecting itself into the very sensitive ADC stages.

While you're doing this, something to note is that the mounting lugs on our MCHStreamer board were not lined up properly. We reckon this was due to sloppiness on the part of whoever (or whichever robot) soldered the threaded standoffs to the board. This can result in the MCHStreamer sockets looking crooked on the rear panel.

After crimping and/or soldering the crimp pins to the end of the wires, push them into the plastic housings and they will click into place.

A 10nF capacitor between the input grounds and rear panel earth lug minimises hum pickup.

If, after mounting your board to the panel, it is noticeably crooked, all you have to do is pack one of its mounting screws on the inside of the panel with an extra fibre washer or two. That should straighten it right up.

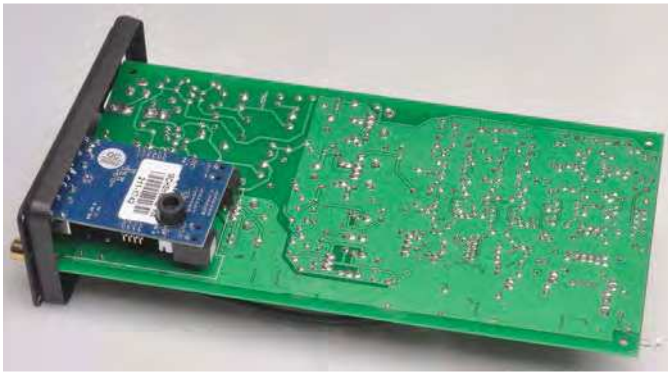
It's also very important that you stick a 7.5-8mm tall rubber foot on the bottom of the MCHStreamer board as shown in our photos. As this board is only attached to the main board via headers, and it's only mechanically mounted at one end (to the rear panel), it's possible for its pins to lose contact due to shock or vibration.

The rubber foot rests on the bottom of the case and holds the far end of the MCHStreamer up so that the headers can't come out of their sockets.

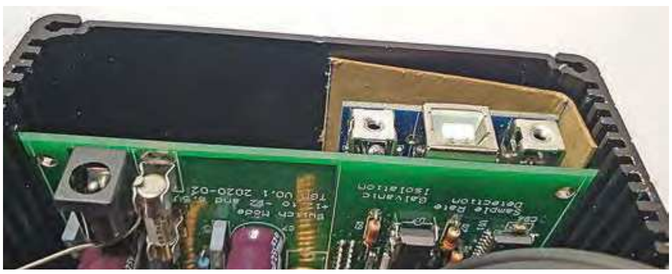
When you slide the PCB into the case, the foot should press against the bottom and provide a little extra resistance to sliding the board in, but not so much that it becomes impossible. This is how you know that it's providing enough force to hold the boards together.

Grounding

If you want to get the 50Hz hum down below –120dB, as we achieved in our prototype, Earthing is very important. To be honest, in testing this, we found that even the slightest change in the configuration can cause changes of 10dB or more. That just shows how difficult it is to achieve such performance.



The pre-assembled USB Streamer PCB plugs into the two 12-pin header sockets on the underside of the PCB.



A piece of insulating material such as Presspahn, located as shown here, will ensure the MCHStreamer is always isolated from the case.

In most tests of amplifiers, you will need the galvanic isolation that the system provides to measure really low noise floors. Where super-low noise is critical, you might find with some system configurations that the earth of the PC does need to be tied to the device under test to eliminate induced 50Hz signals being picked up. This will require experimentation with your overall setup.

You should establish the noise floor with no signal to the unit under test before running any tests.

You should have a 10nF MKT capacitor left over, which was specified in the parts list (in part one) but not used on the board. You should also have a 10mm M3 machine screw, three locking washers, a solder lug and an M3 nut, again specified in the parts list.

Cut a 6mm length of 3mm-diameter heatshrink, then mount the M3 machine screw through the hole in the rear panel with a locking washer either side. Place the solder lug on top, then the third locking washer and finally the M3 nut. Do it up tight.

Put the 6mm heatshrink over the capacitor leg, and solder this to the solder lug. Then solder the other lead of the capacitor to one of the shield braid wires of the output connectors.

Tip: if you envisage using this as a measurement system, put a solder lug on the outside of the case as well. This can use the same screw. As we found in our tests, access to the unit's ground can be useful in some cases to minimise overall system noise. Adding this while building it will be a lot easier than adding it later. Slide everything into the case once it is all working, then mount the panels and you are set!

If you envisage this device being moved around a lot or vibrated, then you might want to add a piece of Presspahn or Elephantide as shown below-left. This is optional.

Using it

If you want to use the *SuperCodec* for playback, you can use just about any audio software. But if you want to take advantage of its full capabilities, you will need high-resolution

Here's what the back panel of your *SuperCodec* should look like when finished. Note the comments in the text re the grounding/insulaton of the sockets to avoid ground loops.

Getting the USB interface up and running

First, you'll need to install the driver on Windows or macOS. Log onto the MiniDSP website with the password you used to buy the MCHStreamer, and navigate to the download section. Download the driver for the MHCStreamer. Follow the instructions to install this from the MiniDSP website, which in summary are:

1. Plug the MHCStreamer module in via its USB cable. It does not need to be plugged into the sound card PCB; it can just be on your workbench (but make sure it's on a non-conductive surface). It is powered from the computer via the USB cable
2. Our Windows 10 PC popped up a window saying it was 'setting up the MCH Streamer', then a second window saying 'the MCHStreamer was ready to go'
3. Extract the contents of the ZIP file you downloaded from their website
4. Navigate to the 'Drv_DFU\WinDrv' subdirectory and double-click on the installer, which in our case was named 'miniDSP_UAC2_v4.67.0_2019-08-15_setup.exe'
5. When asked if you want to allow the App to make changes, click 'Yes'
6. Follow the prompts in the installer, selecting defaults including file locations.

The SuperCodec should now be up and running.

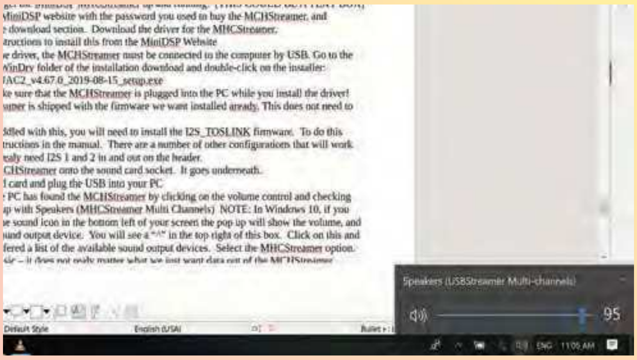
To set the sampling rate, right-click the speaker icon in the taskbar, usually in the bottom-right corner of the screen. Select 'Open Sound Settings' and check that the system has 'Speakers (USBStreamer Multi Channels)' selected as the output device (see below).

This should automatically be selected. If not, select it. Then click on 'Device Properties' in blue, just below the device selection pulldown box. In the new window that appears, look for 'Additional Device Properties', again in blue. Click this.

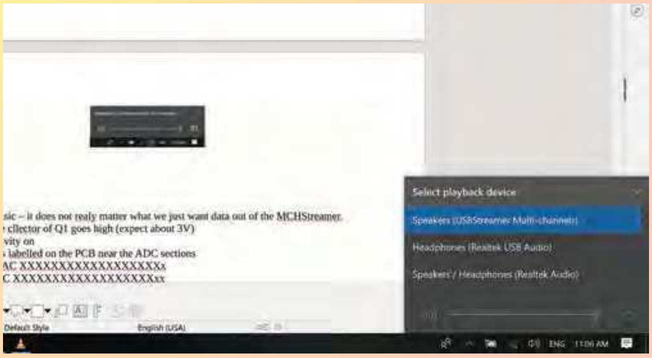
In the pop-up window, go across to 'Advanced'. Here you can select your sampling rate, and also click a 'Test' button. We recommend selecting '24 bit, 192,000 Hz (Studio Quality)'. Then click 'Apply' down the bottom left.

While the download package includes the firmware, the MCHStreamer is shipped with the firmware already installed. This does not need to be changed. If you have fiddled with this, you will need to install the I2S_TOSLINK firmware. To do this, follow the instructions in the manual.

Several other configurations will work for us, as all we need are I^2S channels 1 and 2 in and out on the header.



Once the drivers are installed and the MCHStreamer is plugged into your PC via USB, it is set as the default output device automatically.



If for some reason it isn't, you can select it from the list of available audio output devices by clicking the caret on the right.

content such as 96kHz or 192kHz, 24-bit FLAC files, along with a player that can properly decode such files.

For recording, we suggest that you try the free software package called Audacity (**www.audacityteam.org**). It is available for Windows, macOS and Linux and can take advantage of the Card's full capabilities.

For audio analysis use, such as measuring distortion (THD+N or THD), signal-to-noise ratios (SNRs), frequency responses and so on, various packages are available. We use audioTester (**www.audiotester.de**).

A section of Kapton tape on the USB socket ensures it can't short to any components on the main PCB.

This is 'shareware' so you can download and install it for free, but you can only use it for a limited time without paying for it. It only costs €39 or about £35 for the full version.

We recommend this software because it is easy to use and has many comprehensive features that are ideal for testing audio equipment. That includes a low-distortion sinewave generator, spectral analysis with automatic display and calculation of the signal level and total harmonic distortion (THD) and much more.

Reproduced by arrangement with
SILICON CHIP magazine 2021.
www.siliconchip.com.au

Coming up: a balanced attenuator add-on

Phil Prosser has designed an add-on board for this project which adds balanced inputs and a switched attenuator with settings of 0dB, 10dB, 20dB and 40dB.

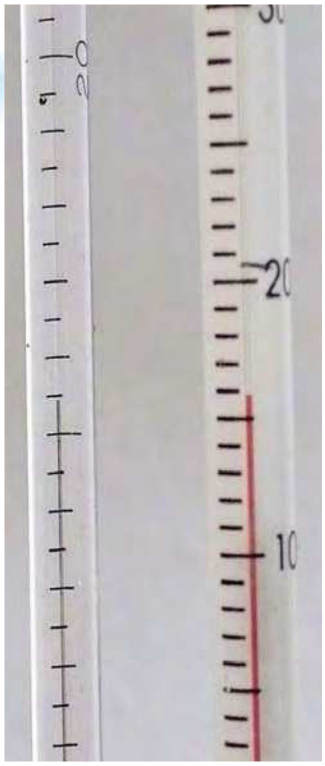
This add-on board greatly improves the flexibility of the SuperCodec when used as a measurement instrument, and only slightly degrades its performance.

If you'd like to build this add-on board, go ahead and start building the SuperCodec but don't fit the headers for the MCHStreamer just yet, and don't drill the case end panels either, as both the MCHStreamer and the main PCB are mounted slightly differently to make room for the add-on board.

The article describing this add-on board will be published in the next two issues.

Low-cost, high-precision

Thermometer Calibrator



By Allan Linton-Smith

Many digital thermometers have readouts with a 0.1°C resolution but rarely are they accurate to within ±0.1°C. Despite their claims, some can be several degrees out, giving a false sense of accuracy. This simple, low-cost thermometer checker will tell you just how accurate your thermometer is. In some cases, you may even be able to adjust the thermometer to be more accurate.

There are many reasons why you might need an accurate thermometer. Checking to see if someone (especially a child) has a fever is an everyday use case. This requires pretty good accuracy, as the difference between a normal-but-elevated temperature (as can happen when someone has been exercising or crying for example) and a fever is just fractions of a degree.

Or maybe you're a keen chef, and you want to use processes like tempering chocolate, where you need to heat the chocolate to a temperature within a fairly small window, eg, 31-33°C.

A 1°C error could mean that you think you're in the window, but you aren't, and the batch could be ruined.

Whatever the reason for using it, if you have a thermometer that will read out to within 0.1°C, you want to know if it's at least 'in the ballpark' before you trust its display fully. This simple device allows you to do that.

In some industries such as food manufacture, storage and distribution, temperatures are critical. This is especially true when food poisoning is a potential problem. So in these cases, it is essential to check that your thermometers are accurate. A device like this is therefore invaluable.

This design is based on the LM35CAZ IC, a temperature sensor that has been available for some time now. But it has really come down in price lately. If managed correctly, it can be expected to give readings within ±0.2°C at 25°C.

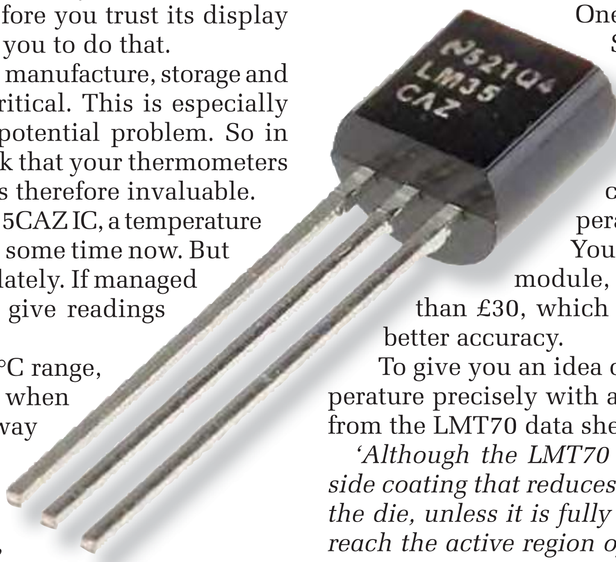
It works over a –40°C to +110°C range, but its accuracy is not as good when reading temperatures further away from room temperature.

It's worth building this yourself because other devices with precise temperature readings, eg, ±0.1°C, are not commonly available and are very expensive. For example, the Fluke 9142 and 9143 are excellent calibrating instruments with a display accuracy of ±0.2°C over their full range, but we recently spotted a *used* one for sale for over £3,000! And new? Don't even ask!

Some say that glass thermometers are very accurate. Usually, their accuracy is accepted as ±0.5 divisions, which typically translates to ±1°F or ±0.5°C, but they are becoming quite rare.

And they are still susceptible to reading errors, some of which are described in the side panel.

When designing this device, we found that there are a few temperature sensor ICs that are even more accurate, such as the LMT70, but we decided against using this (for now) for a few reasons.

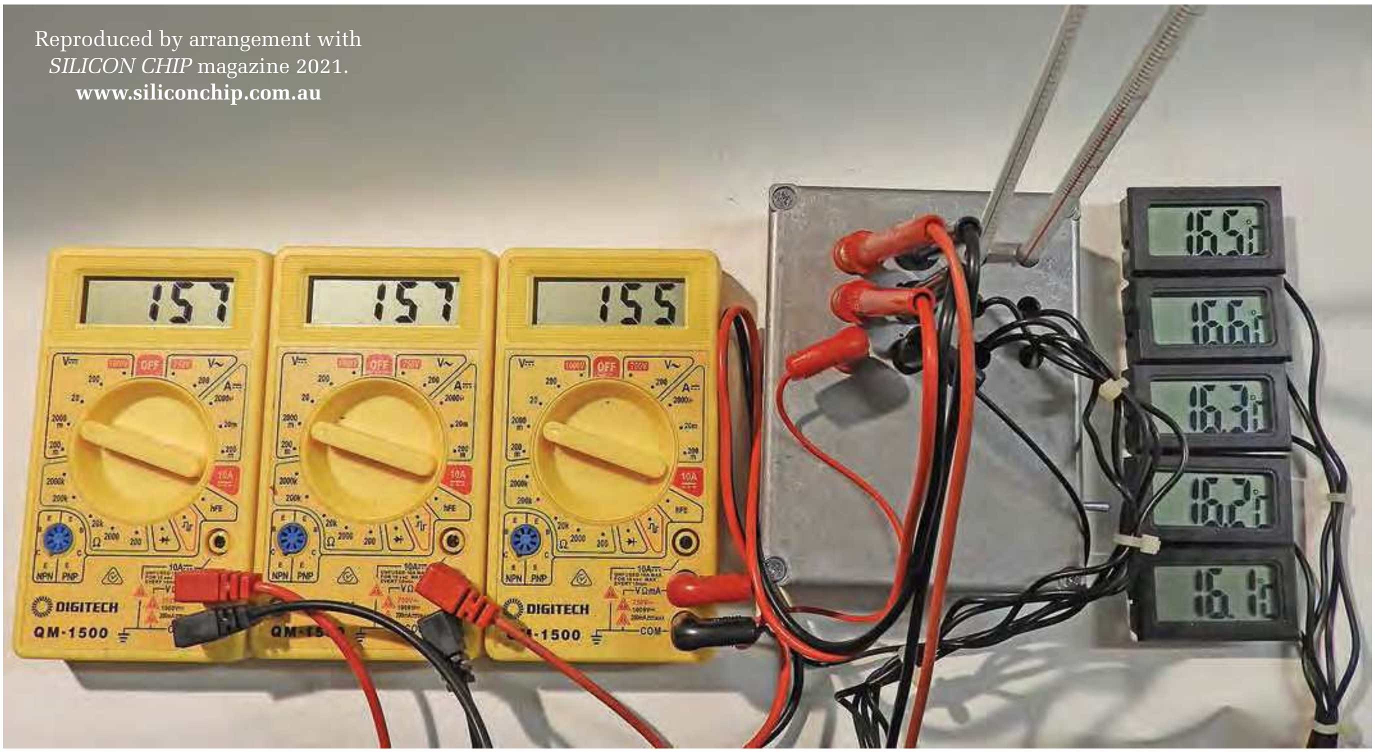


One is that it only comes in a tiny SMD package (0.94 × 0.94mm) which is hard to work with. Another is that its output voltage is non-linear and requires a lookup table or polynomial curve-fitting to convert to a temperature reading.

You can buy them pre-soldered to a module, but these test boards cost more than £30, which is not worth it for only slightly better accuracy.

To give you an idea of how hard it is to measure temperature precisely with a digital sensor, here is a passage from the LMT70 data sheet:

'Although the LMT70 package has a protective backside coating that reduces the amount of light exposure on the die, unless it is fully shielded, ambient light will still reach the active region of the device from the side of the



The three DMMs are reading the outputs of the LM35s but we have also inserted the probes of five cheap digital thermometers and two lab-grade glass thermometers (see opposite) into the device. The cheap thermometers have a 0.5°C spread, quite a bit larger than the 0.2°C difference between the LM35s.

package. Depending on the amount of light exposure in a given application, an increase in temperature error should be expected.'

'In circuit board tests under ambient light conditions, a typical increase in error may not be observed and is dependent on the angle that the light approaches the package. The LMT70 is most sensitive to IR radiation. Best practice should include end-product packaging that provides shielding from possible light sources during operation.'

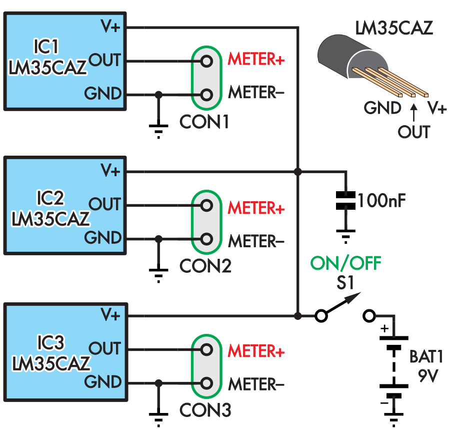
Circuit details

The LM35CAZ is a precision integrated-circuit temperature sensor with an output voltage linearly proportional to the temperature in degrees Celsius.

It requires no external calibration or trimming. It is low in cost, can operate on a wide variety of single supply voltages and has low self-heating.

There's little to the circuit besides three of these devices, and a battery to power them, as shown in Fig.1. IC1-IC3 can run from a wide supply range of 4-20V, so they are very well suited to be powered from a 9V battery.

Fig.1: the circuit couldn't be much simpler; it's just the three LM35s with a shared 100nF bypass capacitor, power switch S1 and a 9V battery for power.



The output of each device can be measured by a multimeter connected across one of CON1-CON3, set to its 1V range or thereabouts (ideally, with 1mV resolution). IC1-IC3 have a nominal 0V output at 0°C, rising by 10mV/°C.

So, for example, in the photo above showing a 155mV reading on the multimeter display means that the temperature is 15.5±0.2°C.

Note that the LM35CA is only guaranteed to be within ±0.5°C at 25°C, but in reality, a typical sample of the device is within ±0.2°C from around –25°C to 50°C.

The reason for using three different devices is threefold. First, it increases your confidence that you have an accurate reading when they are all giving similar results. Second, it also lets you get an idea of which sensors read a little

Parts list – Thermometer Calibrator

1 diecast aluminium box, approx. 115 x 90 x 55m *[eg, Jaycar Cat HB5042]*
3 LM35CAZ temperature sensors *[eg Mouser LM35CAZ/NOPB, Digi-key LM35CAZ/NOPB-ND, RS Cat 5335878]*
3 voltmeters *[eg, Jaycar Cat QM1500]*
3 red banana plug to banana plug leads
3 black banana plug to banana plug leads
3 black chassis-mounting banana sockets
3 red chassis-mounting banana sockets
1 chassis-mounting 9V battery holder
1 9V battery clip with flying leads
1 9V battery (alkaline recommended)]
1 100nF ceramic, MKT or greencap capacitor
1 SPST toggle switch
1 small piece of protoboard
1 3mm ID solder lug
1 M3 x 10mm machine screw and nut
1 adhesive TO-3P or TO-247 insulating washer
1 small tube adhesive heatsink compound *[eg Jaycar NM2014]*
Various lengths of ribbon cable or hookup wire

Accurate temperature measurement is not easy

Making precise temperature readings (say to within ±0.1°C) is difficult. Devices to do this are not commonly available and are very expensive.

For example, if your backyard weather thermometer is showing 40°C, it could actually be 38°C or 42°C. It could even be much higher or lower than this if your thermometer is poorly sited (eg, near an air conditioner or road) or in a poorly designed enclosure or bad position, which allows its reading to be affected by direct sunlight.

Assuming your sensor is linear, you can calibrate it using a stirred ice bath (to determine its reading at 0°C) and vigorously boiling pure water (100°C), both at sea level. But unless you do this correctly, your readings could still be out considerably.

For example, at around 300m elevation, the boiling point of water is about 98.9°C. Normal day-to-day atmospheric pressure variations can have a small effect on the boiling point, too.

Any salt in the water or ice can have a dramatic effect on both the boiling and freezing points. According to the *CRC Handbook of Chemistry and Physics*, 2.92% sodium chloride in solution reduces the melting point of ice by 0.19°C and increases the boiling point by 0.05°C.

Calibrating Theremometers, a practical thermometer calibration method from NFSMI can be downloaded from the June 2021 page of the *PE* website.

Even if your calibration method is flawless, you also need to know that the sensor response is perfectly linear to have confidence in readings between the two extremes.

IC-based temperature sensors like the LM35 do suffer from some level of non-linearity, even though they are designed to be as linear as possible.

And it isn't just electronic sensors that suffer from accuracy problems, either. As one meteorologist pointed out, even the meniscus (bulge in the top of a column of liquid in a tube) in a mercury or alcohol thermometer can lead to significant inaccuracies in the readings. He also mentions:

'... mercury freezes at -38.8°C. It becomes increasingly less malleable as it approaches that temperature and makes low temperatures with mercury thermometers of no value. The 18th century observers of the Hudson's Bay Company using thermometers provided by the Royal Society were unaware of the problem ...'

Because of problems like this, interpreting historical air and sea temperature data is quite tricky!

higher or lower than the others. And third, it also lets you check that the case is at an even temperature before making your readings.

In the photo overleaf, with all three giving readings within 0.2°C of each other, note how the cheap digital thermometers with their probes inserted into the same metal case, and presumably reading the same temperature, are all reading high (by about 0.5-1°C) and also have a considerably greater spread than the LM35CA devices.

You **must** use the LM35CA version for accuracy, as the LM35/LM35A/LM35C/LM35D cannot achieve the same accuracy. (The 'Z' suffix indicates a TO-92 package).

Note though that the LM35CA is limited to measuring in the range of –40°C to +110°C, while the less accurate LM35 and LM35A versions can measure from –55°C to +150°C.

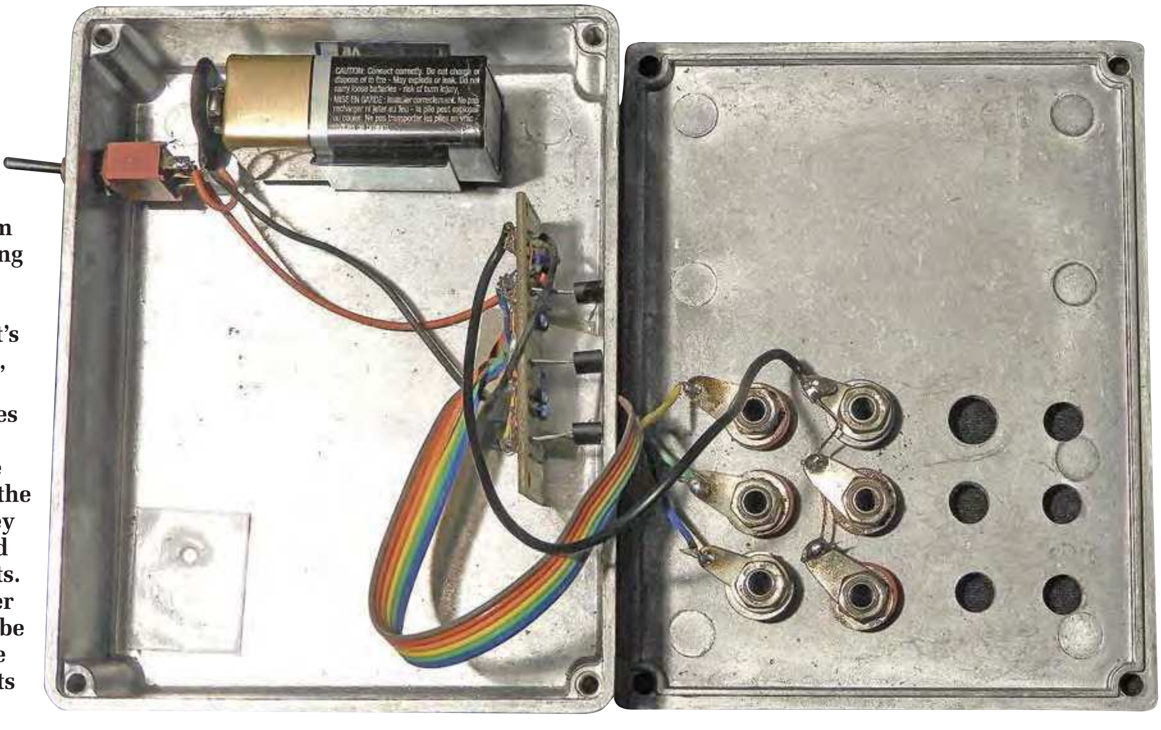
The three multimeters we've used here are low-cost devices that you can get for a few dollars from Jaycar, and we've found that they are very accurate. They have a voltage accuracy rating of ±0.5%, which equates to an additional error of just ±0.1°C in the temperature readings.

To demonstrate the accuracy of the LM35CAs, we also have two laboratory-grade analogue thermometers measuring the same temperature. As shown in the separate photo (frst page), they are both reading just under 16°C, just slightly higher than the figures shown on the DMMs.

Do not buy *cheap* LM35 sensors online if you are expecting accuracy, or even for them to function. We also purchased several LM35Ds cheaply on the internet to compare, but NONE of them worked at all! So it is essential to obtain them from a reputable supplier (eg, the ones mentioned in the parts list).

Construction

We recommend that you build this into a diecast aluminium box. This will not only provide some shielding, it allows you to check glass thermometers and to help maintain



While it might seem like overkill, placing this project in a diecast case has several benefits – it's shielded, of course, and the thick aluminium provides some thermal inertia. Placing the LM35CAZs inside the box also means they will be less affected by external variants. Of course, a smaller diecast case could be used, providing the various components will fit.

	Thermistor	RTD	Thermocouple	LM35CAZ
Range	-100 to +325°C	-200 to +650°C	+200 to +1750°C	-40 to +150°C
Accuracy	±0.05 to ±1.5°C	±0.1 to ±1°C	±0.5 to ±5°C	±0.2°C at 25°C
Stability @ 100°C	0.2°C/year	0.05°C/year	Variable	0.2°C/year
Linearity	Exponential	Fairly linear	Non-linear	Linear
Power	Small current	Small current	Self-powered	4-20V DC
Response	0.1-10s	1-50s	0.1-10s	2-15s
Interference	Rarely	Rarely	Susceptible	Susceptible
Cost	Low to moderate	High	High	Moderate

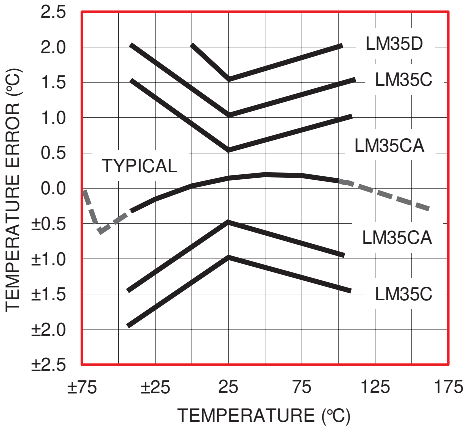


Table 1 shows the typical parameters of various temperature sensors, while the graphs at right show the errors in the different iterations of the LM35.

a uniform and stable temperature, without any thermal gradients. The sensors have very little self-heating, but it is still present; the large thermal mass of the case helps to mitigate this.

The LM35s also detect temperature variations through their pigtails. If these are exposed to small amounts of heat variations, such as human breath or wind, it can disturb the measurements and give false readings. By placing the ICs inside a metal box, we can eliminate these errors.

Solder the three LM35s to a small piece of protoboard, veroboard or similar. Join their V+ and GND leads together, and solder the 100nF capacitor across these rails. Also connect pairs of wires to the GND and OUT terminals of each device, plus one pair of wires between the V+ and GND rails.

Ideally, the pairs of wires should be figure-8 cable (eg, stripped from ribbon cable). If you are using individual wires, it's best to twist them together so that any interference is mostly cancelled out between the two conductors.

Now glue the three TO-92 plastic packages to the inside of the diecast box using thermally conductive adhesive. We used Jaycar NM2014 adhesive thermal paste.

Drill holes in the case for the power on/off switch and 9V battery holder, plus holes for the three pairs of banana sockets in the lid. Also drill a 3mm hole for the chassis grounding screw, near the battery holder, and one or two extra holes in the lid for analogue thermometer calibration, if desired.

Deburr all the holes and mount these parts. Then solder the pairs of wires from the LM35 GND and OUT terminals to the banana sockets, with the OUT terminals going to the red sockets.

The remaining pair of wires then goes to the switch (V+) and case (GND). Solder the other switch terminal to the red lead from the 9V battery, so that V+ is connected to the battery when the switch is in the on position (usually down).

Join the remaining GND wire to the black wire of the 9V battery to the solder lug and attach it to the inside of the case using an M3 machine screw and nut (not shown below).

Stick the insulating washer on the inside of the case directly below the analogue thermometer insertion holes in the lid. This will provide the thermometers with a bit of a 'cushion' so that they do not break when inserted.

Now connect the battery clip to the battery, slot it into its holder and switch on the power. Use a red and black pair of banana plug leads to connect one of the DMMs to one of the pairs of binding posts, and check that you get a reading that's fairly close to ambient temperature.

For example, if it's around 25°C where you are, you should get a reading around 250mV. Verify that all the outputs are similar values.

Using it

Avoid using this device in a windy environment or one with rapidly changing temperatures, such as near a window that's exposed to full sun where clouds may pass by. Ideally, it should be used indoors with still air in an environment with a stable temperature.

Switch it on and allow everything to stabilise for around 20 minutes before using it for best results.

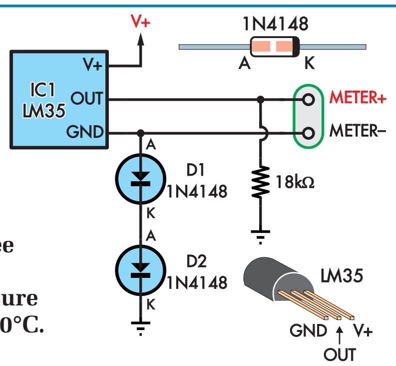
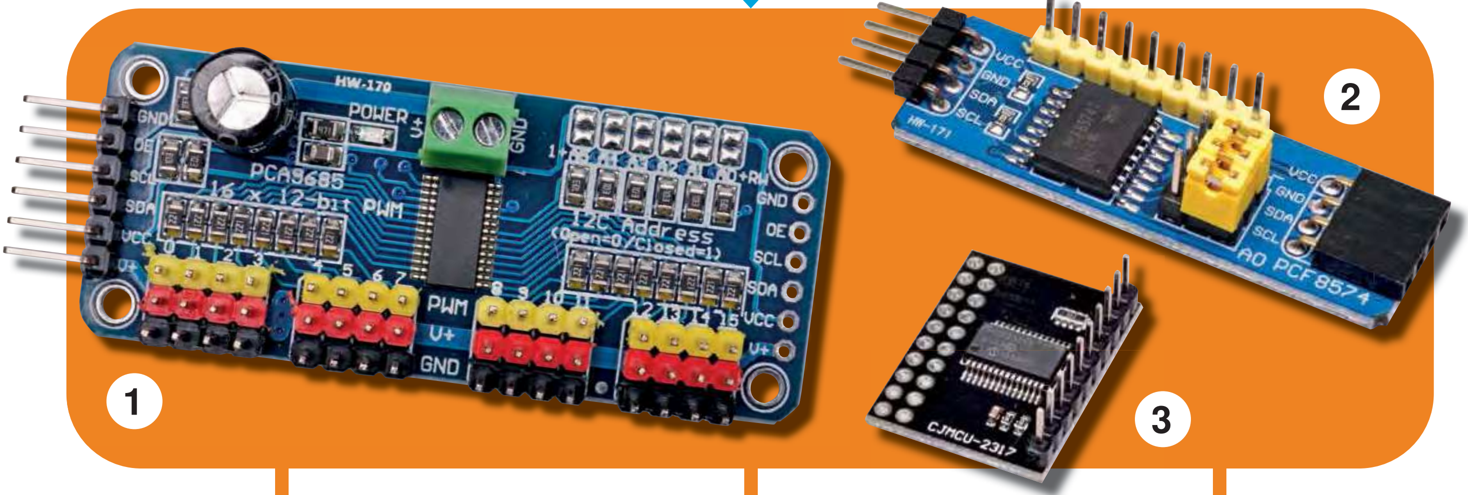


Fig.2: by adding three components to each LM35, you can measure temperatures below 0°C.

I/O Expander Modules



Sometimes, when working with microcontrollers, you just don't have enough pins to do what needs to be done. You might have started with the idea of a simple design but later found out that you had forgotten some crucial features. Uh oh! It can be a lot of work to change to a bigger, more expensive microcontroller, possibly involving learning some new soldering or programming skills. But there's another way out of this pickle.

by Tim Blythman

If you've been working with microcontrollers, you've almost certainly run into the situation where you don't have enough I/O pins to do what you need to. Or you've known in advance that you don't have enough pins, but for whatever reason, you don't want to switch to a bigger part. It can be a conundrum.

The ideal solution is to use an I/O expander module. In this article, we describe three different expander modules. They are all controlled over an I^2C serial bus, so at worst, they take up two pins on your micro. If you're already using the I^2C bus for other purposes, they won't use up any more pins at all.

That's the great thing about I^2C; the addressing scheme means that over 100 devices can be controlled by just two lines. Many microcontroller platforms (including Arduino and Micromite BASIC) include native support for I^2C.

And all three of the modules we present have the option to change the device address, so multiple expanders can be connected using the same bus. I/O pin counts in the hundreds are easily achievable by using enough of these modules.

The three modules we describe here have a variety of different features, so they have different strengths. We'll describe them according to the IC that they are based around; in each case, the IC data sheet is a great resource to help you fully understand each module's capabilities and quirks.

While it's possible to use the bare ICs in your designs, they are all quite small, so by using a module, you not only save the effort of having to solder them, you also get all the other necessary support components along with handy headers to make connecting to other devices a cinch.

Expander 1: PCA9685 module

This module provides up to 16 pulse-width modulated (PWM) or standard digital outputs, which can be used for various purposes including controlling LED brightness or stepper motors.

The PCA9685 module measures 63 × 25mm and features two six-way headers for control, plus 12 three-way I/O headers arranged in groups of four. There's also a two-way screw terminal for power and six pairs of pads which can be bridged to change the IC's I^2C address.

The original version of this board was designed by the Adafruit company, but has been cloned and is also available from several different online stores. The circuit diagram of the original Adafruit version is shown in Fig.1.

We sourced a few variants of this board, and found that there were a few variations, including one that omitted the reverse-polarity protection and one that used different resistor values. Another lacked the bypass capacitor. But they all did pretty much the same job.

The PCA9685 IC

This useful IC is manufactured by NXP; its data sheet can be found at: **https://bit.ly/pe-nov21-nxp**

It runs from 2.3-5.5V, so can work with both 5V Arduinos and 3.3V

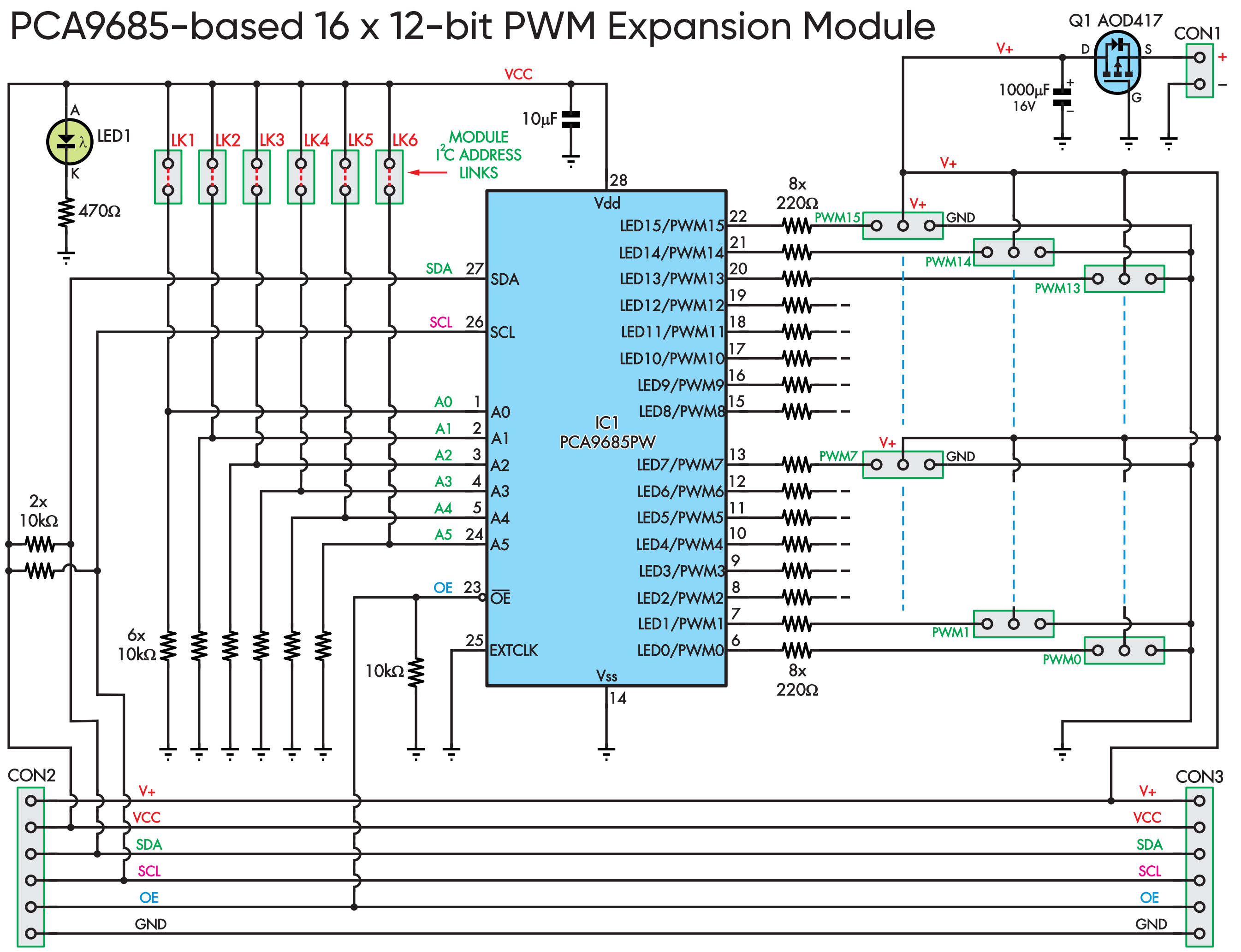


Fig.1: the PCA9685 module circuit (designed by Adafruit). Some variants/clones use different resistor values (eg, 120Ω instead of 470Ω, so the power LED is very bright) or omit the reverse polarity protection MOSFET or the electrolytic bypass capacitor.

Micromites, as well as the increasing number of 3.3V Arduinos.

It comes in a 28-pin SSOP or QFN package (both SMD). While it's possible to hand-solder chips this small, we find it easier to use the module if we have enough space to mount it.

While originally intended to be an LED PWM driver, Adafruit sells their PCA9685 board as a servo motor driver. Its 16 PWM channels can operate at up to 1500Hz with 12 bits of resolution (4096 steps), which is more than enough to generate servo control pulses.

The three-wide rows of pin headers allow many standard servo motors to plug directly into the board. At 50Hz (20ms between the pulses, as in a typical servo signal), pulses can be generated with a resolution of around 5µs, giving just over 200 steps between the standard servo pulse-width limits of 1ms and 2ms.

With these normally corresponding to positions of 0° and 180°, this gives a mechanical resolution of slightly better than 1°.

One interesting feature which may come in useful is that the PWM outputs can be started at different times, giving them different phases throughout the PWM cycle, although all outputs must run at the same frequency.

So for example, if you are driving multiple LEDs at less than full duty, they can be timed to stagger their switch-on times, such that (for example) only one is switched on at a time. This will limit the current steps drawn from the supply and probably reduce EMI too.

With the addition of a high-current buffer (eg, a Darlington array), this board could even be used to drive a stepper motor or brushless DC motor. By staggering the phases and changing the frequency, the output of the PCA9685 can be set to produce a pulse train sufficient to allow the motor to keep turning without further intervention.

We tested out some possible approaches to generate motor drive signals with this module, and some examples of the waveforms we came up with are shown in oscilloscope grab Scope 1.

Module description

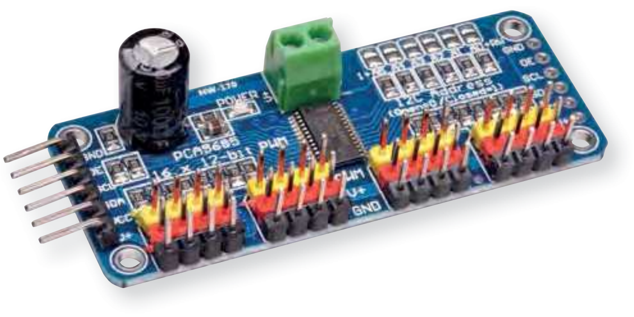
Apart from the 16 sets of output pins (each output is paired with a dedicated GND and power pin), there are also headers for power and I^{2}C bus connections as well as six solder jumpers to allow the address to be set.

An output enable (OE) pin is also broken out on the board, allowing all outputs to be enabled or disabled with a single signal, but an external clock connection is not provided. The module relies on the chip's internal 25MHz oscillator instead. The external clock pin is grounded as per the data sheet's recommendation for when it is not used.

Referring to the circuit diagram in Fig.1, we see that there are two different supply rails on the board. A nominal 5V rail powers the chip and can be found on the six-way headers at the pin marked V_{CC}. In a 3.3V system, this would be connected to the 3.3V rail.

A second rail marked V_+ is also available at the six-way header, as well as the two-way screw terminal. A diode-wired MOSFET provides reverse polarity protection if power is fed into V_+ from the screw terminal but not from the header. A 1000µF capacitor bypasses the V_+ rail.

There is no connection between V_+ and V_{CC}. The intention is that servo motors (if connected) run from the V_+ rail, while the logic runs from V_{CC}, minimising interaction between the logic and



The PCA9685-based module is one of the better designed I/O expander modules. Practically all the available pins are broken out, with the control pins replicated at each end, to allow multiple modules to be daisy-chained.

power parts of the circuit. All they have in common is a ground connection.

A separate bypass capacitor for the IC and the power indicator LED is also fed from V_{CC}. Apart from the external clock pin, all the IC's pins are broken out.

The six address pins (A0-A5) are normally pulled to ground by 10kΩ resistors, but they can be individually pulled high if the associated solder jumper is bridged.

While this might appear to give up to 64 available addresses, due to I^{2}C reserved addresses and auxiliary addresses for the PCA9685, the actual usable number is 55, using the (7-bit) range 64-119, excepting 112.

By default, with no jumpers set, the board has a 7-bit address of 64 or hexadecimal 0x40. The six jumpers effectively set the value of the six low-order address bits.

Address 112 is designated as 'All Call' and can be used to address any PCA9685 device regardless of its set address. This allows initialisation of a large number of ICs to occur quickly, by setting all attached devices to the same initial conditions.

During initialisation (or at any other time), the outputs can be set to open-drain (either pull low or high-impedance), push-pull or to inverted push-pull configurations.

The 16 PWM outputs are brought out to the top (yellow) row of pins on the board, where they are combined with a row of V_+ (red) and GND (black) headers to form a row of servo motor compatible connection points.

The OE (output enable) pin is brought out to the six-way headers but is also pulled to GND by a 10kΩ resistor, so the outputs are enabled by default. This line can be pulled up by a host micro to shut down the outputs if necessary.

The I^{2}C SDA and SCL pins are also brought out to the six-way headers and these have 10kΩ pull-up resistors. While this is higher than the recommended 4.7kΩ value for I^{2}C bus lines, we had no trouble without adding external pull-ups. Later, we will look at how these resistors behave when multiple boards are connected.

Cleverly, the two six-way headers have matching pin-outs, so boards can be stacked end to end, for example, by fitting a female header to one end and a male jumper to the other. The V_+ track is quite thick, and the GND trace consists of a solid copper pour on the back of the PCB, so passing a fair amount of current between boards is possible.

It appears the board is quite well designed and breaks out practically all the useful features of the PCA9685 IC.

What needs to be connected?

For basic testing, only four wires are needed: V_{CC}, GND, SDA and SCL. If you wish to connect a servo motor to the headers, you will need a supply for the V_+ rail too. The basic connections for a Micromite and Arduino are shown in Fig.2.

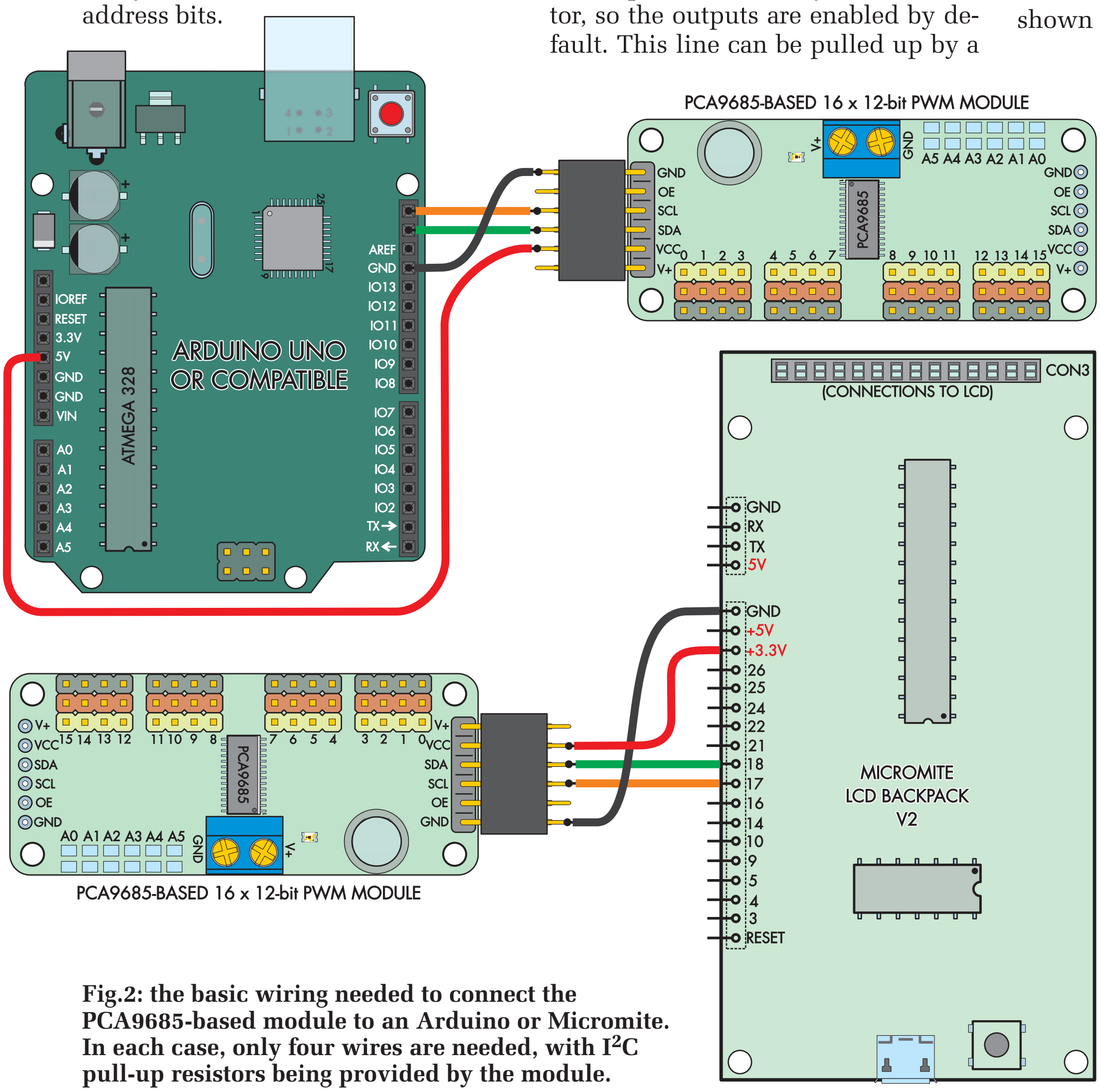


Fig.2: the basic wiring needed to connect the PCA9685-based module to an Arduino or Micromite. In each case, only four wires are needed, with I^{2}C pull-up resistors being provided by the module.

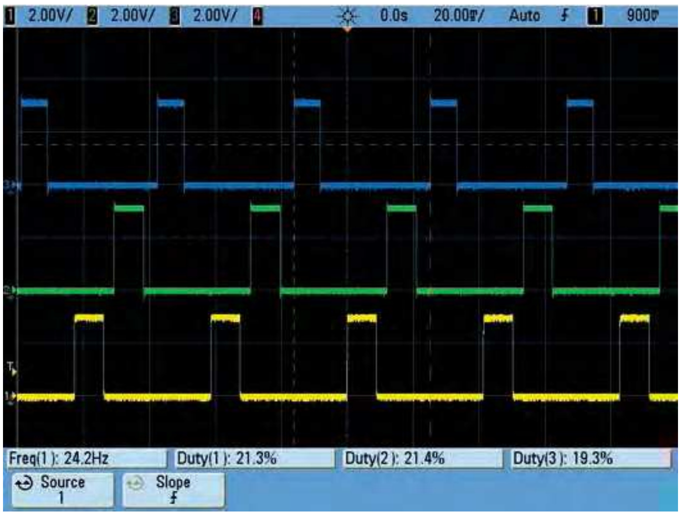
Software

We have written sample programs for Arduino and Micromite. Both of these allow the PWM frequency to be set, as well as the start and duration times of the pulses.

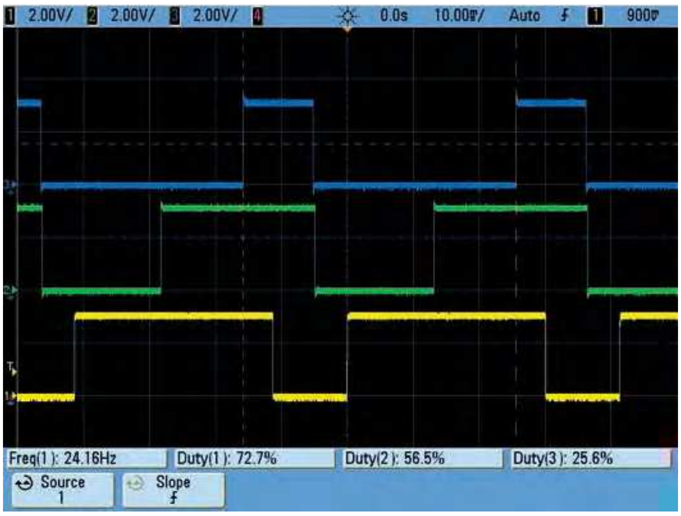
Internally, the PCA9685 uses start and end variables to define the pulse parameters of each output, as well as specific bits to enable full-on and full-off states, so some minor translation is done by the code.

In the Micromite example, these variables are set by sliders on an attached ILI9341 LCD (as you would have on a Micromite LCD BackPack), while the Arduino code uses the serial monitor as a menu to enter the parameters, these being a letter for the parameter followed by its value. Both examples contain some functions to simplify writing your own code to control the module. Adafruit has also written an Arduino library which can be found at **https://bit.ly/pe-nov21-ada**

Our sample program code is available for download in



Scope 1: here we're using the PCA9685 module to generate pulse trains each phase shifted by approximately 120° compared to the last. Waveform like this could be used to drive a brushless motor or spread out the current demand of multiple PWM loads.



Scope 2: this demonstrates using the PCA9685 module to produce three different PWM waveforms with different rise and fall positions, with each duty cycle being fully adjustable. The main restriction is that the repetition frequency of all outputs must be the same.

this month's free bundle from the November 2021 page of the *PE* website.

Expander 2: PCF8574 module

You may have heard of the PCF8574 before, especially if you have ever used any of the I^{2}C-controlled character LCD panels, as described in our June 2018 article. It is the PCF8574 that provides the I^{2}C-to-parallel conversion that makes it so easy to use these LCD screens.

The module we are looking at, designated 'HW-171', measures 48 × 11mm, although other similar modules are also available. Its circuit diagram is shown in Fig.3. It has a wide operating voltage range, 2.5-6V, making it suitable for all 3.3V and 5V applications.

The I^{2}C modules designed to attach to the back of an LCD panel can also be used as I/O expanders, although they usually omit one of the pins as only seven control lines are needed for driving a character LCD.

This module has a simple interface, with a four-pin male header at one end and a four-pin female header at the other end for control and daisy chaining. The pins are designated V_{CC}, GND, SDA and SCL, with the last two being the I^{2}C bus. A nine-way header breaks out the I/O ports on one side (the ninth pin provides an interrupt function), while a row of three three-pin headers with jumper shunts are used for address selection.

The male/female pin header combination allows multiple modules to be easily connected to the same I^{2}C bus, and the addressing scheme allows up to eight unique addresses.

Apart from the main IC, the only other electronic components on the module are a pair of 1kΩ pull-up resistors on the I^{2}C lines. These are much lower values than are typically used as I^{2}C pull-ups, but it still seems to work fine. We'll have a look at the effects of these resistors a bit later.

The PCF8574 IC

Just Like the PCA9685, the PCF8574 is made by NXP. Its datasheet is well worth a look, and you can examine it here: **https://bit.ly/pe-nov21-nxp2**

While it can only have eight different addresses, there is a variant called the PCF8574A, which is identical but has a different set of addresses, giving 16 total possibilities.

The PCF8574 can have a 7-bit address from the range 32 to 39, while the PCF8574A can have an address from 56 to 63. Our units had a default address of 32. Since the chips are interchangeable, if you can't get your module to work, check which of these two chips it has.

The PCF8574 modules are designed to be stacked end-on-end, meaning that it's trivial to connect multiple such modules to a single microcontroller. Note that the address jumpers are set here to give each module a different I^{2}C bus address, to avoid conflicts.

While NXP does not make a DIP version of this IC, Texas Instruments does, so it is possible to replicate the functions of this module on a breadboard with the addition of two pull-up resistors for the I^{2}C bus.

The datasheet mentions the PCF8574's suitability for driving LEDs, but unlike the PCA9685, this device is quite minimalist and so can only switch them on or off. But it does provide the ability to read the state of each pin, allowing them to be used as digital inputs, which the more complex PCA9685 does not.

Each of the eight I/O pins can be set to one of two states. The default power-up state is for the pins to be pulled up by a 100µA current source. In this state, the pin can be used as an input, detecting when a connected device pulls the pin low. The 100µA current source is also sufficient to drive a logic pin high, such as when the PCF8574 is used to drive alphanumeric LCD screens.

The other state is to pull the pin low. Each pin can sink up to 10mA. A brief 1mA pull-up current is applied on a transition from low to high, supplementing the weak 100µA pull-up and speeding up transitions.

While this scheme appears very basic, it allows all the pins to be written and/or read with a single byte command. Since repeated reads or writes can occur during the same I^{2}C transaction, complex wave trains can be generated as easily as port writes on a microcontroller.

This is perfect for controlling devices such as the character LCDs we mentioned earlier, as a stream of digital data is often needed to update a series of characters on the display.

The interrupt pin is an open-drain active-low output, and goes low on

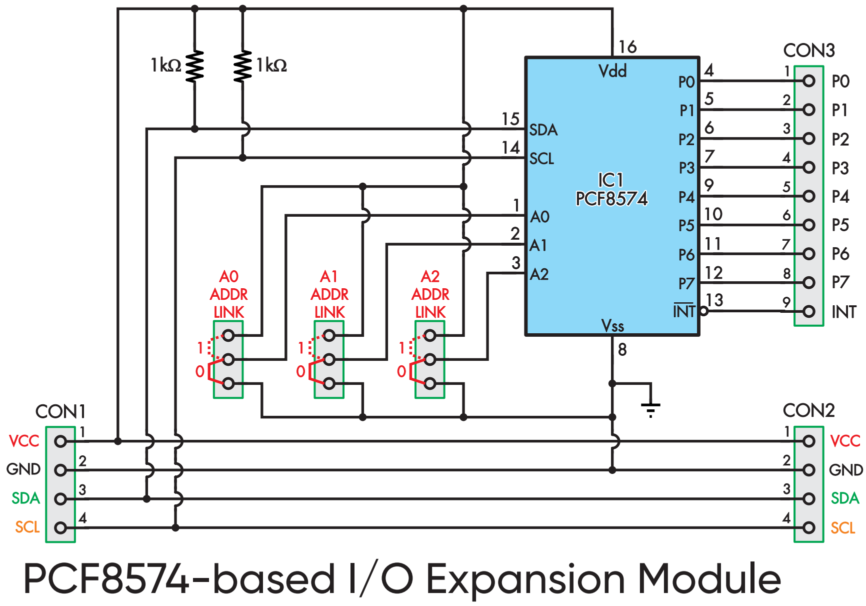


Fig.3: the circuit of the PCF8574-based module. Apart from the main IC, there are just two extra resistors. It's a great module in that all the useful pins are broken out in a well laid out arrangement.

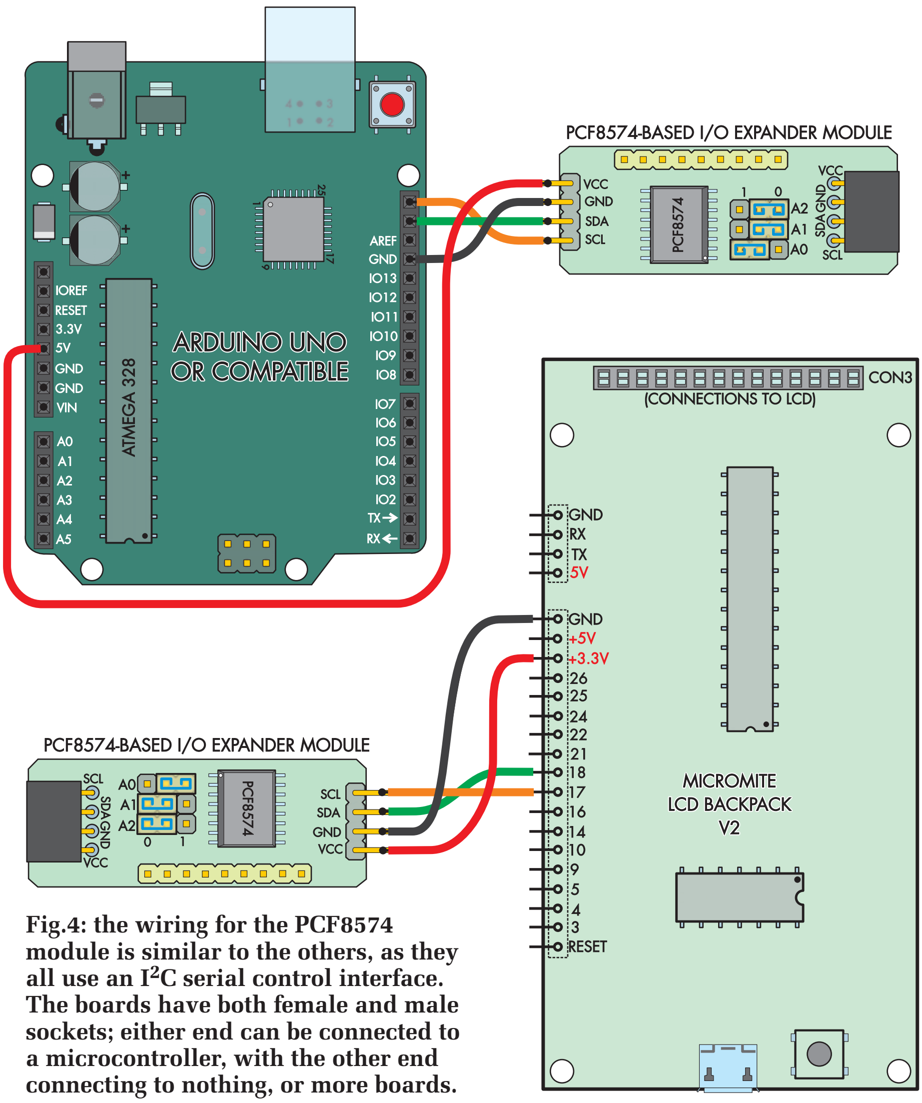


Fig.4: the wiring for the PCF8574 module is similar to the others, as they all use an I²C serial control interface. The boards have both female and male sockets; either end can be connected to a microcontroller, with the other end connecting to nothing, or more boards.

any change of input pin level. It is reset when a read occurs. It is intended to signal to the microcontroller that the input state(s) have changed and require reading. The interrupt pins of multiple modules can be paralleled, as any device can assert a low without conflicting with other modules.

With such a simple control scheme, no initialisation or command codes are needed; the data that is written or read corresponds precisely to the pin states.

Module description

The module itself is quite simple, as noted above, with only two resistors in addition to the main IC.

While the stackable feature of the modules is handy, it's a pity that the interrupt function is not brought out to a fifth pin at each end, which would make it easy to feed this signal back to the controller.

V_{CC} and GND pins near the I/O pins would have been nice too; as it is, there is nowhere convenient to connect the controlled device to the power supply.

As for the other module, only four connections are needed: V_{CC}, GND, SDA and SCL. See Fig.4 for the recommended connections to either a Micromite or an Arduino.

Software

As for the PCA9685 module, we have created both an Arduino and Micromite example program. The Micromite program uses a touch panel interface, while the Arduino program uses a serial interface.

Entering any of the numbers 0-7 will toggle the state of that output pin. The pin states are also read and the current state displayed. A read can also be performed by pressing the 'READ' button or entering 'R' on the Arduino software.

To help with troubleshooting, we've found some small I²C scanner programs (for Arduino and Micromite) and included them in our software download for this article.

These scan all addresses on the I²C bus and determine which addresses are actually in use. That might help you figure out which address your module is set for, if you can't figure it out from the jumpers and IC code.

Expander 3: MCP23017/S17 module

The MCP23017 IC is produced by Microchip, the same company responsible for PIC microcontrollers. It has 16 bi-directional digital I/O ports and is controlled over an I²C bus. There is an SPI version, which is called the MCP23S17. The module suits either version of the IC, as some of the pins are marked with designators for both I²C and SPI signals.

The MCP23017 IC has a working range of 1.8-5.5V, so this module is suitable for use with both Micromites and Arduinos. It is quite compact, measuring just 25 × 20mm, although this means that it only has space to label the functions on the back of the module. It has 30 pins in total, although they do not come fitted with headers.

It supports full bi-directional I/O operation on all pins. The register set is reminiscent of a PIC microcontroller, with control bytes for direction, pull-ups, output latches, port reading and interrupt enable. There's also another byte which can be used to invert the polarity of the port.

Given this many registers, there's a greater level of control than for the PCF8574-based module, including full push-pull output drivers, although it lacks the PWM feature of the PCA9685.

Just like a PIC microcontroller, all the I/O pins start as inputs but can be set to be outputs. The commands are simple, and consist of the IC address (as for all I^2C transactions) followed by a command (register) byte and data byte. Port writes up to eight-bits wide are possible. Its data sheet can be found at: **https://bit.ly/pe-nov21-mc**

Module description

There are two rows of ten pins at one end of the module with the connections to the controlled I/O ports (16 pins) plus connections for interrupt signals and power. There is another single row of 10 pins with the connection to the host for control and power; other non-I/O pins such as the address pins are broken out here too.

But the small size of the module means that some of the nicer features found on the other boards are omitted. For example, although the MCP23017 has three address pins to allow addressing up to eight modules, these pins aren't broken out to jumpers. To use them, you have to solder a wire from one or more of the address pins to the ground pin.

Similarly, there isn't a header to allow multiple modules to be easily stacked. So it's most easily used when it's the only expander module connected to the micro.

The fact that the two rows of output pins are adjacent means that the module does not lend itself well to being used on a breadboard, unless you're happy using just one row of the output pins.

The circuit

The circuit diagram for this module is shown in Fig.5. Apart from the main IC, there are two 10kΩ resistors, one four-way 10kΩ resistor array and a 100nF ceramic capacitor, used to bypass the IC's supply.

The two individual resistors are the I^2C pull-ups, while the resistor array is connected to pull the RESET ($\overline{MR}$) pin high (so the chip will operate as soon as power is supplied) and the address pins low (setting the default address). Otherwise, all the IC's pins are connected directly to pads on the module, with power (V_{CC}) and ground being the only pins connected via both sets.

For basic operation, only four wires need to be connected; power, ground and the two I^2C lines. These connections are sufficient to work with our sample code, and are shown in Fig.6.

Software

Because the MCP23017 works similarly to microcontroller I/Os, we have written our code to emulate the most common microcontroller pin control functions. For Arduino, the functions are named:

```
MCP23017digitalWrite()
MCP23017digitalRead()
MCP23017pinMode()
```

These work the same as their native counterparts. Our sample code is nothing more than the classic 'blink' routine (which toggles an output between high and low at 1Hz), with code added to read back the set state.

The Micromite code is similar, although the syntax of the commands is slightly different from the inbuilt statements. The functions are named:

```
MCP23017SETPIN
MCP23017READPIN
MCP23017WRITEPIN
```

The pin modes are:

```
OUT
IN
IN_PULLUP
```

The Micromite code draws buttons on an attached ILI9341 LCD screen in landscape mode. Four rows of sixteen buttons correspond to the 16 I/O channels and four states; the states are: input, input with pull-up, output high and output low. A further row shows the last states read from the I/O pins. Pressing any of the buttons, including the 'read' button, will cause the read states to be updated.

I^2C pull-ups

All three of these modules communicate via I^2C, and all have onboard pull-up resistors. The total pull-up resistance decreases as more boards are added and the resistors are effectively paralleled. We investigated what range of resistances allowed for correct operation, to get an idea of how many boards could realistically be used without modification.

For the Micromite, 220Ω pull-up resistances for SDA and SCL result in 15mA being sunk from the 3.3V supply when the pins are driven low. This is the absolute maximum pin current of the PIC32, and even under these conditions, I^2C communications at 400kHz (the Micromite's upper speed limit) worked flawlessly.

So a maximum of four PCF8574-based modules or 45 PCA9685-based modules can be connected to a Micromite, based on current draw on the I^2C pins. This does not take into account extra capacitance which may be added to the bus lines when extra modules are added, so these numbers may not be achievable in practice.

Removing the resistors from some of the modules will decrease this load, as will adding a second I^2C bus.

Similarly, the ATmega328 processor on an Arduino Uno supports a maximum of 40mA on each pin, which corresponds to 125Ω pull-ups to the 5V supply. So we tested using 150Ω pull-up resistors.

This too proved to work fine for both modules, suggesting up to six PCF8574-based modules or 66 PCA9685-based modules can be connected to an Arduino board. This includes the same assumptions as earlier, and these results may not be achievable in practice.

It appears that the I^2C bus is quite robust, and can work well if it's operating slightly outside its recommended conditions.

Although we didn't run any tests on the MCP23017-based module, based on these results, it should work fine with up to eight modules (the maximum that would be addressable).

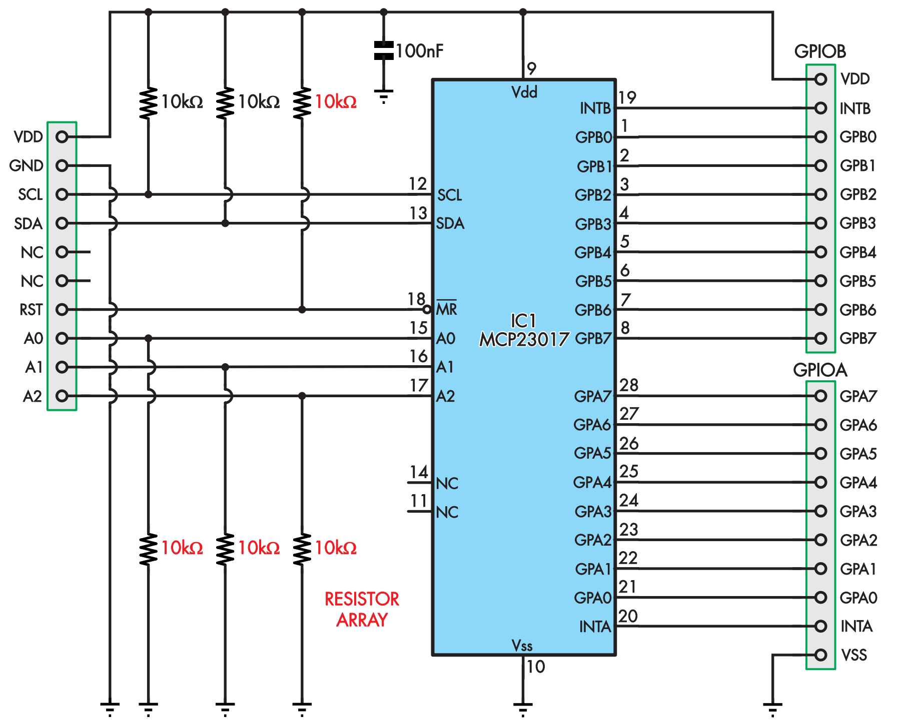
Level shifting

Another interesting possibility that arises in using I/O expander modules is that it allows for parts of the circuit to operate at different voltages.

I^2C is an open-collector bus, so devices either pull the SCL and SDA lines to ground or let them rise to a higher voltage due to the pull-up resistors. Thus, it isn't necessary for all devices on the bus to have an identical logic high voltage.

If the bus pull-ups are connected to the lowest voltage supply used, no

The MCP23017 module does not feature stackable headers or address jumpers, but it is very compact and provides full digital I/O control of 16 pins, similar to that of a microcontroller. Due to its small size, the pins are labelled on the back of the module.



CJMCU-2317 (MCP23017) I/O Expander Module

Fig.5: the MCP23017-based module is quite compact, although this does leave it at a minor disadvantage for usability compared to the other two modules. The I²C and power pins are on one side, with the I/O pins on the other side.

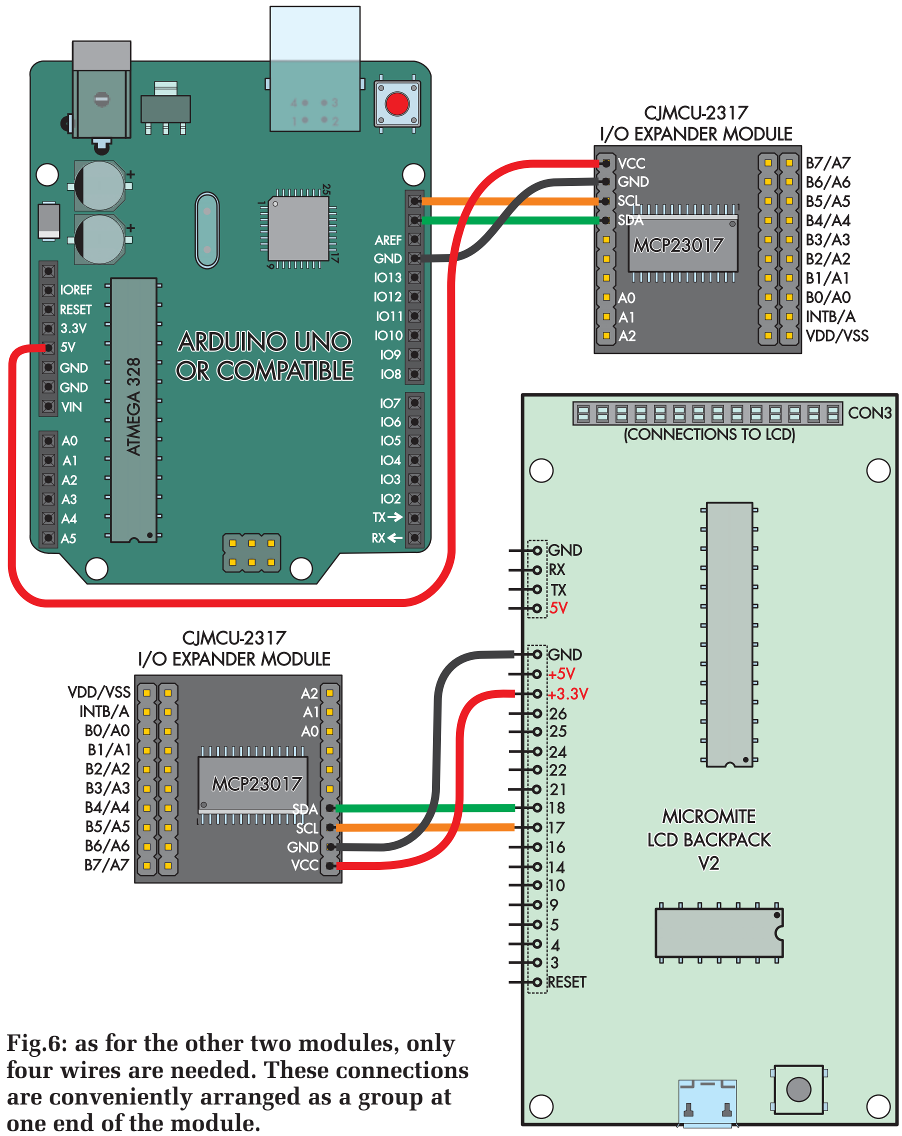


Fig.6: as for the other two modules, only four wires are needed. These connections are conveniently arranged as a group at one end of the module.

damage can occur through over-voltage. As long as this level is detected as high by the device with the highest logic voltage, then it will still work, although with reduced margin in clearly defined logic levels.

It's important in this sort of situation to ensure that the pull-up resistors that are connected to the bus go only to the lower voltage supplies (although most chips have internal clamp diodes which will clamp the high voltage to a safe level anyway).

So for example, you could connect an I/O expander module running off 5V to a 3.3V Micromite and it should work just fine.

You would then have 3.3V I/Os available direct from the Micromite, and 5V I/Os from the expander. Ideally, the I²C pull-ups should go to the 3.3V supply.

Similarly, you could connect a 3.3V I/O expander to a 5V Arduino micro. In this case, you would want to use the pull-ups on the expander module. The Arduino will read 3.3V as a high level, and while it will have its own 5V I/Os, you can also use the 3.3V I/Os of the expander module to communicate with other devices running at 3.3V.

One of these expander modules may even be the easiest and cheapest way to communicate with a chip that has a digital interface operating at a different level to your micro.

Note though that the resulting I/O speeds will not be very high; this is another factor to be considered.

Summary

Each module described here provides quite a different set of features, so which one is best for you will depend on your needs. You may even find it handy to connect multiple different expander modules to a single micro to perform different jobs.

For PWM or servo control, or LED brightness control, the PCA9685 module is the most useful. Its large number of possible addresses is also a strength. But it doesn't provide you with any extra digital inputs.

The PCF8574 module is the simplest and easiest to use.

If you need more full-fledged microcontroller type I/O pins, then the MCP23017 module has the advantage. There is extra overhead in controlling it compared to the PCF8574, but this is offset by extra features and more I/O pins.

As mentioned above, you can mix and match the modules, although it is an unlucky coincidence that the MCP23017 and the PCF8574 both share the same address space.

Colour Maximite 2 (Generation 2)

Part 2 | Words Phil Boyce | Design Geoff Graham | Firmware Peter Mather

Assembly and setup of our retro home computer

Last month, we introduced the new, and extremely powerful, *Colour Maximite 2 Generation 2* computer (*CMM2 G2*, or just '*G2*' for short). The specification and features list (see Part 1, last month) shows that this latest version of the *Maximite* certainly packs a lot of punch into a small package. Fig.1 shows various screenshots from different programs, and from these you can begin to get an idea of the *G2's* graphics capability; and do remember that all these programs are coded in BASIC.

In keeping with all previous versions of the *Maximite* family, the *G2* makes for a very rewarding self-assembly electronics project. Fig.2 shows that the *G2* PCB contains many surface-mount devices (SMDs); however, don't let this put you off building it yourself. I guarantee that even if you have never soldered an SMD component before, providing you can use a soldering iron with through-hole components then you will be more than able to build the kit version of the *G2*; and probably complete it in just a couple of hours. If you look at the circuit diagram in Fig.3, you may be wondering why I am so confident with this claim. Put simply, the *G2* kit contains a part-assembled PCB that just needs finishing off with a handful of *through-hole* parts.

PCB, kit, or fully assembled?

The *G2* PCB has been optimised for automated machine assembly, but what does this actually mean? Well, you could use the PCB Gerber files (supplied in this month's downloads from the November 2021 page of the *PE* website) and get a PCB made from one of the many Far-East PCB fabricators. You would then need to painstakingly solder all the SMDs yourself. I have done this on several occasions when the prototype was being developed; however, this method of assembly does require you to be extremely confident with SMD soldering, not to mention very patient – especially when soldering the

Fig.1. Various screenshots from some early programs showing the potential of the *G2's* impressive graphical output – all coded in MMBASIC.

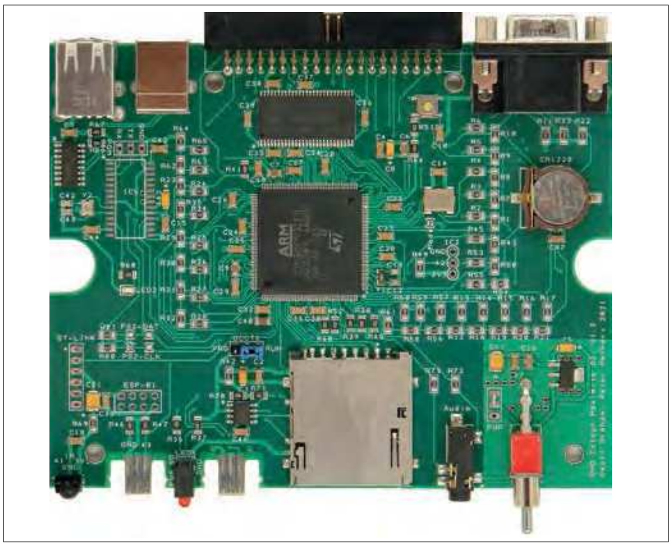


Fig.2. The assembled (4-layer) *G2* PCB mostly contains SMDs, all mounted on the top side. Around the edge, there are a handful of connectors, sockets and through-hole components.

176-pin STM processor. For most people, this assembly approach is not an option (and is not one that I would recommend). Instead, you can request that the PCB fabricator use their automation tools for soldering the SMDs for you. To do this, you need to supply two additional files alongside the Gerber files – the BoM and CPL (Bill of Materials and Component Layout).

The list of SMDs that the PCB fabricator can solder for you are shown in *Parts List 1*, and this should result in you receiving a board with most of the SMDs pre-soldered (see Fig.4). This saves a lot of soldering time, and also eliminates any silly mistakes and/or damage to SMDs. It is then just a matter of soldering the remaining components by hand, and then carefully fixing the assembled PCB (Fig.2) into the recommended enclosure. This approach is a much more practical assembly method and what we recommend if you wish to build your own *G2*. To save you having to deal with the PCB fabricator, and generating the two assembly files in the correct format (often specific to whichever fabricator you use), all the parts required to perform this process are included in the kit version of the *G2*. However, if this still seems like too much soldering, then you'll be glad to hear that the *G2* is also available as a ready-assembled unit.

NOTE: As discussed last month, sourcing the required STM32 processor is currently very dificult. This is frustrating, but outside of our control, so it may be a while before the STM32 becomes readily available again from the manufacturer.

Assembling the kit

See *Parts List 2* for what is included in the *G2* Kit. It includes the part-assembled PCB, along with all the connectors, sockets and through-hole components required to finish the PCB (in other words – the parts that the PCB fabricator won't assemble). Also included is the case, front and rear panels and all the mounting hardware to fix the PCB into the case. We will now briefly step through these items in the correct sequence to end up with a complete working *G2* kit.

Begin with the IR receiver, ensuring that the rounded face is closest to the edge of the PCB (this is the receiver side which detects any incoming IR signal, so it needs to be outward facing). Ensure you push it down all the way into the PCB (rather than 'floating' in the air). This will ensure it lines up with the 'IR window' hole in the front panel.

Next, mount the dual LED assembly, mounting it flush with the PCB. Next is the SD socket, and even though this is an SMD component, the spacing of the pins is large enough to make soldering it in place very straightforward. Do not worry if you don't have a temperature-controlled soldering iron (a bit of a misconception is that you must have one if you're soldering SMDs). I recommend a standard iron which has a power rating of 15-25W. The important point to bear in mind is not to leave the iron on any one of the SD socket's pins for too long as this could melt the plastic body that holds the pins in place. Liquid flux and a clean soldering iron tip is the key to a neat finish. The SD socket has a pair of plastic locating pins on the underside that holds it in correct alignment while soldering it in place. Begin by soldering the four larger corner pins (legs), then move onto the nine pins on the back edge, followed by the three pins on one side.

Next, solder in the stereo socket. This is also an SMD component with locating pins to assist with alignment. Once positioned, solder the five pins, but do be careful not to allow too much solder to form on any one pin. Use de-soldering wick if you do end up with excess solder on any joint. Next, solder the power switch into position – nothing to point out here apart from ensuring it remains upright (and at right-angles) to the PCB.

The coin-cell holder is another SMD component, but with only two large legs to solder. Ensure that the holder's orientation matches the shaped outline shown on the silkscreen (otherwise the battery will connect into the circuit with reverse polarity!). Do *not* install the battery at this stage – that comes at the end of assembly.

Next, mount the four connectors across the back-edge of the PCB. These are all through-hole parts so nothing to really mention other than to ensure that they all sit flush on the PCB. Once they've been soldered in place your PCB should look similar to Fig.2 (but with nothing in IC5's location near the top-left corner).

Finally, solder the two 3-way headers into place. The BOOT header (JP2) is the one that is close to the top-left corner of the SD socket; and the PWR header (JP1) is the other one between the stereo socket and the power switch. Slide the two jumper links into place – one into the 'PRG' position (ie, the left two pins of the BOOT header), and the other jumper link onto the top two pins of the PWR header (closest to C11). The latter makes the power switch active in the *down* position.

That completes the electrical assembly of the *G2*, so before we proceed it is now worth doing a thorough visual check of all the joints on the parts just soldered to ensure there are no shorts. Once you're happy with your soldering, place the PCB onto a cleared working surface so that there is no risk of any damage. I like to use the bottom half (the base) of the enclosure as this minimises the risk of accidentally placing the PCB onto any odd strand of metal which could damage it.

Initial testing

Before we load the MMBASIC firmware into the STM32 processor, it's worth testing how much current the unit draws from a 5V PSU. For this test, you will need a meter capable of measuring up to a minimum of 250mA and a 5V supply that supplies at least 300mA. I like to use Raspberry Pi PSU along with a decent thickness USB type-B lead. At around £8, these PSUs are relatively cheap, yet are more than capable of supplying enough current.

To measure the current, attach only the 5V PSU, and if you're not using a lab power supply, then you can put your meter leads across the 3-way PWR jumper (JP1). Place one probe on the bottom pin of the 3-way PWR header, and the other probe on either of the top two pins (or onto the jumper link if it has exposed metal). With the power switch in the up position (off), your meter will short out the 5V rail from the USB socket (CON2) direct to the 3.3V voltage regulator (REG1) allowing you to measure the current draw.

If everything is OK, you should see a reading of around 50mA. If you observe something significantly different to this, then first check that the PSU is indeed outputting 5V, and if so, thoroughly re-check all solder joints resolving any issues you find. Excessive current draw indicates that there is likely to be a solder short somewhere (or a misplaced component).

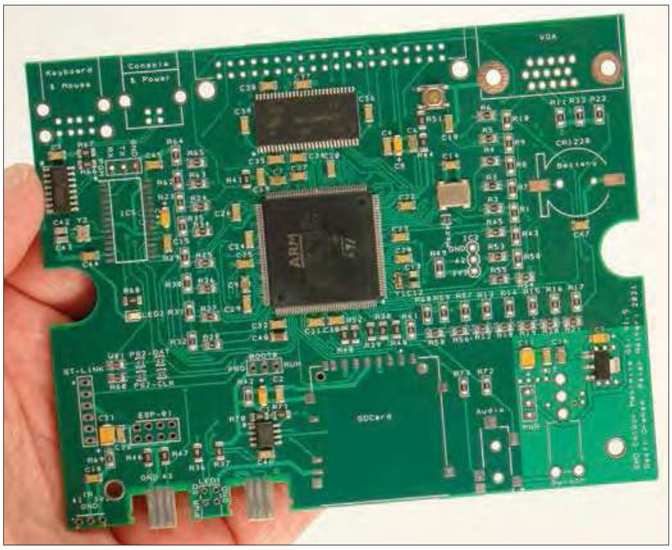


Fig.4. A PCB fabricator can pre-solder all the smaller SMD components, resulting in a board like this.

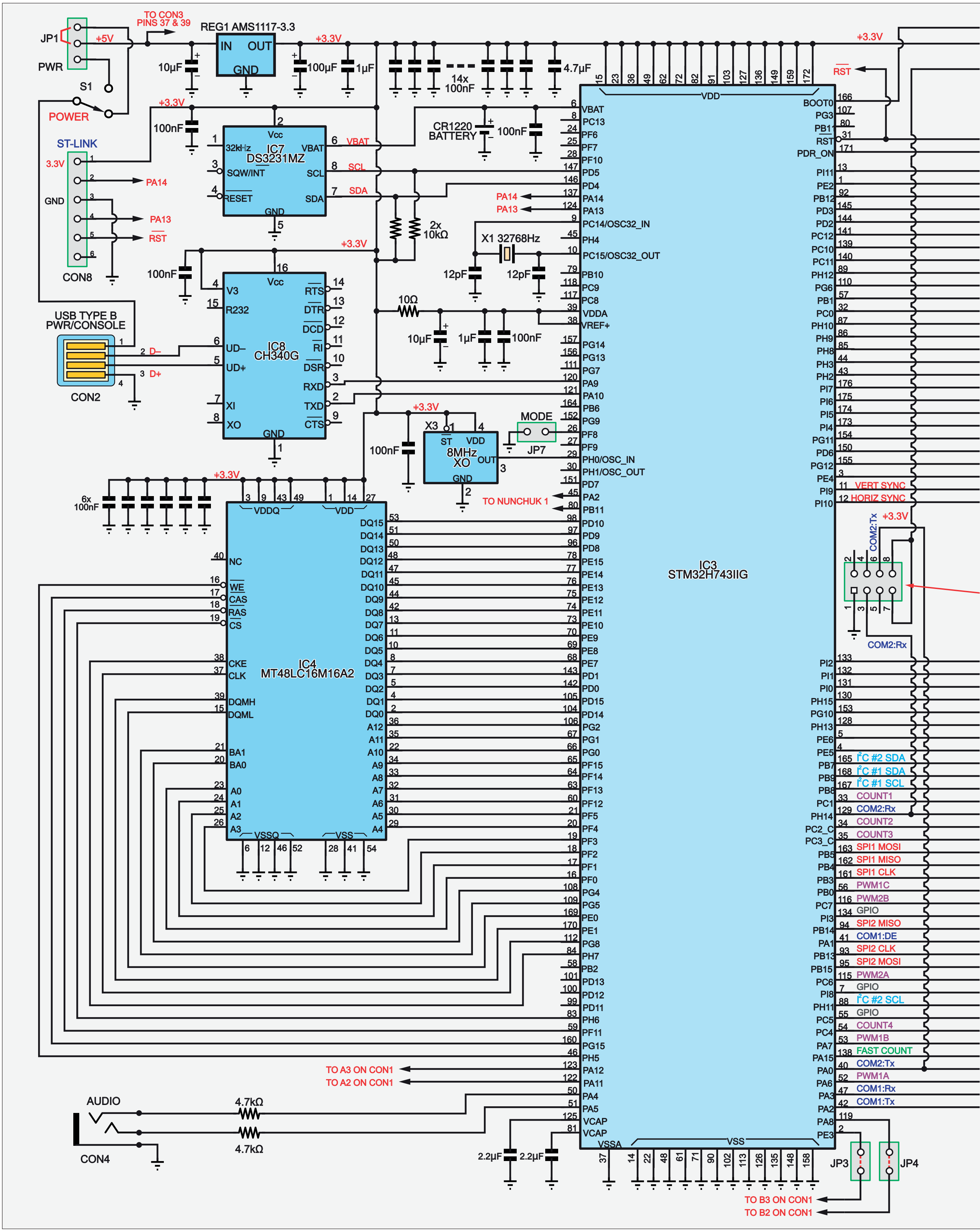
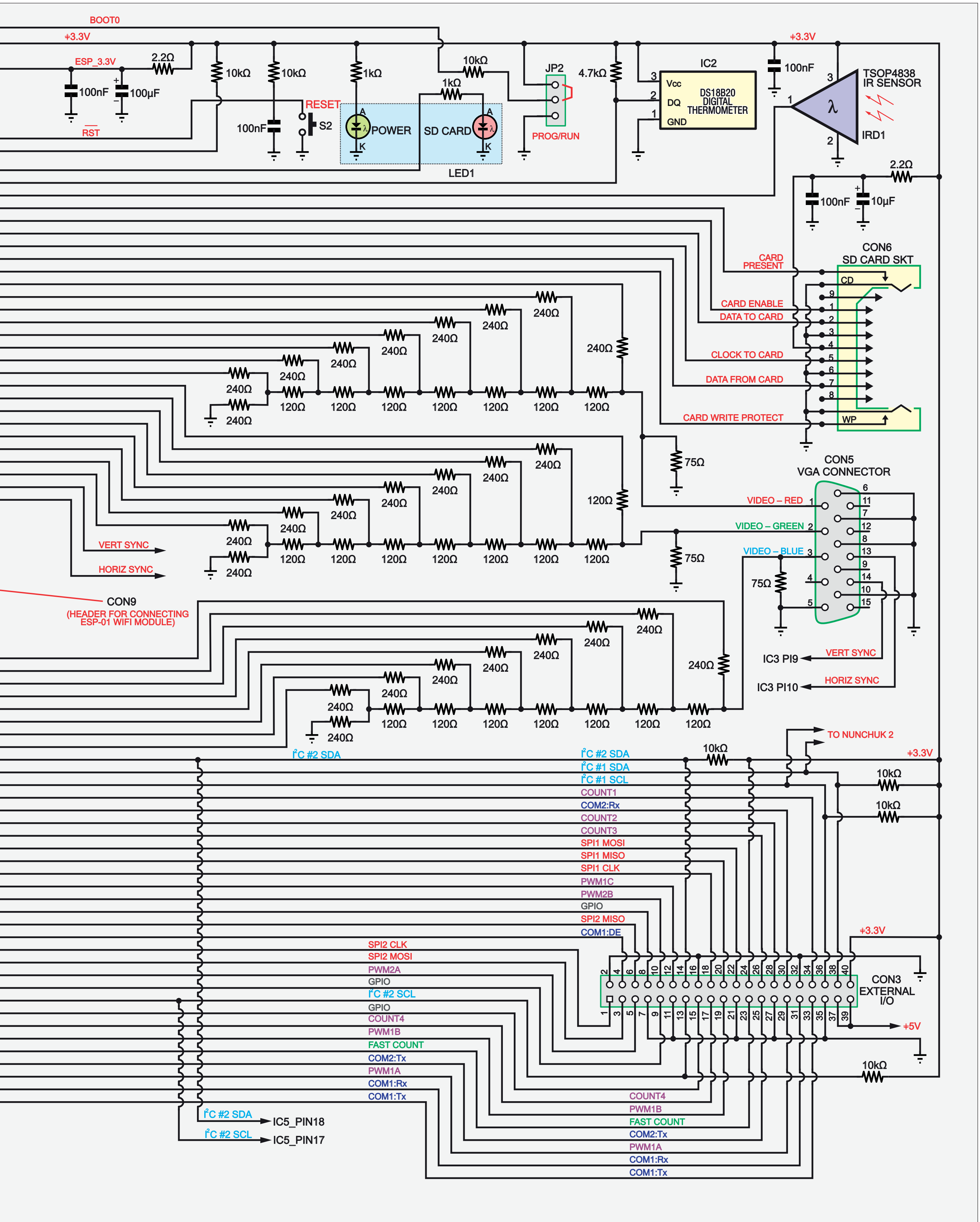


Fig.3. The circuit diagram highlights that the *G2* is built around the STM32 processor. There are just a handful of connectors, sockets and support components directly connected to the STM32. Three other support ICs (RAM, USB driver, and real-time clock), and one optional IC (for using a USB-protocol mouse) complete the circuit. (Diagram courtesy of *Silicon Chip* magazine)

USB driver

To install the MMBASIC firmware into the STM32 processor, you simply need to connect the *G2* to a computer (this can be either a Windows PC or Mac)



via a single USB (type-B) lead (this single lead will also supply power to the *G2* from your computer's USB port). However, you might need to install a USB driver first so that your computer's operating system (OS) can detect the *G2*. The *G2* uses the popular CH340G USB-to-serial IC (IC8) and the relevant driver can be downloaded and installed from here: **https://bit.ly/pe-nov21-drv**

Once installed, connect the *G2* to your computer and check it can detect the *G2* (for example, use Windows Device Manager and see which COM port has been assigned to the *G2*).

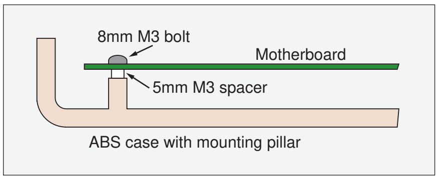


Fig.5. The PCB is fitted to the base using four 5mm nylon spacers and M3 screws.

Fig.6. Lower the PCB together with the front and rear panels into the base, ensuring the panels slide into the grooves. Four M3 screws fasten the PCB into place.

Installing MMBASIC firmware

To load the MMBASIC firmware, you will first need to download the STM32Cube-Programmer software, which is available from: **https://bit.ly/pe-nov21-stm32**

This software is free, but STM do require you to have an STM account, or at least to provide your name and email address. They will email you a link to download the software. Then install this software on your computer (Windows, Linux and macOS are supported).

You will also need the latest version of the MMBASIC firmware. Visit **https://geoffg.net/maximite.html** and scroll down to the bottom of the page to download the latest firmware file. Extract the file, and save it on your computer. (At the time of writing, the latest version was **CMM2v5.07.01.bin**)

With the *G2* connected to your computer (via the single USB type-B lead), ensure that the BOOT jumper link is positioned on the PRG position (ie, the left two pins are shorted). Then turn on the *G2* and check the green power LED lights up – this will indicate that the 3.3V line is operating correctly. Then press the Reset button (S2 – located near the VGA socket) which will put the STM32 into firmware programming mode.

Next, run the STM32CubeProgrammer software on your computer. On the top right of the program window select UART as the communications method. If the program does not automatically recognise the UART connection, click on the small blue circle to the right of the Port drop down list to refresh the entry.

Click on the 'Connect' button. You should then see a series of messages finishing with the message 'Data read successfully'. Any messages in red will indicate an error.

Click on the download button on the left side of the STM32CubeProgrammer window and the software will switch to the 'Erasing and Programming' mode. Use the 'Browse' button to select the firmware file (it will have an extension of .bin). Tick the 'Verify programming' checkbox. Finally, click on the 'Start Programming' button.

The STM32CubeProgrammer software will then program the firmware into the Flash memory of the STM32 (the STM32CubeProgrammer software calls this 'downloading'). After a short time, a dialog box will pop up saying 'File download completed'. Do not do anything at this point as the software will then start reading back the firmware programmed into Flash. When this has successfully completed, another dialog box will pop up saying 'Download verified successfully'. The whole operation will take around a minute – and remember, any messages in red will indicate an error.

Once the firmware has been successfully installed, disconnect the *G2* from your computer, and set the jumper-link on the BOOT header to the RUN position (right-two pins shorted). It is now time to carefully insert the CR1220 coin cell.

Initial testing

Now for the moment of truth – testing your *G2* powers up successfully. Connect your 5V PSU along with a VGA monitor, and a USB keyboard (to the top USB port). Use the front panel power switch to turn the *G2* on, and after less than a second, you should see the *Maximite* logo and the version number of the firmware that you have just loaded. If your monitor remains blank, ensure that you have not left the BOOT jumper link in the PRG position.

Note that depending upon your monitor, it may take longer than one second for an image to appear (the VGA signal will however be outputted from the *G2* within one second of powering up). The delay is purely the time it takes for your monitor to 'sync' to the VGA signal; I have one 'branded' monitor that takes eight seconds, and a TV that is instant.

The first time the *G2* is powered up, it will ask for the keyboard layout – press 1 for UK (other language options will be listed). Then it will request the date and time – simply enter the relevant numbers as guided.

If nothing appears to happen when you press a key, it may be because the USB keyboard you're using is not fully compatible – simply try another one. I prefer to use a Logitech K400+ wireless keyboard (costs around £30) since it is nice and compact, and it has never failed me during many hundreds of *CMM2* tests. There is another bonus to using the aforementioned keyboard (or any other wireless keyboard for that matter). When hooked up to a large TV the *G2* can be comfortably used from the sofa!

At this point, it is worth performing another power test (especially if you fail to see anything on the monitor). If you disconnect the monitor and keyboard, you should see a current draw of around 200mA. Any excessive deviation from this means you'll need to go back and check things over

Testing the SD socket

Before mounting the PCB into the case, there are a couple of quick tests to perform. With the monitor and USB keyboard connected, insert an SD card (ideally a branded one with a capacity between 8GB and 128GB). Power down, and after a couple of seconds, power back up. This time you should see the command prompt appear immediately (without requests for the keyboard layout or the date and time options). If it does ask for these items again then the chances are that the battery is either missing, incorrectly located, or flat. With a fresh battery correctly mounted, these options are retained whenever the *G2* is powered down (ie, the main USB 5V power is removed).

Now check in the very bottom-right corner of the display for the correct date and time. If not correct, then you can use the `DATE` and `TIME` commands to amend this (see the *CMM2 User Manual* for more details).

Next, eject the SD card (by gently pushing it in to release the internal latching mechanism), and in the bottom-centre of the screen you should see a 'Check Disk' message appear. Re-insert the SD Card and make sure the message disappears.

If 'Check Disk' remains on the screen no matter whether an SD card is inserted or not, then ensure that the solder joints on the SD socket's three side pins are not shorting to the main body of the socket.

If everything is working up to this point, we can move onto the final step.

Mounting the PCB

The case used for the *G2* is the same compact one that has been used for all *Colour Maximites*. It comprises two main body parts, the base (which has four mounting pillars built in), and the lid. The *G2* PCB is mounted in the base, and four 5mm spacers are added between the PCB and the pillars, as shown in Fig.5. These pillars raise the PCB and the connectors so that they will match the cut-outs in the front and rear panels. The pre-cut front and rear panels need to be positioned over the connectors before lowering the PCB down so that the panels slide into the grooves cut into the base (see Fig.6). This will then line up the four PCB holes with the four pillars and spacers. To save your sanity, it is worth using a small piece of Blu Tack under each spacer to avoid them falling off the pillars when lowering the PCB into place.

Use four M3 screws to fasten the PCB securely into place but do *not* over-tighten them. Finally, slide the lid down into position by ensuring the panels line up with the lid's grooves, and use the two longer case screws to hold the case together securely. This results in a professional-looking assembled unit.

Welcome Tape

Now that you have finished assembling your *G2*, you are probably keen to see what it can do. To get you started, we recommend the *Colour Maximite 2 Welcome Tape*. This was curated by Thomas Williams, and it comprises a collection of programs written by the user community. These include games, demonstrations and utilities. It has an easy-to-use menu, and you can always press `CTRL-C` to break out of a program and then `LIST` its code to see how it works. Download the *Welcome Tape* here: **https://bit.ly/pe-nov21-maxwel**

Keeping up to date

It's well worth regularly visiting Geoff Graham's website (**https://geoffg.net/maximite.html**) for links to review websites, and other sites of interest for both the *CMM2* and *CMM2 G2*. Also keep an eye on whether you have the latest version of the MMBASIC firmware from this same link. If you wish to upgrade the firmware, then simply download the relevant `.bin` file, connect your *G2* to your computer (via a single USB lead), and move the *G2's* BOOT jumper link to the PRG position. Press the small reset button and use the STM32CubeProgrammer software as outlined above to download the `.bin` file into the STM32. Once installed, move the BOOT jumper link back to RUN. Then, on power-up, check that the new firmware version number is displayed in the start-up screen.

Finally, there is a great community on The BackShed Forum that discuss all topics related to *Maximites* and Micromites. if you have any questions, then this is a good place to seek answers. The link to this friendly forum is: **https://bit.ly/pe-nov21-bshd**

Parts list 1 (SMDs on part-assembled PCB)

Resistors
All 0805 unless otherwise indicated

2x	2.2Ω	R42, R69
1x	10Ω	R44 (1206)
2x	75Ω	R11, R22, R33
21x	120Ω	R1, R7-10, R12, R18-21, R23, R29-32, R35, R43, R50, R56, R58, R62
27x	240Ω	R2-6, R13-17, R24-28, R34, R45, R53-55, R57, R59-61, R63-65
3x	1kΩ	R36, R37, R68 (1206)
5x	4.7kΩ	R49, R72, R73, R80, R81
13x	10kΩ	R38-41, R46-48, R51, R52, R66, R67, R70, R71 (1206)

Capacitors
All 0805 unless otherwise indicated

2x	6pF	C12, C13
2x	12pF	C42, C43
32x	100nF	C2-4, C10, C14, C15, C17-20, C22-39, C44-47 (1206)
2x	1µF	C6, C16 (1206)
2x	2.2µF	C7,C9 (1206)
1x	4.7µF	C40 (1206)
4x	10µF	C1, C5, C8, C41 (CASE-A_3216)
2x	100µF	C11, C21 (CASE-D_7343)

Semiconductors and crystals
All one-off

STM32H743IIT6	IC3 (LQFP-176) processor
CH340G	IC8 (SOIC-16) USB driver
MT48LC16M16A2	IC4 TSOP54 SDRAM
DS3231MZ	IC7 (SOIC-8) RTC
AMS1117	REG1 (SOT-223) 3.3V LDO
32.768kHz	X1 (SMD-3215_2P) crystal
8MHz	X3 (SMD-7050) crystal oscillator
16MHz	X2 (SMD-CRY-3225_4P)
LED	LED2 (0805)

Switch

Tactile Switch	S2 (SMD-SW-4_5.1x5.1x2.5) Reset Switch

Parts list 2 (kit)

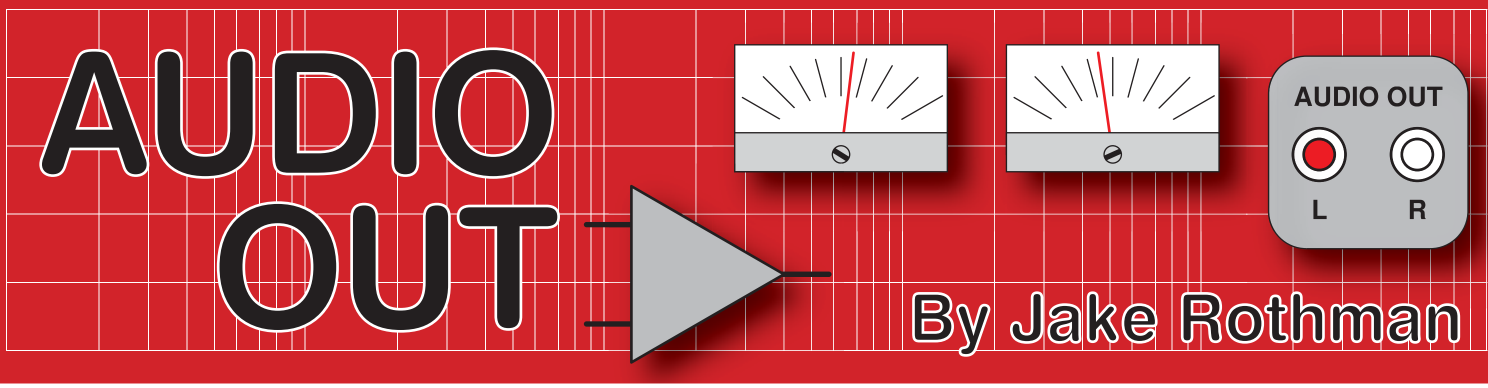
All one-off unless otherwise indicated

Part-assembled PCB
IR remote receiver, Vishay TSOP4838/40 (IRD1)
3mm dual LED assembly, Dialight 553-0112F (LED1)
SD card socket, Hirose DM1AA-SF-PEJ(72) (CON6)
3.5mm stereo socket, Switchcraft 35RASMT4BHNTRX (CON4)
Right-angle vertical PCB-mount SPDT toggle switch (eg, RS 734-7107) (S1)
Coin cell holder, HARWIN S8411-45R
15-pin VGA socket (CON5)
40-way, 2 row, right-angle PCB header, 2.54mm pitch. Hirose HIF3F-40PA-2.54DS(71) (CON3)
USB V2.0 type-B connector Amphenol FCI 61729-0010BLF, power (CON2)
Dual USB type-A PCB socket, Amphenol ICC 72309-8034BLF
CR1220 3V lithium battery
Multicomp Pro G738 case 140x110x35
Front panel
Rear panel
2x 3-way header 0.1-inch pitch (JP1, JP2)
2x Jumper link (0.1-inch slide on)
4x 5mm nylon spacers
4x 8mm M3 bolt

Next month, we will add the optional USB mouse controller chip if you want to use a USB-protocol mouse with the *G2*.

Until then, *have fun!*

Questions? Please email Phil at:
contactus@micromite.org



Analogue Vocoder – Part 1

Time for a new music project! Many of you are probably familiar with the vocal effect of an electronic instrument called the 'vocoder'. It's a fascinating piece of electronics, and over the next few months we will design and build a very high-quality example. The vocoder's primary claim to fame is where human speech is superimposed on a musical instrument. A popular example is the creepy 1961 recording *Sparky's Magic Piano* – see **https://youtu.be/Km19Iohd1YA** (strictly speaking, Sparky's Piano used a Sonovox which was not a vocoder, but a pair of special loudspeakers (compression drivers) applied to the vocalist's throat. It was also used in Dumbo for the talking train whistle.)

A more recent example is the definitive vocoder song, Laurie Anderson's 1981 release, *O Superman* – **https://youtu.be/S39NaDPNDtk** (see our editor's prized, pristine 12-inch vinyl copy in Fig.1!). It was recorded with the Roland VP-330: **www.vintagesynth.com/roland/vocoder.php**

Fig.1. Laurie Anderson's *O Superman*, a vocoded treat in avant guard minimalism using the Roland VP-330.

Fig.2. Software-based vocoders are possibly the cheapest entry point for most musicians, assuming you have a powerful computer and digital audio workstation software installed. Ironically, they can be a good way of optimising an analogue vocoder design before building.

Origins

As with most electronic audio/music technology, vocoders were originally developed for telecommunications. Invented by Homer Dudley at Bell Labs in 1936, they enabled more speech channels to be put through long low-bandwidth cables and other transmission systems.

Vocoders are an analogue data compression system which operate by representing speech data in its simplest form, amplitude changes of frequency bands, akin to a spectrum analyser. These slow changes can then be transmitted and the speech resynthesised at the other end. The military used them to disguise voices and encode secret messages. This is where the name comes from: VOice enCODER.

A vocoder can be thought of as a system for transferring the spectral energy of a speech signal onto another sound. An unfiltered organ or string synthesiser works especially well. If white noise is used, the vocoded voice will sound like a whisper.

An analogue vocoder is complex, the design I'm presenting consists of 28 precision filters and 14 voltage-controlled amplifiers (VCAs). For a 'normal'

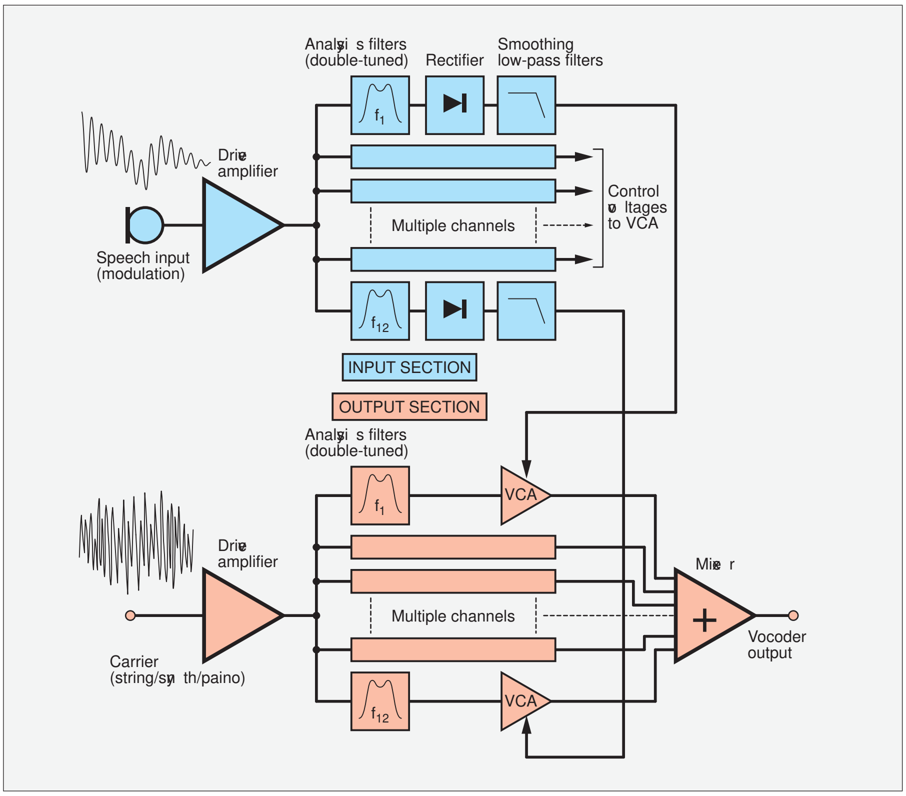


Fig.3. *Analogue Vocoder* block diagram, showing analysis, control and resynthesis sections.

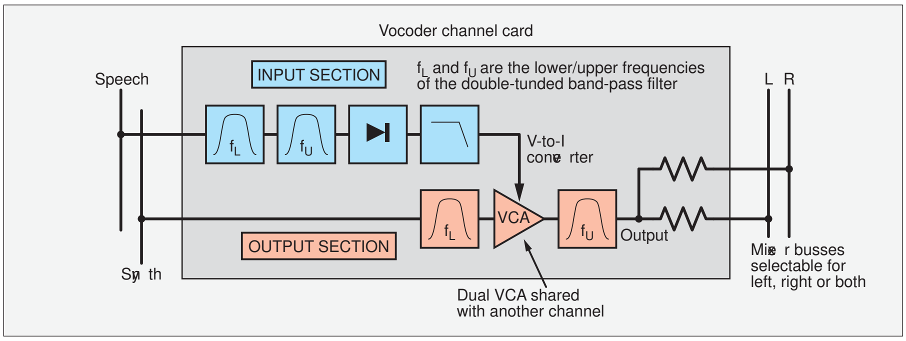


Fig.4. Each vocoder channel consists of these internal building blocks.

electronic engineer this is obviously something best achieved by digital signal processing using FFT (Fast Fourier Transform) and DSP hardware and software.

Vocoder software plug-ins are the entry point for most musicians wanting a vocoder sound. In my case, I played about with the digital Vocodex Channel Vocoder in FL Studio (Fig.2) to find the best frequencies and Q values to use. However, digital vocoders have their own problems. They need a lot of computing power and often exhibit considerable time delay or latency. They all use similar maths which imparts a peculiar 'under water' colouration at low signal levels, which I find unpleasant. Most smartphones use vocoding as part of their compression algorithms, so most people will be familiar with this nasty effect. Software vocoders are also complex to use with too many features and parameters to be set up. On the other hand, analogue vocoders have an immediate playability that lends them to composition and live performance.

DIY analogue vocoders are expensive to build, typically costing a few hundred pounds. However, buying a ready-made one, such as the Roland VP-330 Vocoder Plus, or Tim Orr's Electronic Music Studios (EMS), will set you back several thousand pounds: **https://bit.ly/pe-nov21-voc**

We are getting to the point where analogue electronic musical instruments are becoming expensive antiquities in their own right. This is one of the few occasions in electronics where building ones own can save serious money. Ironically, just as we go to press, and after spending five years building this *Vocoder*, Behringer have brought out a £400 Chinese surface-mount copy of the Roland, called the VC340. Still, my design is stereo *and* it's modifiable – important features, because having one's own unique sound signature is an essential attribute for electronic musicians.

System architecture

The input part of a vocoder is concerned with *analysing* the changes of a signal, usually speech in musical applications, and often called the 'modulator' in communications. This section generates slowly changing DC control voltages representing the amplitudes of the different frequency bands.

The output part is concerned with the generation or *synthesis* of the final sound. The block diagram is shown in Fig.3. In music applications, chords, from say a piano or string synthesiser, (or the carrier/excitation in comms) are split into corresponding frequency bands. These bands are then modulated by the control signals derived from the input section. This requires a VCA to control the output of each frequency band. The end result is that the piano or other musical input can be made to 'talk in tune'. Alternatively, the control signals can be patched from one frequency band to another to make crazy noises and facilitate speech scrambling.

The internals of the frequency-band modules are shown in the block diagram

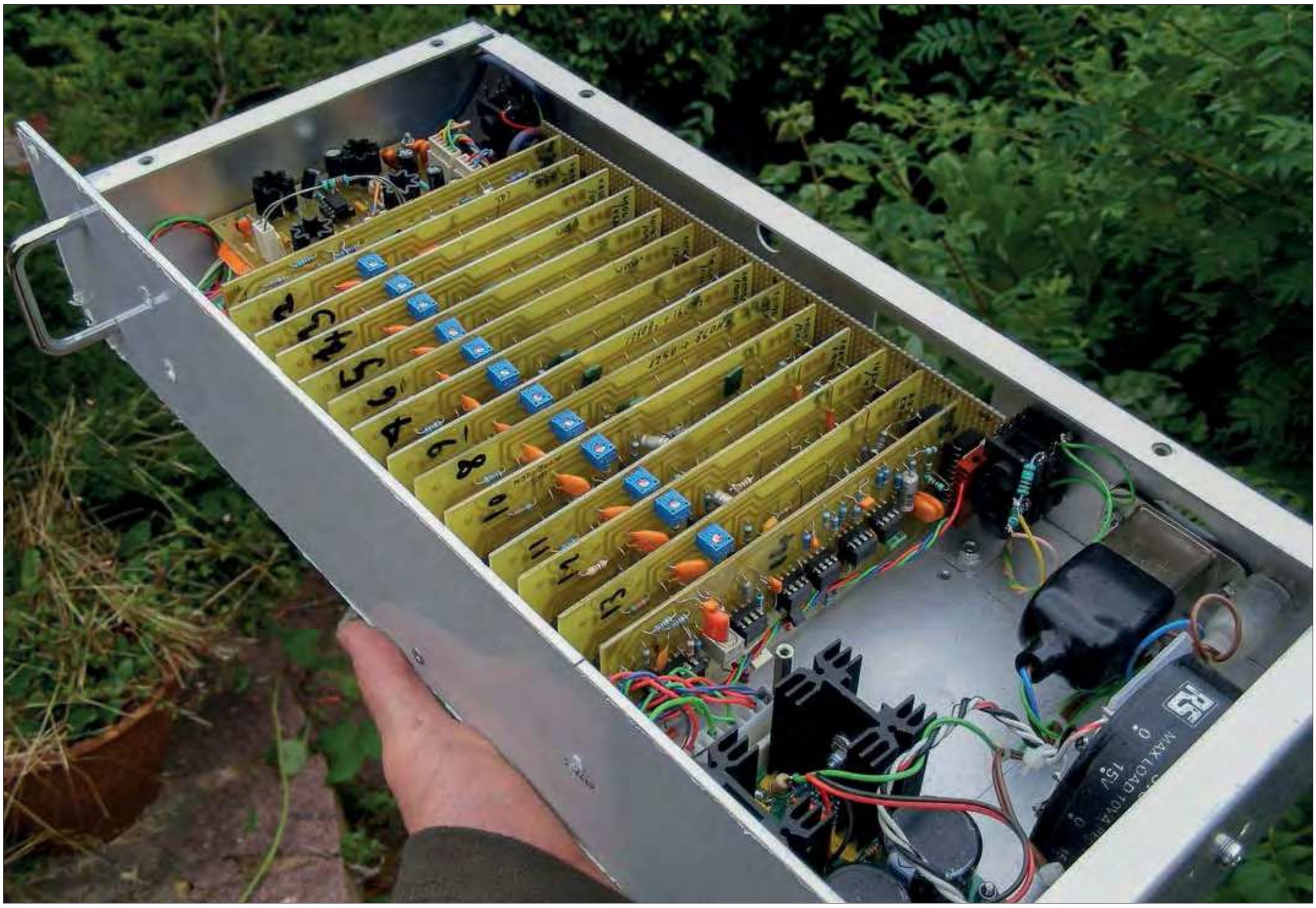


Fig.5. Early photo of the prototype analogue vocoder. No knobs yet.

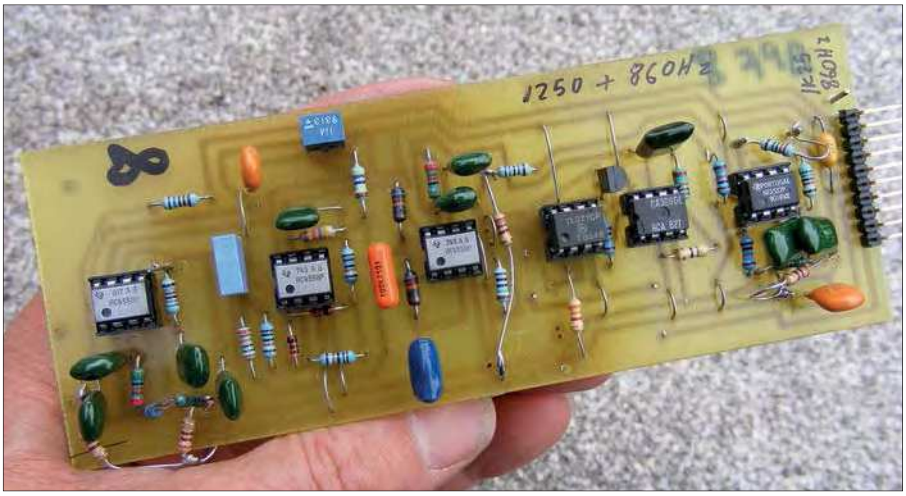


Fig.6. Prototype Vocoder channel PCB. Each channel frequency has to have different values of filter capacitors. (The final version will be double-sided to remove all those links.)

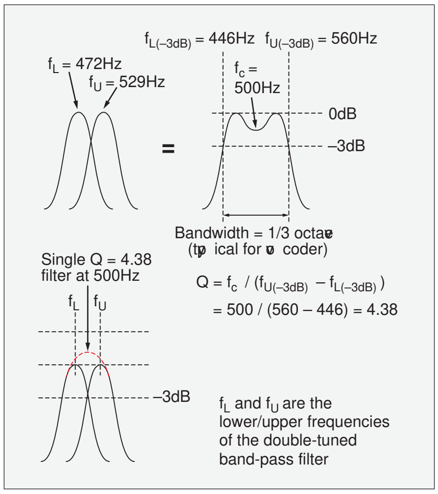


Fig.8. By putting two high Q band-pass filters in series a good compromise curve can be obtained with steep skirts and a flat top. This forms a double-tuned band-pass filter, familiar to most radio designers.

and are further expanded in Fig.4. Each of these are made into a PCB card that can be plugged into a bus-board to connect them all together. The prototype *Analogue Vocoder* is shown in Fig.5.

This design uses 12 band-pass filter modules with frequencies optimised for speech; plus, the system topped and tailed by fourth-order low-pass (f_c = 120Hz) and high-pass filters (f_c = 8kHz). This builds to a total of 14 filters, all fourth-order, and requiring 112 close-tolerance capacitors which represent a major chunk of the parts bill. The EMS vocoder has 22 band-pass filters! A prototype band-pass vocoder channel board is shown in Fig.6.

Band-pass filters

There are many ways to build a band-pass filter. The state-variable and its alternative the bi-quad are very effective. The bi-quad's resonant frequency (f_c) can be adjusted with just a single resistor; and the state-variable's f_c with two resistors – see Fig.7. However, both these configurations need three op amps per filter, which would be excessive in a multiple filter system like a vocoder. For perfect frequency band discrimination, *digital* brick-wall band-pass filters are used – but I suspect in musical applications these would sound horrible because of the group-delay-induced ringing.

Better bring in Q, Bond

The band-pass frequencies in the speech band need to be spaced at around one third of an octave steps. This necessitates a filter Q of around 4.38, to put the cut-off (–3dB) frequencies in the right place. To obtain a flatter overall frequency response, double-tuned band-pass filters can be used, which comprise two filters whose curves overlap, as shown in Fig.8. These have two resonances close together with a Q of 8.8, which gives a 'flat' top with initial steep (–40dB per octave)

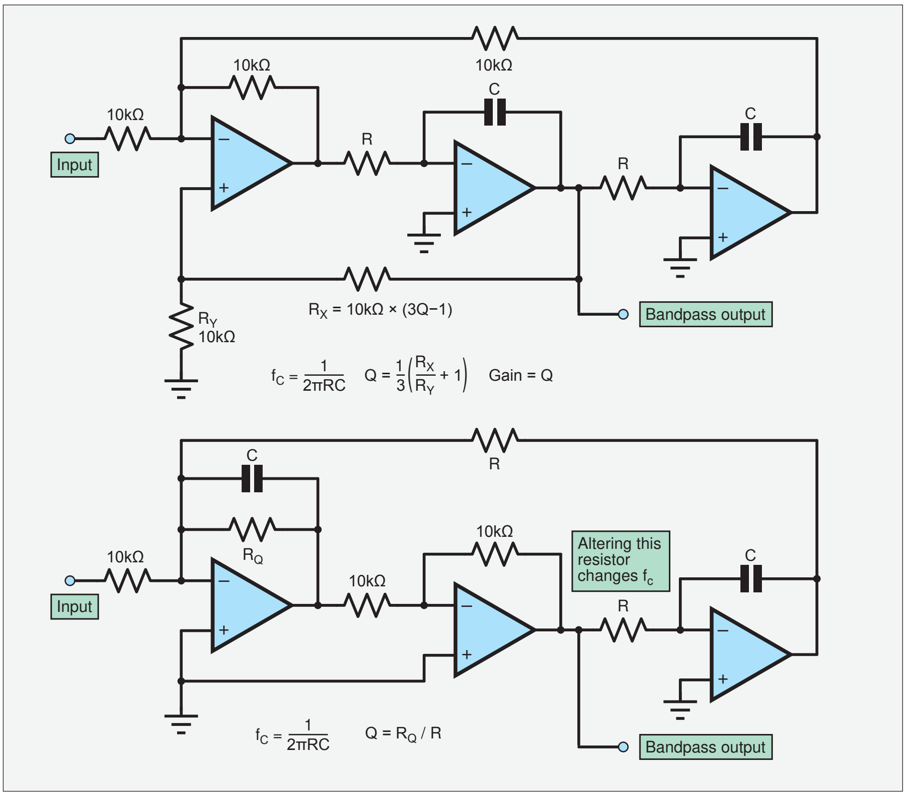


Fig.7. The state-variable (above) and bi-quad (below) are the ideal building blocks for band-pass filters but are too complex and expensive for the multiple channels needed in a vocoder. However, for a vocoder with a few channels where the parameters need front panel controls, the bi-quad would be used.

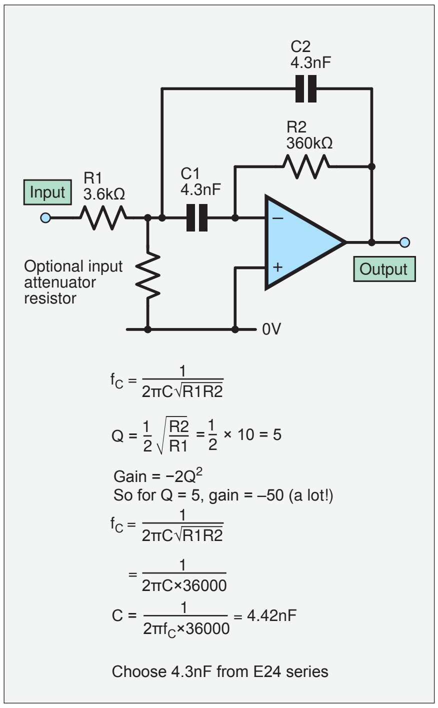


Fig.9. The multiple-feedback filter, the simplest and cheapest. High gain makes it noisy. Two of these are needed to make the flat-topped filter. An input attenuation resistor to ground is often added, but the parallel resistance of the attenuator must still equal 3.6kΩ to avoid altering the response.

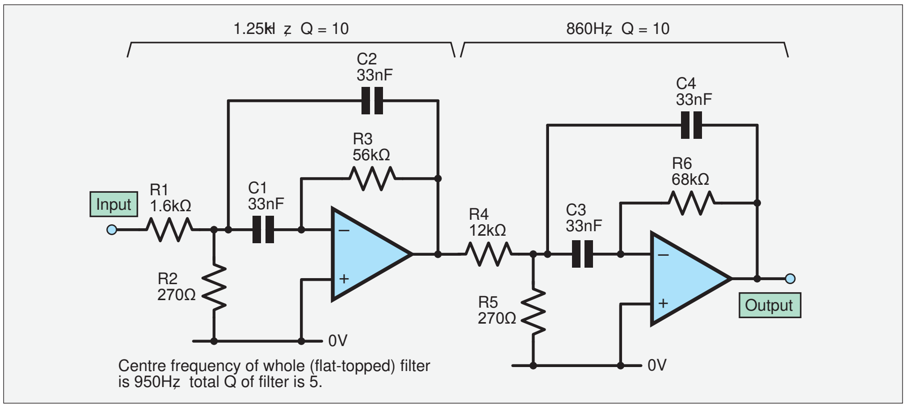


Fig.10. A double-tuned band-pass filter circuit. One-third octave bandwidth with a centre frequency of 950Hz.

skirts. The result is much less peaky than a single filter and the out-of-band attenuation is double. The frequencies of the two filters are selected so as to correspond with what would be the –3dB points of a single filter. So for a 500Hz (centre frequency) double-tuned filter, the resonant frequencies (f_L and f_U) of the two filters would be 472HZ and 429Hz respectively. Note the slight asymmetry is due to the neeed to match (approximately) exponential frequency scaling. The double-tuned technique was originally developed for radio intermediate frequency transformers in superhet radios.

Multiple-feedback band-pass filter

The circuit shown in Fig.9 is the simplest band-pass filter; it's called the multiple-feedback filter because it uses two feedback paths. The disadvantage of this simple design is that the gain is 50 at a Q of 5, which is a problem because the op amp output will clip.

For third-octave double-tuned filters, the Q of each section needs to be 8.8, the practical limit for this type of filter. A circuit of a double-tuned multiple feedback filter is shown in Fig.10. The signal has to be attenuated on the input to the band-pass filters by even more than for a single filter. Therefore, the signal-to-noise ratio is poor. This has little effect in speech filters, since these are followed by full-wave rectifiers to produce smoothed DC control voltages for the VCAs. On the carrier/output section it is more significant, but it is only the output filters after the VCAs that contribute continuous noise. The noise from the filter stages preceding the VCAs is effectively muted when the VCAs cut off.

An alternative filter, which I intend to try at some point in the future, is the dual-amplifier band-pass configuration (DABP) developed by Sedra and Espinoza, and shown in Fig.11. This avoids the excessive noise gain problem of the multiple-feedback type. However, it adds 24 extra op amp sections to the vocoder, so we'll leave this for a future upgrade.

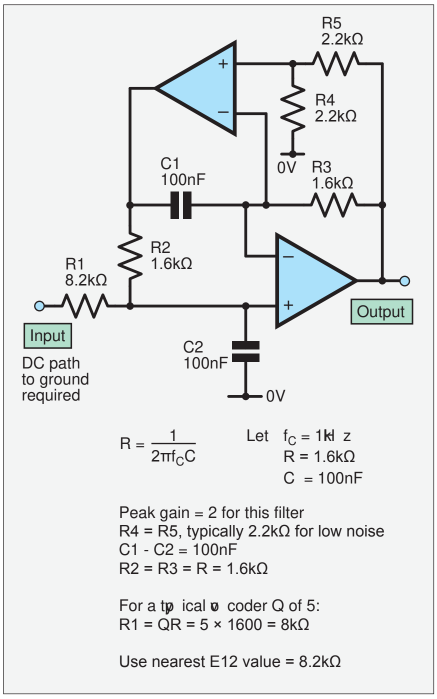


Fig.11. The dual-amplifier band-pass configuration filter. Twice as many op amps, but more controlled gain.

High / low-pass filters

The high-pass and low-pass filter topology is basically the same as the vocoder channel topology (Fig.4), but with the band-pass filters replaced with fourth-order high-pass and low-pass filters. These filter circuits are shown in Fig.12.

These extra modules are only needed if a full frequency response is desired for a stand-alone-performance vocoder. In a band or studio situation where there is a bass player and drummer for example, the vocoder only needs to cover the speech band – ie, the middle frequencies of around 100Hz to 8kHz. A treble-boosted portion of direct speech signal can be mixed in to give a clean top-end. By eliminating the high-pass and low-pass vocoder modules one saves £70, and gains a less cluttered mix into the bargain.

Next month

That's all for this month – in the next issue we'll start to look in more depth at the thinking behind the design.

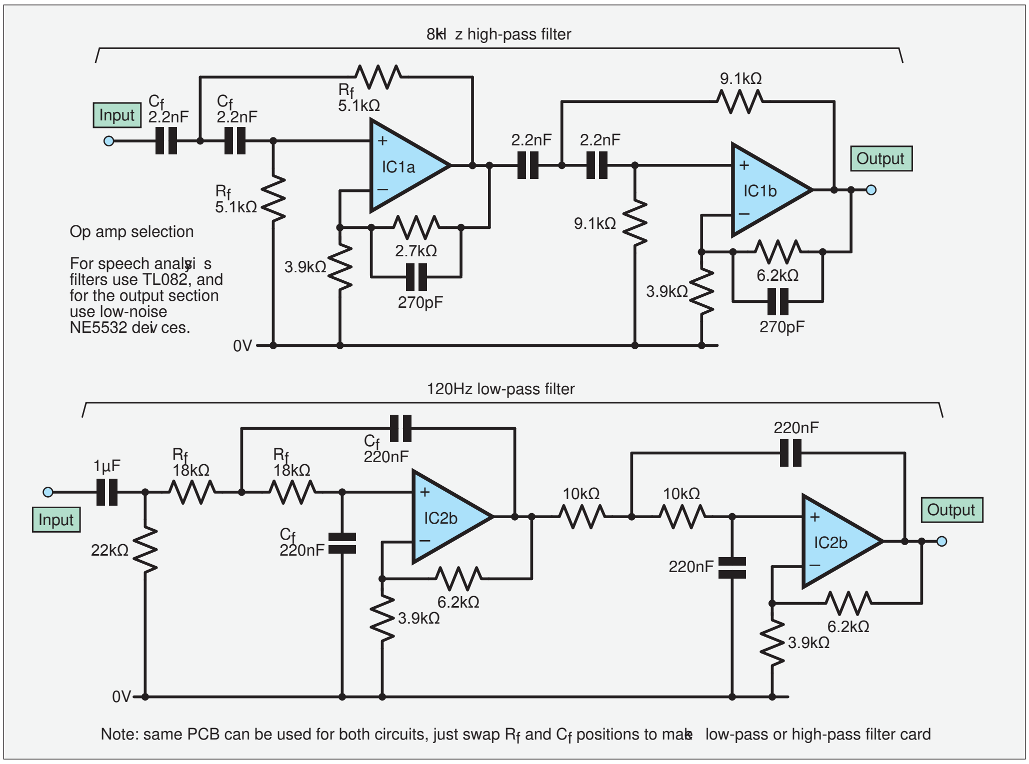
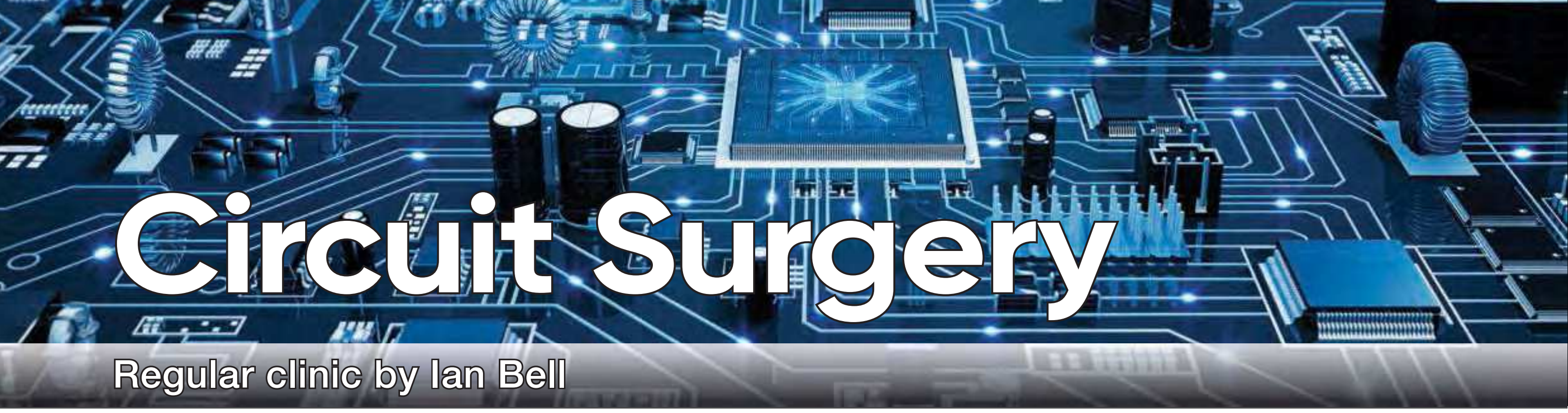


Fig.12. For the top and bottom bands of a full-range vocoder, fourth-order low-pass and high-pass filters are used. These require a different PCB and are not needed for a speech-only vocoder.



Capacitor Dielectric Absorption

John Curtin asked a question about capacitors on the EEWeb forum: 'I have a question, but have not found an answer on the Net. An electrolytic capacitor, if it is in short circuit for a long time, discharges and its voltage goes to zero. But if you remove the short circuit, the voltage on the terminals slowly rises again. Why? Thank you very much.' As was quickly pointed out by forum expert Peter Traneus Anderson, this effect is called 'dielectric absorption'. Other names are also used, such as 'voltage recovery' and 'dielectric soakage'.

As John indicates, dielectric absorption is a non-deal phenomenon which occurs in capacitors – it is not exclusive to electrolytic capacitors but tends to be larger in capacitors of this type. The effect is most obvious if a capacitor is charged for a long time, then discharged to zero volts very quickly and immediately left open circuit. After this, both real and ideal capacitors will start at zero volts. An ideal capacitor will stay at zero volts, but a real capacitor will develop a non-zero 'recovery' voltage across it after a time delay due to dielectric absorption.

Dielectrics

As the name suggests, dielectric absorption is due to the properties of the capacitor's dielectric – the insulating material which is placed between the two conducting plates to form the capacitor. The dielectric is an insulator, so unless the breakdown voltage is exceeded, very little electric charge flows through it when a voltage is applied across the capacitor. However, significant internal charge redistribution does occur in the form of polarisation of dipoles. Dipoles are parts of a material which can have a distribution of charge (in simple terms, positive at one end and negative at the other). This can occur at the atomic, ionic, molecular and higher structural levels (such as cells in biological tissues).

An electric field applied to a dielectric (in our case, a voltage applied across a capacitor) causes the dipoles to rotate to be more aligned with the applied field (with no field applied they are randomly aligned) – see Fig.1. The field created by the aligned dipoles interacts with the applied field and threby allows the capacitor to store more energy (hold more charge) than if there was just a vacuum between the plates. The more polarisation, the more additional storage (so more capacitance for a given physical size of device). When the field is removed (capacitor discharged) the dipoles return to their random orientation – referred to as 'relaxation'.

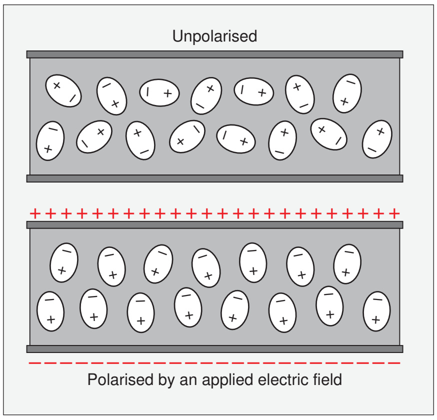


Fig.1. Polarisation of dipoles in a capacitor dielectric.

The polarisability of the material is characterised by its permittivity. The permittivity of a material can be stated as an absolute value (symbol ε – Greek epsilon), or as relative permittivity (ε_r), which is relative to the permittivity of free space (vacuum) (ε_0); that is, $\varepsilon = \varepsilon_r\varepsilon_0$. The symbol κ (Greek kappa) is also used for relative permittivity. Materials used for capacitor dielectrics tend to have a large relative permittivity (sometimes referred to as 'high K') as this allows for smaller capacitors for a given capacitance value and voltage rating.

The polarisation of dielectrics is a complex process because there are typically multiple types of dipole within materials and the polarisation and depolarisation (relaxation) is not instantaneous. In simple terms, dielectric absorption is due to the fact that not all of the dipoles depolarise in the time taken to rapidly discharge a capacitor to zero voltage. They continue to hold charge ('absorbed' by the dielectric). Initially, if the capacitor is open circuited after discharge, this does not result in a voltage at the plates. However, the charge will get transferred to the capacitor plates over time as the dipoles depolarise. This will cause a charge build up on the plates leading to the recovery voltage that John asks about.

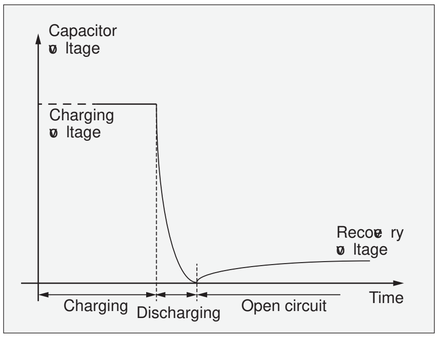


Fig.2. Behaviour of a capacitor exhibiting dielectric absorption.

Capacitor behaviour and measurement

The behaviour of a capacitor exhibiting dielectric absorption is illustrated in Fig.2, which is a graph of the voltage across a capacitor against time, undergoing the charge/discharge/open-circuit sequence just mentioned. Subjecting capacitors to this process can be used to make measurements to observe or characterise their dielectric absorption. A circuit concept for doing this is shown in Fig.3. The specifics of implementation may vary; for example, in

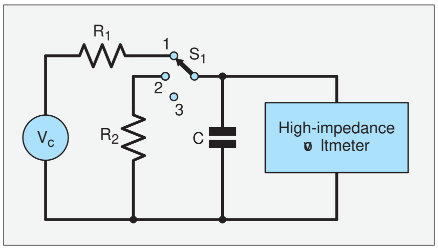


Fig.3. Circuit concept for observing or measuring capacitor dielectric absorption.

Simulation files

Most, but not every month, LTSpice is used to support descriptions and analysis in *Circuit Surgery*.
The examples and files are available for download from the *PE* website.

the simplest cases manual switching could be employed, but automated switching using relays and various improvements to the basic idea can be incorporated.

In the first part of the graph the capacitor has been charged for a long time, so the voltage across it is effectively equal to the charging voltage. With the circuit in Fig.3 this is achieved by applying V_C through R_1 with S_1 in position 1. A 'long time' could be from minutes to hours. Typically, real capacitors would be charged to a suitable voltage (maybe their rated working voltage) when making dielectric absorption measurements, taking care to safely limit the initial current surge when charging is started. Applying the maximum working voltage to a discharged capacitor through too small a resistance (R_1 in Fig.3) may result in a damaging current surge (eg, by connecting the capacitor directly to a power supply at that voltage).

Once fully charged, the capacitor is discharged quickly relative to the charge time, this is achieved by connecting a suitable resistor across the capacitor (R_2 in Fig.3, with S_1 in position 2). The switchover from charge to discharge mode should occur quickly with the capacitor briefly open-circuit (S_1 break-before-make). The capacitor will hold its charge during this time. Capacitors tend to discharge slowly if left charged and open circuit, so if the aim is to observe dielectric absorption, the time between charge and discharge should be brief.

To allow the dielectric absorption phenomenon to be observed, the discharge time should be just sufficient to get very close to zero volts as quickly as possible. This means discharge through a low resistance (R_2) – but, for real tests, the current must be limited to a value that will not cause any problems. Standard measurements may use fixed times and resistance values for discharge to ensure the same conditions are used when comparing devices.

After the quick discharge the capacitor is disconnected from the discharge resistor so that it is open circuit (S_1 in position 3 in Fig.3). The capacitor voltage will rise, as shown in the final section of the graph in Fig.2. Initially, the increase will be rapid, but the rate of change will slow over time. The final voltage reached is the recovery voltage. The voltage across it over time can be measured using a voltmeter, which should have a very high input impedance if accurate results are required.

Dielectric absorption (DA) figures for capacitors are defined as the percentage of the recovery voltage (V_R) relative to the charging voltage (V_C) under specified conditions (charge and discharge time, discharge resistance, open-circuit time after which voltage is measured, temperature and so on):

$DA(\%) = (V_R/V_C) \times 100$

As well as describing the behaviour of a capacitor exhibiting dielectric absorption, the preceding discussion has sketched a basic approach to measuring the DA value. You will find references to MIL-C-19978D and EN 60384-1 as standard test produces, but their relevance in the context of modern electronic systems may be debatable. For further discussion on this, and a more comprehensive exploration of measuring dielectric absorption, readers might be interested in a paper published online by Leslie Green: *Practical Exploration of Dielectric Absorption in Capacitors for the 21st Century* (see: **http://lesliegreen.byethost3.com/articles/Dielectric.pdf**).

Effects of dielectric absorption

The recovery voltage due to dielectric absorption varies significantly for different types of capacitors. Capacitors with polymer dielectrics, such as polyphenylene sulphide, polypropylene and polystyrene tend to have the lowest DA values, which can be well below 0.1%. Dielectric absorption is higher for ceramic capacitors (around 0.5 to 2.5%) but depends on the type of ceramic, with Class 1 C0G (NP0) having lower DA values than class 2 X7R. The highest DA values are for electrolytic capacitors, which can reach 10% or 15%.

Dielectric absorption makes itself felt (literally in some cases) in different ways depending on the type of capacitor and the application. For high-value, high-voltage capacitors there is a possibility for the recovery voltage to deliver a significant, even lethal electric shock to someone handling the capacitor. Such capacitors are usually shipped and stored shorted out, but can still recover after the short is removed, so in situations where this is a potential problem people need to follow appropriate safety procedures.

Dielectric absorption can also cause problems in some signal processing applications, with one of the most important examples being sample and hold circuits (see Fig.4) – these are commonly used to provide steady input voltage samples to analogue-to-digital converters (ADCs). The input signal is sampled by rapidly charging a capacitor to the value of the input voltage at the instant that a conversion is required (by closing the sample switch briefly). The sample switch is then opened, and the capacitor holds the value steady while the conversion is performed, buffered by a high-impedance unity-gain amplifier to prevent the capacitor from discharging.

When used with a single input signal, in many cases the voltage will not change much between successive samples (with a smoothly changing signal) and dielectric absorption will not be a problem. However, some signals will have abrupt changes, and it is common for ADCs to be used with multiplexers so that a single ADC can convert signals from multiple input channels (see Fig.4). Here it is quite likely that the capacitor will be charged to near the maximum input voltage for one sample and close to zero on the next – if different channels are at opposite ends of the input voltage range. This creates a situation very similar to that shown in Fig.2. Even if the conversion takes place relatively quickly compared with the time taken to reach the full recovery voltage, the effect of dielectric absorption will result in an unwanted change in voltage on the sample capacitor – remember, the rate of change is relatively fast initially. This can result in errors in the converted value. For example, for a 12-bit ADC a change of only 0.025% of the full-scale voltage is sufficient for a change of the converted value (by 1 least-significant bit).

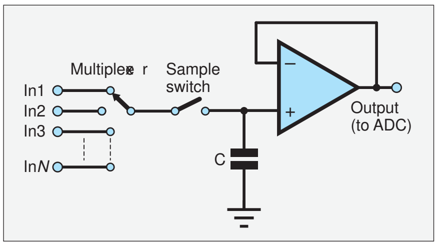


Fig.4. Sample and hold with input multiplexer. The switches are electronic (typically MOS transistors).

Another circuit where dielectric absorption may have a significant effect is the op amp integrator (see Fig.5), one of the 'standard' op amp applications. The output of this circuit is proportional to the integral of the input voltage over time. In some situations, it is necessary to reset the integrator (or more precisely, set the initial conditions of the integration). This is shown in Fig.5, implemented as a switch discharging the capacitor, although setting a specific voltage other than zero may be required. The situation for the capacitor in Fig.5 is again very similar to the scenario in Fig.2. The capacitor may be at a relatively large value when the reset (discharge) occurs, and the resulting recovery voltage will be added as an error to the output from the circuit after the switch is opened.

Model circuit

It is possible to model the dielectric absorption of a capacitor using an equivalent circuit of the form shown in Fig.6, which

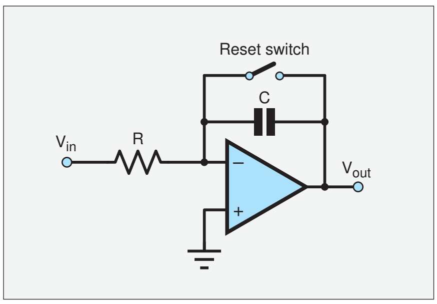


Fig.5. Basic op amp integrator circuit.

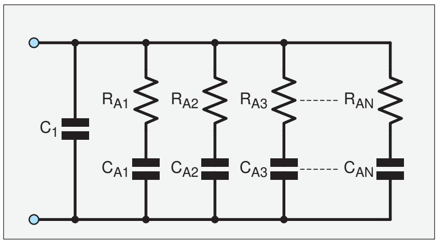


Fig.6. Equivalent circuit model for capacitor dielectric absorption.

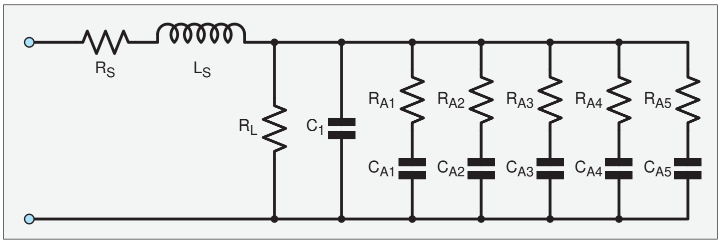


Fig.7. Equivalent circuit model with five *RC* pairs used for dielectric absorption and other non-ideal capacitor characteristics.

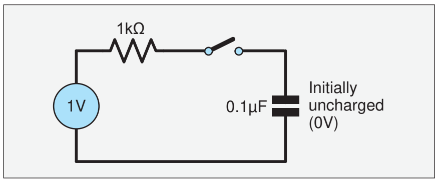


Fig.8. Example circuit for *RC* time constant – charging.

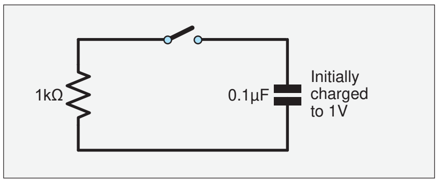


Fig.9. Example circuit for *RC* time constant – discharging.

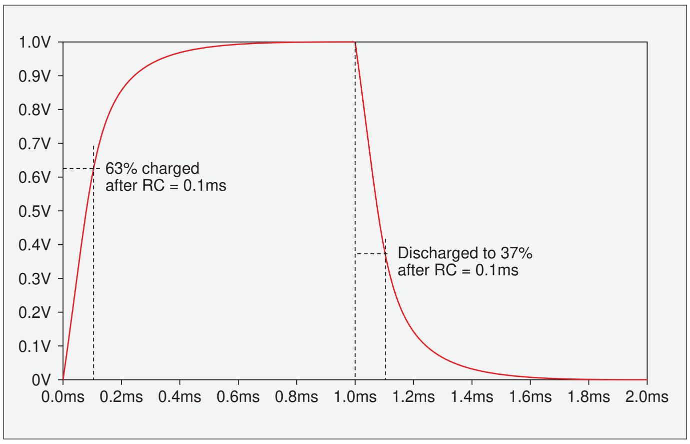


Fig.10. Example *RC* charge and discharge waveforms based on Fig.6 (0 to 1.0ms) and Fig.7 (1.0 to 2.0ms).

was published by Paul Dow in 1958 in a paper in the *IRE Transactions on Electronic Computers*. The model helps us understand the behaviour of capacitors with respect to dielectric absorption and simulate these effects at circuit level without needing a deep knowledge of the physics of dielectrics at the atomic and molecular level. The equivalent circuit comprises a set of N RC circuits (R_{A1} and C_{A1} to R_{AN} and C_{AN}) in parallel with an ideal capacitor, C_1. C_1 has the rated value of the capacitor. The capacitors modelling the absorption (C_A) will have a smaller value than C_1 (the total absorption capacitance relative to C_1 determines the final recovery voltage). The resistor values will typically be large (megohms to giga-ohms) to account for the slow change of the recovery voltage.

Intuitively, the model in Fig.6 explains the behaviour shown in Fig.2. The long charge time ensures that all the capacitors in the circuit are charged to the applied voltage, despite the large resistor values in the absorption part. The quick discharge through a relatively small resistor removes the charge from C_1, but the large resistors in series with the other capacitors mean that little charge is removed from them. During the open circuit period the capacitors in the equivalent circuit will share charge – specifically charge will move off C_{A1} to C_{AN}, and onto C_1 until the voltages on the capacitors equalise (if the capacitors are at different voltages this difference will appear across the resistors, causing current to flow, moving the charge between the capacitors). We observe the voltage on the terminals (across C_1) rise – this is the recovery voltage, or the capacitor appearing to recharge itself. The large resistor values mean that this will take a relatively long time compared with the discharge of C_1 just performed.

The circuit in Fig.6 only models the effect of dielectric absorption, but other components can be added to cover other non-deal characteristics of capacitors if required. An example is shown in Fig.7, which uses five RC pairs to model absorption and also includes equivalent series resistance (ESR) (R_S), equivalent series inductance (ESL) (L_S) and leakage resistance (R_L).

Curves and time constants

The concept of RC time constants is important in the dielectric absorption equivalent circuit. This can be illustrated by the basic charge and discharge circuits shown in Fig.8 and 9. The well-known exponential charge and discharge curves for these ideal RC circuits are shown in Fig.10. The time constant for an RC circuit is the value obtained by multiplying the resistor and capacitor values together (the result has units of time). For example, in the circuits in Fig.8 and 9 we get $RC = 1.0\text{k}\Omega \times 1.0\mu\text{F} = 0.1\text{ms}$. When a capacitor is charged via a resistor from a fixed voltage, the voltage across the capacitor will be 63% of the applied voltage after time RC has elapsed (as seen in Fig.10). Similarly, when a capacitor is discharged through a resistor the voltage will reach 37% of the initial value after the time constant time has elapsed.

The time constants of each of the set of RC circuits in the equivalent circuit in Fig.6 typically increase by a factor of ten along the sequence from 1 to N. Time constants for dielectric absorption can be very long – implying many hours or more to approach the final recovery voltage. It is also worth

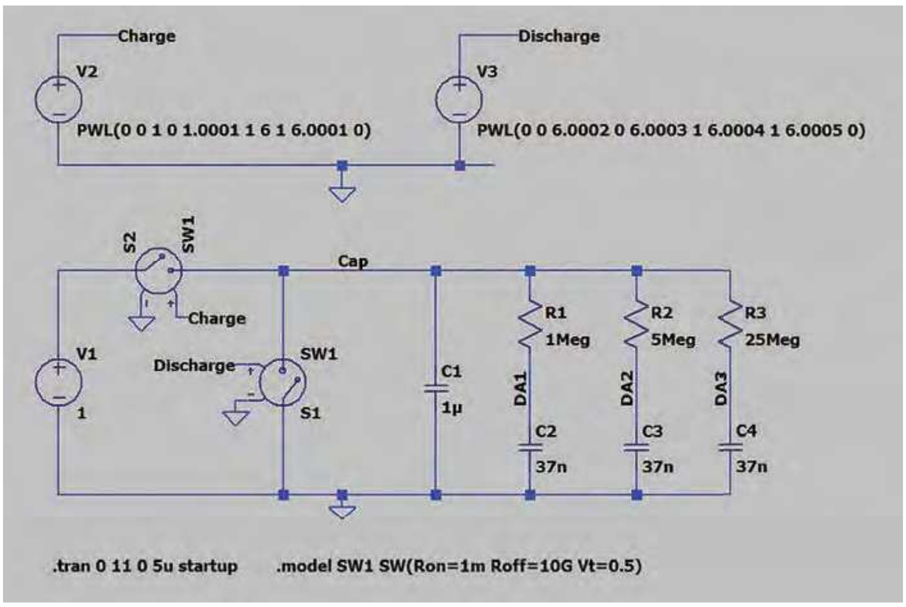


Fig.11. Simulation example.

noting that a dielectric absorption means that for real capacitors the charge and discharge curves do not follow the perfect exponential characteristic of ideal devices.

Simulation example

The circuit in Fig.11 is an LTspice schematic which can be used as a starting point to explore the equivalent circuit in Fig.6. The switches (S1 and S2) are controlled by the pulse waveforms from V2 and V3 to operate the circuit in a similar way to that described earlier for Fig.3. The simulation is configured with very idealised switches (very high off and very low on resistance, as defined by the SW1 model), so this is not a realistic circuit in terms of what is done to the capacitor. The switch model can be changed, or resistors added (as in Fig.3) to make this more like a real test circuit.

Just three absorption *RC* circuits are used to keep the simulation simple, with their time constants increasing by a factor of five, so the simulation does not have to be too long. The longest of these is R3C4 which is just under one second. Thus, it is reasonable in this context that charging the capacitor for a 'long time' has a duration of five seconds – an *RC* circuit charges to more than 99% of the applied voltage in 5*RC*. V2 is configured to close switch S2 from simulation time 1s to 6s to charge all the capacitors in the equivalent circuit (see the simulation waveforms in Fig.12). Immediately after this, C1 is discharged very quickly by closing S1 for 200µs, after which both switches are open. The short discharge duration and relatively large values of R1 to R3 compared to S1's resistances mean that C2 to C4 will lose very little charge while S1 is closed – this can be confirmed by zooming in on the waveform.

It is difficult to see the details of the recovery voltage in Fig.12, so Fig.13 provides a zoom-in. The final voltage is 100mV, which is about 10% of the applied voltage. We can also obtain this figure by analysing the charge sharing – the total absorption capacitance is 3 × 37nF = 111nF. The overall total capacitance with C1 is 1111nF. With C1 starting at 0V and C2 to C4 at 1V, the four capacitors sharing charge will result in a final voltage of (111/1111) × 1V = 100mV. The recovery time is dominated by the R3C4 time constant, and we simulate the recovery for 5 × R3C4, so we are close to the final value.

The simulation can be modified to study the effect of different dielectric absorption characteristics or measurement scenarios. For example, Fig.14 shows the effect of adding a 10MΩ resistor across C1 – which could be the resistance of a voltmeter. This effect is to discharge C1 at the same time as it is being charged from the absorption capacitors, so the recovery voltage peaks and then decreases.

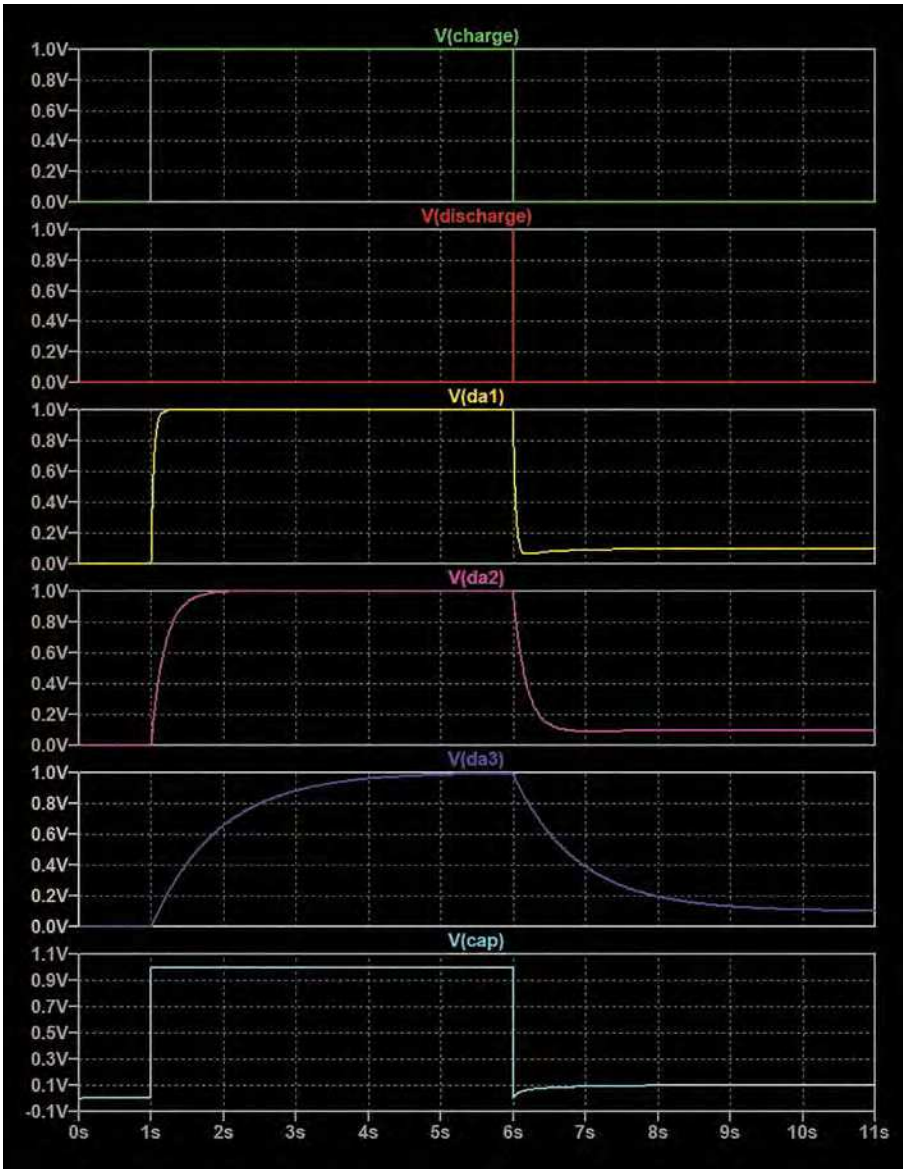


Fig.12. Simulation results from the circuit in Fig.11.

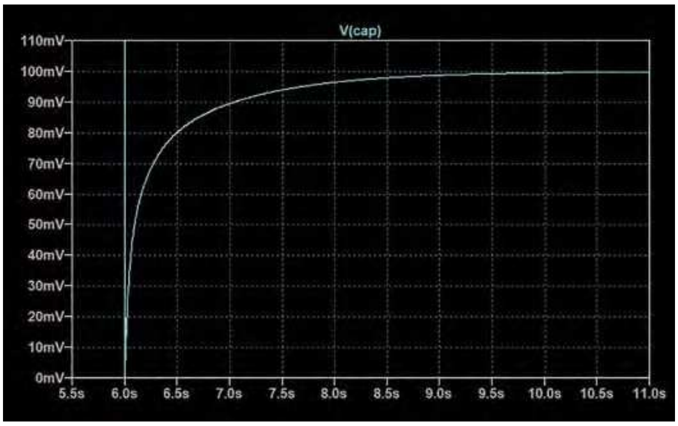


Fig.13. Zoom in to show recovery voltage.

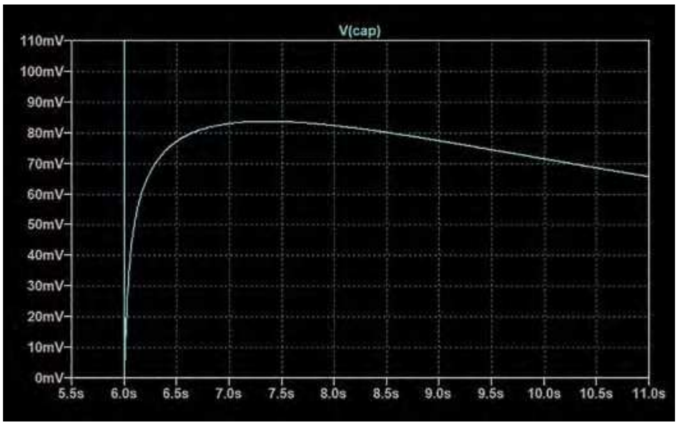


Fig.14. Recovery voltage with 10MΩ resistor across C1.

Max's Cool Beans

By Max the Magnificent

Flashing LEDs and drooling engineers – Part 21

I live in a maelstrom of weird and wonderful happenings. For example, I often get questions pertaining to my projects like, 'What on earth possessed you to come up with that?' (I can hear my mother's voice ringing in my ears as clearly as if she was in the room with me, and now my wife – Gina the Gorgeous – has taken over for her). Well, if you are at all interested in discovering the answer to this question, you'll be delighted to discover that I was recently interviewed in a *Fish Fry Special Edition: Makers Today!* podcast hosted by the inimitable Amelia Dalton (**https://bit.ly/3C6bIiS**).

Have you ever paid any attention to the indexes in technical books? I don't know about you, but I find it to be incredibly frustrating when I return to a book looking to find something to use as a reference but fail to find what I'm searching for in the index. The reason I mention this here is that, at the time of this writing, three of my chums – Adam Taylor, Dan Binnun and Saket Srivistava — recently finished writing a book called *A Hands-On Guide to Designing Embedded Systems* (**https://bit.ly/3hnu9If**). They kindly sent an early PDF copy for me to peruse and ponder. I immediately had a quick skim before taking a deep dive, and my eye was caught by the index, whose sole contents were a note saying, 'Index Goes Here.' Since I spend a humongous amount of time hand-crafting the indexes for my own books, I was moved to pen a column on this topic: *Just Call Me an Indexing Fool* (**https://bit.ly/3C0prba**).

I was just chatting to Matt Pulzer who is the illustrious publisher-editor of *Practical Electronics*. Matt is a font of esoteric erudition, so I was delighted to add to his treasure trove of trivia by informing him of a nugget of knowledge pertaining to the phrase 'It was a dark and stormy night...' As you may recall, this was used to comic effect in the *Peanuts* cartoons by Charles Schulz (**https://bit.ly/2XfgOdS**). In fact, Snoopy wrote so many stories starting this way that these openings ended up forming a book in their own right (**https://amzn.to/3C5XVJ9**).

This archetypal example of a florid, melodramatic style of fiction writing comes from the opening sentence in the 1830 novel *Paul Clifford*, which was penned by English novelist Edward Bulwer-Lytton. In fact, this opening sentence is so bad that it led to the annual Bulwer-Lytton Fiction Contest, which challenges participants to write an atrocious opening sentence to the worst novel *never* written. Personally, I class this competition (**https://bit.ly/2Xh8bQ3**) alongside the Darwin Awards, which salute the improvement of the human genome by honoring those who accidentally remove themselves from it in a spectacular manner (**https://bit.ly/2YNUb0N**), and the Ig Nobel awards, whose stated aim is to honor achievements that first make people laugh, and then make them think (**https://bit.ly/3C4Qj9Q**).

Are you nuts?

As I've mentioned on occasion, I'm a big fan of steampunk, which is a subgenre of science fiction that incorporates retro-futuristic technology and aesthetics inspired by 19th-century industrial steam-powered machinery. Many of my projects include brass control panels or facias, in which case I will also use brass fasteners. For some of my projects, like my *Victorian Spectrum Analyser* and my *Countdown Timer* (**https://bit.ly/3yYmypv**), whose job is to display the years, months, days, hours, minutes and seconds to my 100th birthday celebrations, which will kick off at 11:45am British Summer Time on 29 May 2057 (mark your calendar), I've used brass acorn nuts to good effect. With other projects, I've used machine screws with a variety of head shapes and drive styles as appropriate.

There is, of course, an incredible variety of head shapes available, where each of these head shapes may be available with a diversity of drive styles, only a small selection of which are presented here (Fig.1). Just to increase the fun and frivolity, combinations of drives are also available, like Phillips + slotted and hexagonal + slotted. If you are aiming for historical authenticity and attempting to avoid the appearance of anachronisms, then you can't go wrong with slotted or hexagonal heads (also, acorn nuts, which are not shown here). Just about everything else is an artifact of the 20th century, including Square (Robertson) drives, which were invented by Peter Robertson in 1907, Hexagonal (Allen) drives, which were invented by William Allen circa 1910, Phillips drives, which were invented by Henry Phillips in 1936, and Torx screws, which were invented by the Camcar Textron company in 1967.

For my own projects, I predominantly use slotted screws, although I'm not averse to Allen or Robertson drives, and

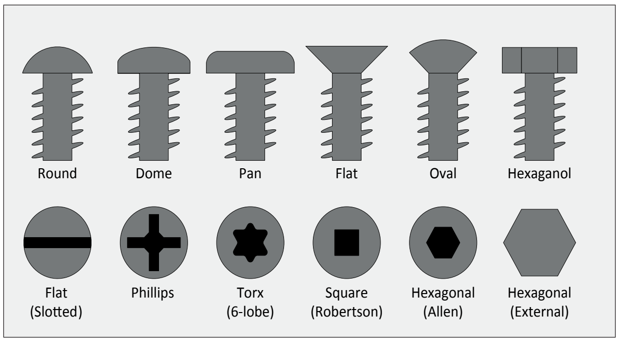


Fig.1. Machine-screw heads (top) and drives (bottom).

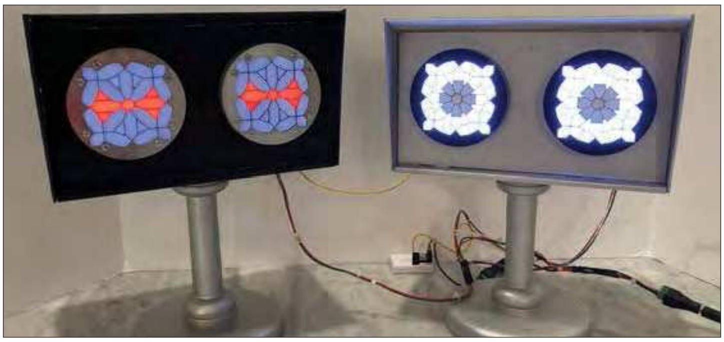
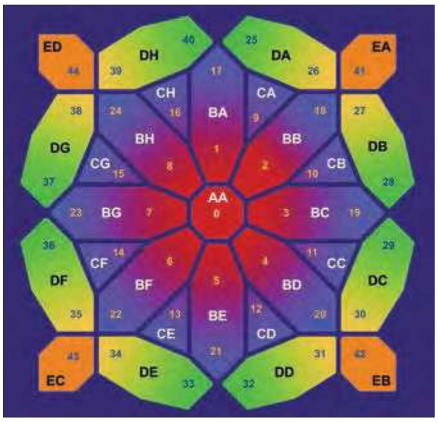
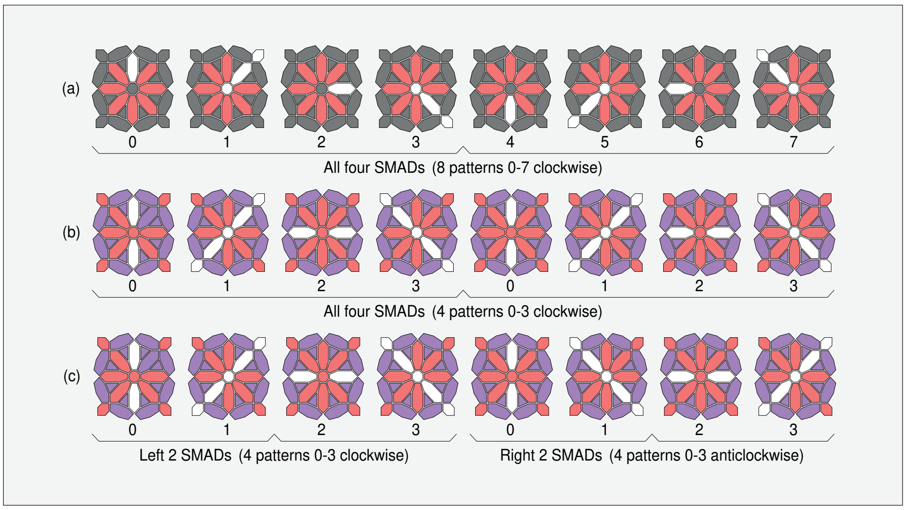


Fig.2. Two pseudo robot heads showing different types of 'eye' effects.





(Above) Fig.3. *SMAD* segment map.
(Right) Fig.4. Variations on a simple windmill effect.

sometimes even hexagonal heads should the occasion demand. On the other hand, although I love them for household tasks, you will never, ever find me using a Phillips screw on one of my steampunk extravaganzas.

Of course, how you create your own projects is totally up to you; it's not for me to cast aspersions (not the least that my throwing arm isn't what it used to be). Having said this, I think a little attention to detail goes a long way. In the case of my two pseudo robot heads, each featuring two of our *SMADs* (*Steve and Max's Awesome Displays*), for example, I'm using eight slotted drive, pan-headed machine screws to attach each of the *SMADs* to the front panels (Fig.2). Furthermore, although the side panels are actually attached to the main bodies of the heads (if you see what I mean) using hot glue, as seen in the image on the front cover of this issue, I've still augmented each panel with seven screws, just to convey the impression that they are more robust and sophisticated than is, in fact, the case.

One last point on this subject: the silver head has black *SMAD* shells and facias while the black head has silver (well, nickel-coloured) *SMAD* shells and facias. As we just saw, the side panel screws on the silver head are a regular steel colour, while their equivalents on the black head are black. Similarly, the screws holding the silver *SMADs* on the black head are steel, while the screws holding the black *SMADs* on the silver head are black. In the not-so-distant past, I used to spend an exorbitant amount of time trying to track down black versions of various fasteners. More recently, unless I'm using brass, I simply purchase steel versions of everything and then use Gun Blue if I need any to be black (**https://amzn.to/391DKj6**).

Feast your eyes

In previous columns, I've mentioned the idea of using different *SMAD* segments and colours to represent eyes. In Fig.2. we see two versions I just threw together. On the left, I'm using blue to represent the eyeball and red to denote the pupil; on the right, I'm using white to signify the eyeball and black to act as the pupil.

I'm going to continue experimenting with different representations, including making the eyes look left (west), right (east), up (north), and south (down). I'll probably experiment with looking northwest, southwest, northeast, and southeast, also.

Furthermore, when the eyes are looking directly forward, as shown in Fig.2, it might be interesting to see if we can achieve some sort of blinking effect. All of this is something I'm noodling over in the back of my mind, but I'd love to hear your suggestions.

Lest we forget

As a reminder, our *SMAD* segment map is shown in Fig.3. The numbers, which are applicable to both types of *SMAD* shell (29-segments and 45-segments), refer to the positions of the LEDs in the string, while the letter combinations are the names we use to identify the segments in just the 29-segment versions.

In my previous column (*PE*, October 2021), we ended up with all four *SMADs* displaying a simple windmill effect comprising eight patterns (Fig.4a). The key part is where we create an array containing the LEDs we wish to light up in each pattern. For this first effect, we defined `NUM_PATTERNS_IN_EFFECT` as being 8 (this corresponds to the fact that we have 8 'spokes,' as illustrated in Fig.3), and `MAX_LEDS_IN_EFFECT` as being 4 (there are two LEDs in each spoke). We then created a two-dimensional array of 8-bit integers called `EffectMap[][]`, that is comprised of `NUM_PATTERNS_IN_EFFECT` (ie, 8) rows, each containing `MAX_LEDS_IN_EFFECT + 1` (ie, 5) items, and we initialised this array as follows:

```
{
  {2, 1, 17,  0,  0}, //    BA
  {4, 0,  2, 18, 41}, // AA BB EA
  {2, 3, 19,  0,  0}, //    BC
  {4, 0,  4, 20, 42}, // AA BD EB
  {2, 5, 21,  0,  0}, //    BE
  {4, 0,  6, 22, 43}, // AA BF EC
  {2, 7, 23,  0,  0}, //    BG
  {4, 0,  8, 24, 44}  // AA BH ED
};
```

Remember that the first value in each row tells us how many LEDs we wish to light for this pattern. The remaining values in the row are the numbers of the LEDs (any 0s on the right are just placeholders). You can remind yourself as to how all this worked by downloading the code (file **CB-Oct21-05.txt**) from the October 2021 page of the *PE* website at: **https://bit.ly/3oouhbl**

Take your turn

At the end of my column last month, I invited you to perform some thought experiments of your own. The first was to consider how we might go about modifying our latest program so that it has four patterns, each with two arms at 180° to each other. You can find my solution in file **CB-Nov21-01.txt** on the November 2021 page of the *PE* website at: **https://bit.ly/3oouhbl**

Although not part of our main mission, I added a third 'Deep Background' colour of purple just to make life a little more interesting (Fig.4b). As you will see, implementing the new effect is easy peasy lemon squeezy. First, we change `NUM_PATTERNS_IN_EFFECT` to be 4 and `MAX_LEDS_IN_EFFECT` to be 6. Next, we redefine the contents of our `EffectMap[][]` array as follows:

```
{
  {5,0, 1,17, 5,21, 0}, // AA+BA+BE
  {6,2,18, 6,22,41,43}, //    BB+BF+EA+EC
  {5,0, 3,19, 7,23, 0}, // AA+BC+BG
  {6,4,20, 8,24,42,44}  //    BD+BH+EB+ED
};
```

Our next thought experiment was to ponder how we might modify this latest incarnation such that the windmill patterns on the left-hand robot head's eyes rotate clockwise while the patterns on the right-hand robot head's eyes rotate anticlockwise (Fig.4c).

You can find my solution in file **CB-Nov21-02.txt**. To be honest, although my code functions as planned, it feels a little 'clunky' to me and I have a sneaking suspicion that this could be implemented in a more elegant fashion. Perhaps you might take a look and offer some suggestions.

The final countdown

The last problem I posed was to implement a sort of counter that I think of as an 'Alien Countdown' (I had a disturbed childhood). The idea here is to return to a version of our original 8-pattern windmill effect (Fig.4a) but with our new colour scheme (Fig.4b and Fig.4c).

What we want to do is to create a program involving two or more *SMADs* that commences with them each displaying a single windmill arm pointing upwards. We start with the least-significant *SMAD* (the one on the right) spinning its arm clockwise, completing four full rotations each second (I originally specified one full rotation a second, but the effect looks more exciting and interesting when we speed it up).

Every time this arm returns to its vertical position, the arm on the adjacent *SMAD* advances by one position (pattern). Similarly, every time the arm on the second *SMAD* returns to its vertical position, the arm on the third *SMAD* advances by one position, and so on for all of the *SMADs* in the chain.

You can find my solution in file **CB-Nov21-03.txt**. In this case, I have to admit that I'm rather 'chuffed' with the way things turned out. Stripped of the nitty-gritty code, the main loop is as follows:

```
int  iSmad = 0;
bool done  = false;

do
{
  int smadOffset = iSmad * NUM_NEOS_PER_SMAD;

  // For current SMAD
  // Set Neos in old pattern to background color
  // <code goes here>

  // For current SMAD
  // Increment SMAD pointer to next pattern
  // <code goes here>

  // For current SMAD
  // Set Neos in new pattern to foreground color
  // <code goes here>

  // For current SMAD
  // Test to see if we've wrapped around
  if (SmadPtrs[iSmad] != NORTH)
    done = true;
  else
    iSmad = iSmad + 1;
} while ( (done == false) && (iSmad < NUM_SMADS) );
```

This means that all we have to do is change our definition of `NUM_SMADS` (the number of *SMADs* in the chain) to accommodate any number of 'digits' in our counter. As usual, for your delectation and delight, I've captured a quick video showing all of the effects discussed in this column (**https://bit.ly/3lJk1uN**).

What next?

There are so many things we can do with *SMADs* in general, and with my robot heads in particular, that I'm undecided as to which direction to take things next. Do we continue playing with variations on the simple effects we've seen thus far, building our (my) confidence that we (I) have a clue what we are (I am) doing? Alternatively, do we take the plunge and start experimenting with some eye-wateringly cunning colour effects? Do you have any thoughts you'd care to share at this point in the proceedings? As always, I welcome your sage comments, insightful questions, and helpful suggestions.

Cool bean Max Maxfield (Hawaiian shirt, on the right) is emperor of all he surveys at **CliveMaxfield.com** – the go-to site for the latest and greatest in technological geekdom.

Comments or questions? Email Max at: **max@CliveMaxfield.com**

Max's Cool Beans cunning coding tips and tricks

I fear this column is going to be reminiscent of one of my mother's tortuous tales, which typically kick off along the lines of: 'I bumped into Mrs. Greebles at the fishmongers the other day. You remember, she was the oldest of three sisters; the youngest, Beryl, was a strumpet, while the middle girl eloped with an Australian taxidermist and they had two sons who were terrified of bananas and...'

The amazing thing is that, after wandering so far out into the weeds that her audience starts to consider sending out a search party, she somehow manages to bring the story home to a triumphant conclusion: 'And that's why you should never name a walrus Wally!' (And people wonder why I drink.)

In an earlier *Tips and Tricks* (*PE*, July 2021), we noted that microcontrollers like the Teensy (which we are using to drive our 10-character, 21-segment V*ictorian Display*) and the Arduino (which I use for all sorts of things) contain three types of memory: Flash, SRAM, and EEPROM. The Flash, which is non-volatile (it remembers its contents when power is removed from the system), is used to store the program. By comparison, the SRAM, which is volatile (it forgets its contents when power is removed from the system), is where the program creates, stores, and manipulates variables when it runs. Also, we have a small amount of non-volatile EEPROM (electrically erasable programmable read-only memory) in which we can store modest quantities of long-term information.

In the case of the Teensy 3.6, we have 4KB of EEPROM (that's 4,096 bytes numbered from 0 to 4,095 in decimal or 0x000 to 0xFFF in hexadecimal). If we want to use this EEPROM in our programs, we first need to include a special library that's provided as part of the Arduino's integrated development environment (IDE) using the following statement:

```
#include <EEPROM.h>
```

Just to remind ourselves how this works, suppose we declare an integer variable called `Address` to which we assign a value of 0, along with an 8-bit (byte-sized) variable called `Data` to which we assign a value of 128. In this case, we can write the data into our EEPROM using:

```
EEPROM.write(Address, Data);
```

Contrariwise, we can read the data back out of our EEPROM using:

```
Data = EEPROM.read(Address);
```

In the aforementioned *Tips and Tricks* column, we also made mention of the fact that we are going to use the EEPROM to store a set of byte-sized unsigned integer values (we'll use the data type `uint8_t` that we introduced in the September 2020 *Tips and Tricks*) to keep track of a variety of user settings, such as the preferred formats in which to display the date and time.

Setting the scene

In the real world, we are going to have a bunch of 'settings bytes.' For the purposes of these discussions, however, let's assume we are working with a very limited subset comprising just ten bytes, which we can think of as being numbered from 0 to 9. As part of this, we have defined `NUM_SET_BYTES` as being equal to 10.

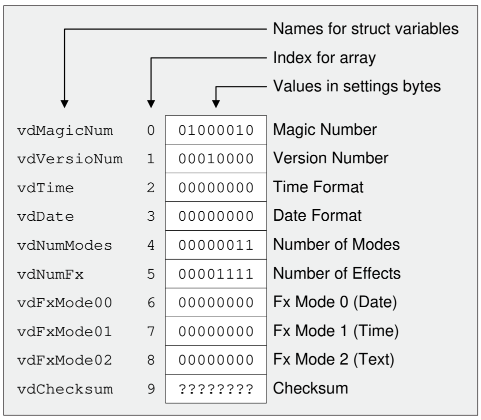


Fig.5. Example contents of DefSettings.

Let's also remind ourselves that we plan on having three copies of these settings bytes. First, we'll have a default set called `DefSettings` that's stored in the Flash memory as part of the main program. Next, we'll have a customised set that's been tweaked by the user and is stored in the EEPROM. Finally, we'll have a working set called `WrkSettings` that's stored in the SRAM; it is this latter set that will ultimately be used by the program once it's up and running.

Let's start by considering the default set of values stored in `DefSettings` using a generic visualisation (Fig.5). First, we have a magic number, whose purpose will be explained shortly, and whose value I've set to be 0x42 in hexadecimal (01000010 in binary), which is, as we all should know by now, *The Answer to the Ultimate Question of Life, the Universe, and Everything* (**https://bit.ly/pe-nov21-42**).

This is followed by a version number, which I'm considering to be presented in the form of two 4-bit nybbles (remember, 'two nybbles make a byte'), where the most-significant nybble holds the primary version number and the least-significant nybble holds the secondary version number. Since our nybbles contain 0001 and 0000, this means we are at version 1.0 in our example.

Next, we have the time format (0 = 12-hour, 1 = 24-hour) followed by the date format 0 = YYYY/MM/DD, 1 = MM/DD/YYYY, 2 = DD/MM/YYYY). These are followed by the number of modes and the number of effects. For the purposes of these discussions, let's assume that we have three modes numbered 0, 1, and 2 that display the time, date, and some scrolling text message, respectively. Observe that, for the purpose of this example, only two of these (the time and date) have associated format settings.

When it comes to effects, let's assume we have sixteen (numbered from 0 to 15). For example, effect 0 could be static white text on a black background, effect 1 could be static black text on a white background, effect 2 could be an animated rainbow of coloured text on a black background, and so forth. Next, we have three bytes containing the effects associated with each of our modes. As we see, we're assuming that the default will be to associate effect 0 (white text on a black background) with each mode.

Finally, we have a special value called a checksum, which will primarily be of use when we wish to determine if the contents of the EEPROM are valid or not. We will return to this point in a little while, but first...

Let's form a union!

As we previously discussed (yes, I'm afraid you really, *really* should re-read the July 2021 *Tips and Tricks*), we are going to define a new data type in the form of a `union` that we will call `Settings`. A union offers multiple ways to visualise and manipulate the same area of memory. In our simple case, sometimes we wish to think of things as a *structure* of byte-sized variables we've called `vds`, while other times we wish to think of the same memory locations as an *array* of byte-sized variables we've called `vda`. (The '`vd`' part of these identifiers stands for '*Victorian Display*,' while the '`s`' and '`a`' parts stand for 'structure' and 'array', respectively.)

```
typedef union
{
  struct
  {
    uint8_t vdMagicNum;
    uint8_t vdVersionNum;
    uint8_t vdTime;
    uint8_t vdDate;
    uint8_t vdNumModes;
    uint8_t vdNumFx;
    uint8_t vdFxMode00;
    uint8_t vdFxMode01;
    uint8_t vdFxMode02;
    uint8_t vdChecksum;
  } vds;

   uint8_t vda[NUM_SET_BYTES];
} Settings;
```

Next, we declare our `DefSettings` and `WrkSettings` variables as being of this union type, as shown below. The clever thing here is the way in which we initialise the values in our `DefSettings` variable as follows:

```
Settings DefSettings =
{
  .vds =
  {
    .vdMagicNum    = 0x42,
    .vdVersionNum  = 0x10,
    .vdTime        = 0x00,
    .vdDate        = 0x00,
    .vdNumModes    = 0x03,
    .vdNumFx       = 0x0F,
    .vdFxMode00    = 0x00,
    .vdFxMode01    = 0x00,
    .vdFxMode02    = 0x00,
    .vdChecksum    = 0x00
  }
};

Settings WrkSettings;
```

The `0x00` that we've assigned to the checksum value is for completeness only because we will be calculating a new value almost immediately (although this `0x00` value will never be used, neglecting to assign a default value here would bug me incessantly – also bugging me, why isn't there a word 'cessantly'?).

Address in EEPROM

Address	(a) Uninitialised	(b) Default	(c) Custom	
0x000	xxxxxxxx	01000010	01000010	Magic Number
0x001	xxxxxxxx	00010000	00010000	Version Number
0x002	xxxxxxxx	00000000	00000001	Time Format
0x003	xxxxxxxx	00000000	00000010	Date Format
0x004	xxxxxxxx	00000011	00000011	Number of Modes
0x005	xxxxxxxx	00001111	00001111	Number of Effects
0x006	xxxxxxxx	00000000	00000111	Fx Mode 0
0x007	xxxxxxxx	00000000	00001001	Fx Mode 1
0x008	xxxxxxxx	00000000	00000101	Fx Mode 2
0x009	xxxxxxxx	????????	????????	Checksum

Fig.6. Contents of EEPROM at different stages.

Stay constant

Actually, this provides another teachable moment. If we never intended to change any of the values in our `DefSettings` variable, we should (would) have declared it as being a constant value as follows:

```
const Settings DefSettings = {etc.}
```

This would confer several advantages, the first being that the compiler would 'throw a wobbly' if our code attempted to change any of the values. Also, any initialised variables that are *not* declared as being of type `const` are stored in the Flash program memory and then copied by the start-up code into SRAM (all this takes place before the user program is executed). By comparison, in the case of any initialised variables that *are* declared as being of type `const`, the system knows these values aren't going to change, which means they don't need to be copied into SRAM, thereby saving valuable SRAM resources.

Having said all this, we are planning on changing the checksum value in `DefSettings`, which explains why we didn't use the `const` keyword in this case.

The proof of the pudding

Remember that the structure and the array are intended to provide two different ways to manipulate and display the same ten bytes of memory in our union. But how do we know that this actually works? Well, as I've mentioned before, I'm a hardware design engineer by trade, and writing code is not my strong suit, so I performed a little test as follows:

```
for (int i = 0; i < NUM_SET_BYTES; i++)
{
    WrkSettings.vda[i] = DefSettings.vda[i];
}

for (int i = 0; i < NUM_SET_BYTES; i++)
{
    Serial.print(i);
    Serial.print("   ");
    Serial.println(WrkSettings.vda[i]);
}
```

When we initialised the values in our `DefSettings` variable, we used its structure aspect. Now, in the first loop above, we use its array aspect to copy these values from the `DefSettings` variable into the `WrkSettings` variable, after which we use the second loop to print the values from the `WrkSettings` variable. It works! (Phew!). If you wish, you can peruse and ponder this sketch in all its glory (file **CB-Nov21-04.txt**, which is available on the November 2021 page of the *PE* website at: **https://bit.ly/3oouhbl**).

Check my checksum

Let's remind ourselves that our last settings byte is used to contain a value called a 'checksum.' All we need to know at this point in our discussions is that a checksum is a small block of digital data that is derived from a larger block of digital data for the purpose of detecting errors that may have been introduced during the storage or transmission of the larger block of data. In our case, we are going to derive a byte-sized checksum value based on the contents of all of our other settings bytes.

The procedure by which we generate our checksum value is referred to as a 'checksum function' or a 'checksum algorithm.' A good checksum algorithm will generate a significantly different output value for even a small change made to the input data. I hate to do this to you again (I might remind you as to how we introduced this column), but I think you know what I'm going to say, which is that we will return to this point shortly, but first…

Meet my shell game

Have you ever seen someone perform a shell game? This involves sleight of hand in which three inverted cups, or nutshells, are moved about and the contestant must identify which one ends up with a pea or other object underneath. If you are the contestant in question, you are convinced you know where the pea is hiding… right up to the point where the shell you selected is shown to be empty and your hard-earned dosh bids you a fond farewell. Well, we are poised to do something similar with our settings bytes in general and our checksum byte in particular.

Before we proceed, let's make a decision that one of the very first things we will do when we initially run our program is to use our checksum function to generate a checksum byte based on all of the settings bytes (except the checksum byte itself) that are stored in our `DefSettings` variable. We will store the result in the checksum byte associated with `DefSettings`.

Now, let's assume that we've just taken a new microcontroller out of the box, plugged it into our system, and loaded our program into it. Since this is a new microcontroller, the EEPROM is going to be uninitialised, which means it will contain unknown values (Fig.6a). We've indicated these values as 'X' in this figure, but they would typically be all 1s (or, more rarely, all 0s).

When we run our program, one of the first things we'll do (after generating and storing the checksum byte in the `DefSettings` variable as discussed above) will be to check to see if the EEPROM contains valid settings data. How do we do this? Well, do you recall the magic number byte in Fig.5? We decided to load this byte with a value of 0x42, but any value other than 0x00 and 0xFF would suffice.

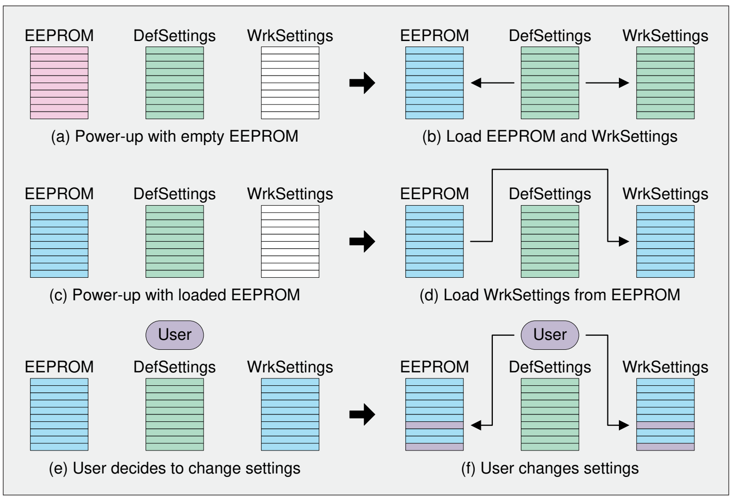


Fig.7. Alternative action sequences.

(a) Parity byte	(b) Sum complement	
01000010	01000010	Magic Number
00010000	00010000	Version Number
00000001	00000001	Time Format
00000010	00000010	Date Format
00000011	00000011	Number of Modes
00001111	00001111	Number of Effects
00000111	00000111	Fx Mode 0
00001001	00001001	Fx Mode 1
00000101	00000101	Fx Mode 2
01010110	10000100	Checksum

Fig.8. Simple checksum algorithms.

What we are going to do is read the byte from address 0x000 in the EEPROM to see if it matches the magic number byte in `DefSettings`. However, as we've already discussed, since this is the first time we've run the program, we are powering up with an empty EEPROM (Fig.7a), so our test to read the magic number will fail. In this case, we will load the default settings values, including our recently calculated checksum byte, from `DefSettings` into both the EEPROM and `WrkSettings` (Fig.6b and Fig.7b).

OK, suppose we play with our *Victorian Display* and then power it off again without changing any of the settings. In this case, when we next power the system up again, the EEPROM will contain a copy of the default settings (Fig.6b and Fig.7c). As before, one of the first things we do is to read the byte from address 0x000 in the EEPROM to see if it matches the magic number byte in `DefSettings`. In this case, the test passes, so we copy the values stored in the EEPROM into `WrkSettings` (Fig.7d).

At this point, we would use our checksum function to verify that the data we've copied into our `WrkSettings` variable is good (we'll discuss how we do this in a moment). If this test fails, we know that the data in the EEPROM has become corrupted in some way, in which case we will have to mediate the situation (perhaps by regressing to Fig.7b). Alternatively, if the test passes, then we know we are ready to rock and roll.

Now, suppose that while our program is running, we decide to change some of the settings (Fig.7e). For example, we might decide to modify the time format from 12-hour to 24-hour. Similarly, we might decide to modify the date format from YYYY/MM/DD to DD/MM/YYYY. Also, we might decide to change the effects associated with the various modes.

Obviously, we are going to have to equip our program with the ability to allow the user to select new values for everything. Each time the user does change one of the values then: a) the new selection will be copied into the appropriate byte in both the EEPROM and `WrkSettings` and b) a new checksum value will be generated, and this too will be stored in both the EEPROM and `WrkSettings` (Fig.6c and Fig.7f).

Checksum Redux

The reason the checksum bytes in Fig.6 and Fig.7 are shown as '`????????`' is that we haven't yet decided what algorithm we are going to use. The simplest checksum algorithm is called the 'longitudinal parity check' (Fig.8a). This involves breaking the data into 'words' with a fixed number (n) of bits and then computing the exclusive or (XOR) of all those words, *excluding* the as-yet-unknown checksum value itself. In our case, of course, our words are 8-bits wide.

The easiest way to think about this is that we want to end up with an even number of 1s in each column. In the least-significant (bit 0) column we have six 1s (even), so the corresponding checksum bit will be 0. In the bit 1 column we have five 1s (odd), so the corresponding checksum bit will be 1. In the bit 2 column we have three 1s (odd), so – once again – the corresponding checksum bit will be 1. And so on for the rest of the bits, resulting in a checksum value of 01010110.

In order to check the integrity of the data once we've read it out of the EEPROM and loaded it into `WrkSettings`, we will use our checksum function to compute the exclusive or (XOR) of all the settings bytes in `WrkSettings`, including the checksum byte. If the result is *not* 0x00 (all zeros), then we know an error must have occurred.

Another simple technique is the 'sum complement' algorithm (Fig.8b). In this case, we treat the settings bytes as unsigned binary numbers and add them all together, *excluding* the as-yet-unknown checksum value itself. In this case, 01000010 + 00010000 + 00000001 + 00000010 + 00000011 + 00001111 + 00000111 + 00001001 + 00000101 = 01111100. If the result had been bigger than our word size of eight bits, we would have simply discarded any overflow bits. Next, we generate the twos complement of this value (swap all the 0s for 1s, swap all of the 1s for 0s, and add 1 to the result), which gives us a checksum value of 10000100.

When we wish to check the integrity of the data once we've read it out of the EEPROM and loaded it into `WrkSettings`, we add all the bytes, *including* the checksum byte, and discard any overflow bits. Once again, if the result is anything other than 0x00 (all zeros), then we know an error has occurred.

Earlier we noted that, 'A good checksum algorithm will generate a significantly different output value for even a small change made to the input data.' Is this true of the two algorithms we just discussed? I fear not. Try flipping a single bit in one of our settings bytes and see what effect it has on the output. In my next T*ips and Tricks* column, we will consider a much more cunning algorithm, but first...

Mission critical

While fighting your way through this column, you may have wondered why we need to use checksum values to verify the integrity of our EEPROM contents. 'Is this form of memory really that unreliable?' you may be asking yourself. Well, the answer is 'Yes' and 'No' (I bet you saw that coming). It's certainly true that the typical Arduino EEPROM has a specified life of only 100,000 write/erase cycles, but that would become an issue only if we did something silly like write to it every millisecond (how many times do we really expect a user to modify the settings for something like our *Victorian Display?*).

One problem is that writing data to the EEPROM takes a relatively long time in the scheme of things, not that you would notice anything yourself, of course, especially since we are writing only a handful of bytes in the case of our *Victorian Display*. Still and all, we have to worry what might happen if a brown-out (low-power) condition was to occur while writing to the EEPROM. Similarly, for a poorly timed user-initiated power-down or reset occurring while we were writing to the EEPROM. In such a case, the checksum will not match on read-back and we'll know to treat the data as junk. Contrariwise, if we simply assume the data is fine, we could end up trying to work with junk, which will almost invariably end in tears.

As an aside, one way to overcome data corruption caused by cell wearing or power failure or whatever is to sequentially write two (or more) identical sets of data. If one checksum is bad, we still have one good copy; if both checksums are bad, then – speaking in the engineer's vernacular – we're up a certain creek without motive power. Of course, we could also employ error-correcting strategies like using Hamming codes, which can detect up to two-bit errors or detect-and-correct one-bit errors, but this is outside the scope of these columns.

Finally, something else you might be thinking is, 'Do we really need to go to all this effort for the *Victorian Display*, which – after all – is only really a glorified clock, for goodness' sake?' Ah, well, that's a very good question. My reasoning is that what we learn here will prove useful later in life should we ever find ourselves in the position of having to design real-world mission-critical or safety-critical systems.

OK, that's all for this *Tips and Tricks*. As always, I welcome your comments, questions and suggestions.

Part 8: PIC18F Development Board

In our previous article, we completed the software required for our *PIC18F Development Board* to monitor the voltage rails on the battery and the solar panel of an industrial solar-powered air quality sensor. The details of the sensor itself are unimportant, but it was the development and testing of that device which triggered this series of articles. Not the design of the air quality sensor itself, rather the desire to monitor how well a solar panel could recharge the battery supplying the sensor, while the sensor operated continuously. The customer wanted the sensor to operate for over two years without servicing, which is reasonable, since replacing batteries on a remotely located industrial sensor can be inconvenient and time-consuming. The sensor drew on average 15mA at 5.0V, so the battery to achieve this requirement would need to have a capacity of close to a kWh – completely unrealistic for a low-cost product. Fortunately, this is an outdoor sensor, so we could consider adding a solar panel to charge a much lower capacity lithium ion (Li-ion) rechargeable battery. The question now became how to balance the size of the solar panel (big solar panels are expensive, and require large enclosures, which are expensive) versus the size of the Li-ion battery (bigger capacities are expensive, large and difficult to ship.)

Doing some basic calculations, an off-the-shelf 5.6Ah (20Wh) 3.7V Li-ion battery pack with a small 4×6-inch solar panel appeared to provide the balance required to provide continuous operation through night and day cycles. There is a 'gotcha' however – this calculation was based on datasheet analysis, and while that is normally appropriate for circuit designs, when it comes to solar panel performance datasheets cannot account for the weather. As these devices will be deployed in the UK, where clear skies are unpredictable and typically infrequent, we needed some real-life testing over an extended period and ideally covering the worst weather seasons to see if the solar panel would receive enough daylight to keep the system running. In such a system, monitoring the setup over months of operation cannot be done manually. An automated monitoring system is required, which is where this project came into being.

The author lives in an area with even cloudier weather than the UK – Ireland! Therefore, testing the setup there offered a degree of safety margin for the results and their applicability to the customer's UK market.

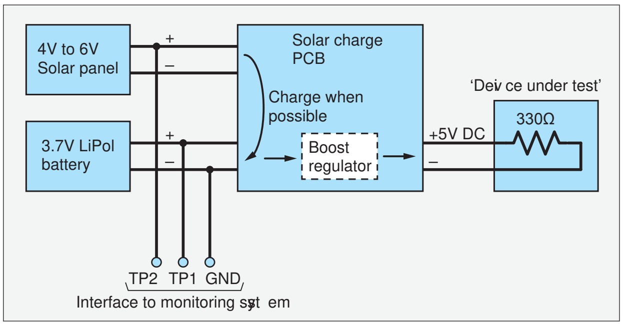


Fig.1. Block diagram of the 'Device Under Test'.

Creating the test system

We know the average current consumption of the sensor system – 15mA – so it was simple to devise a test system. We needed only a resistor that would draw 15mA at 5.0V. Ohm's law tells us a 330Ω resistor would suffice. Although the real device will have peaks of power consumption when transmitting data, and long periods in very-low-power sleep mode, the impact on the battery can be safely simulated by providing this fixed, *average* current consumption. The resistor will be dissipating just 50mW, so just about any resistor power rating would do.

Now let's turn to the circuit. We need our solar panel, the charger circuit with combined 5V regulator (an off-the-shelf PCB purchased online) and the battery. It's very simple, as shown in Fig.1. The fully integrated PCB is small, measuring just 63×33 mm. (For this customer application, we chose an off-the-shelf PCB because the quantities to be built are low, but the circuit is straight forward with an easily available IC, so an integrated design could be possible should a future cost reduction or size reduction be desired.)

Finally, we added the monitoring points for measuring the instantaneous solar panel output voltage and the battery output voltage, the latter tells us if the sensor circuit is being supplied with sufficient power: day and night.

With our 'Device Under Test' system configured, it's time to look at how to configure the *PIC18F Development Board* to monitor the system. For this, we build on the software and hardware setup developed in the previous two articles, where we buffer the incoming voltages from TP1 and TP2 to the processor's analogue-to-digital converter pins using the op amps configured as unity-gain buffers. This way, the potential divider circuit (necessary to

Fig.2. The solar charger PCB, available from DFRobot.

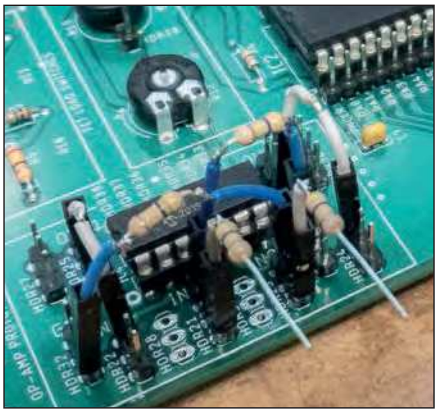


Fig.3. Wiring up the op amps / resistors.

reduce the voltages to below the microcontroller's 3.3V supply level) can be high values (here, 470kΩ) so as not to present an additional 'load' to the test device's battery. At 5V, this potential divider will draw just 5µA additional current. The average load of the 'Device Under Test' is 15mA, so it's clear the impact will be negligible. We attached the resistors to the op amps by cutting a pair of female-female jump wires, and soldering them to the resistors, as shown in Fig.3. This is a bit of a waste of two jump wires, but we wanted to avoid soldering to the pins. The choice is yours, but header pins are hard to re-use with jump wires once you've soldered to them.

While the resistor potential divider circuit has no impact on the current consumption of the 'Device Under Test', the current consumption of our test system is another matter. With Wi-Fi active it will draw approximately 60mA on average, so it does need to be powered by an independent mains power source to ensure that we do not interfere with the target sensor we are testing. This setup was going to be left running continuously for months, so we used a mains 9V DC power brick and a long cable to keep the system operational. For initial testing of the system setup however, we used a large 7.4V LiPol battery, which ran for several days and gave us the interesting results shown later in the article. The block diagram for this test system is shown in Fig.4.

Adding Wi-Fi

Having wired the system up as shown in Fig.1 and Fig.4, placed it in a nice waterproof transparent enclosure and installed the application developed in the previous article, the system meets our basic requirements, storing the readings into a text file on a MicroSD Media card. However, there are limitations with this solution. It's inconvenient having to remove the card to extract the data and even then, with the data copied to your computer, how do you visualise it? Not everyone has Microsoft Office installed on their computers and the knowledge to draw graphs in Excel.

The solution to this is to make use of the Wi-Fi interface integrated into the *Development Board*, a capability provided by a plug-in ESP-01 module. We've not made use of this yet, so this will be an ideal opportunity to test its capabilities. With the *Development Board* connected to the Internet via a Wi-Fi network, we not only gain easier access to our data, but we can also send the data automatically to the Internet, and specifically to one of the freely available IoT Dashboard services, where we can visualise our data as an interactive graph – updating in real time!

The ESP-01 module is a small PCB with very little to it – an ESP8266 IC, a Flash memory IC to hold the code, and a tiny integrated PCB trace antenna. The ESP8266 was released in 2014, and it made a significant impact – once documentation in English became available. It is a very capable processor with integrated Wi-Fi communication costing just a couple of pounds, or less. It does have very limited peripherals, but for IoT uses (reading a sensor connected by SPI or I²C and sending that data to the cloud) it met many requirements and was quickly adopted world-wide for the blossoming IoT market. Data throughput is not fast, at less than 1MB/s, so you won't be watching YouTube videos with this module (or could we? We'll have to think about that!) For IoT applications, however, where we are typically sending a few hundred bytes per minute, it's ideal.

Following the success of the ESP8266, Espressif launched the far more capable ESP32 processor in 2016, and their latest processor the ESP32-S3 was released in 2020. This latest IC offers a dual CPU chip running at 240MHz, 512KB of SRAM and both Wi-Fi and Bluetooth LE communications on chip. While it is great to see more complex Wi-Fi-enabled ICs becoming available, the 'old' ESP8266 is still a perfectly usable processor for IoT projects. Thankfully, the Far-East manufacturers agree, and simple modules that provide basic Wi-Fi connectivity are widely available, and incredibly cheap.

The ESP-01 is a generic module manufactured by many different vendors, some with slightly different features than others. Our intended use of the device is simple (just basic Wi-Fi communication to a Wi-Fi Access Point) so any generic version will do, as long as it has the same pin-out to match our *Development Board's* header pins. The firmware on these devices can be updated over the header pins, and our *Development Board* provides the physical connections from the header to GPIO on the PIC processor to do so if required.

The module comes pre-programmed with special Wi-Fi communications firmware, using a simple command set that is accessed over a 3.3V UART serial interface. The commands loosely follow the Hayes AT command set, so anyone familiar with old modems will feel at home. For the rest of us, the commands are simple enough to use, and if you connect an ESP-01 to a PC with a serial interface (such as a 3.3V USB-to-UART cable) you can type the commands using a serial interface terminal program such as Microsoft's Hyperterm (although we favour the free alternative, called 'Putty' – see: **www.putty.org**)

A reference to the command set can be found on the Internet. A quick search for 'ESP-01 AT Commands' yielded the following web page, which we used for our experimentation: **https://bit.ly/pe-nov21-at**

There are various versions of firmware supplied with the different clones, and the ESP-01 we received had old firmware on them (determined by running the command `AT+GMR`) Running older software does not seem to limit us, and we could if we wanted to perform a firmware update, but to avoid a complicated upgrade process we stuck with the old firmware. There are several interesting firmware updates that can be deployed, including full Arduino code

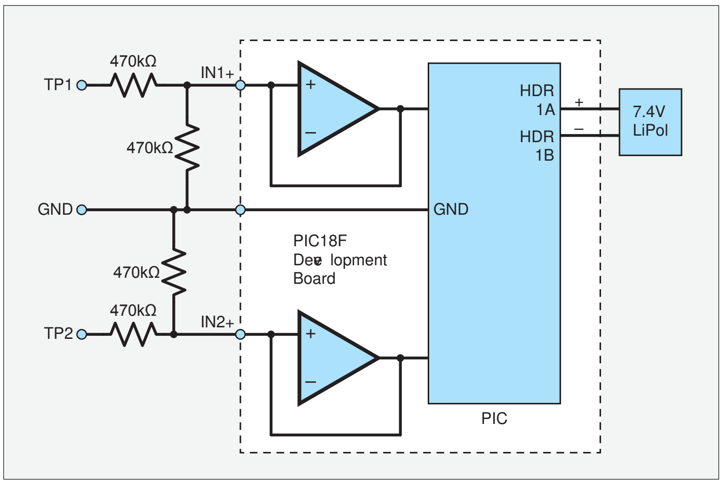


Fig.4. Test system block diagram.

GND	GPIO2	GPIO0	RX
TX	CH_PD	RESET	VCC

Fig.5. ESP-01 module.

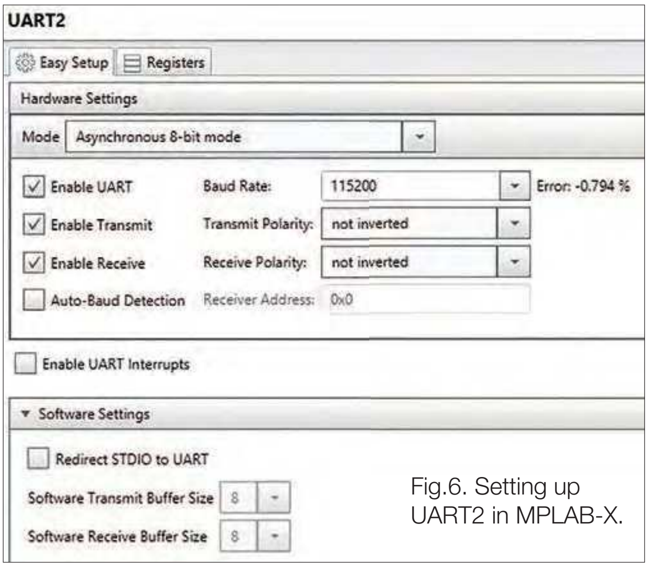


Fig.6. Setting up UART2 in MPLAB-X.

development compatibility, and we can cover doing this in a future article if there is sufficient interest. When we designed the *PIC18F Development Board* we anticipated needing to update the firmware on the ESP-01 and added the control signals to the header to be able to do it. We'd just need to write the software.

The ESP-01 is more than a simple Wi-Fi communications module; it's a fully functional, programmable microcontroller with a well-supported free development environment of its own. This begs the question, 'Why add another processor to our *Development Board*, why not just use the ESP-01 processor?' There are several good reasons, but the main one is a lack of simultaneously accessible peripherals. There are only a few pins available on the ESP-01 module, so it is not suitable for creating a generic development system with many peripherals available for use at the same time. In specific applications, however, it can be an ideal solution, hence its popularity over the years in both module and IC formats. The ESP-01 module runs at up to 80MHz, has UART, SPI, I²C and Wi-Fi peripherals, 96KB of on-chip RAM and an external 1MB of Flash memory for code – so a much higher specification than the PIC processor, albeit with fewer pins available.

Integrating the module

The module is shown in Fig.5. Note that it runs at 3.3V only, so the *Development Board* must be set to run at 3.3V too, by fitting JP1.

Before writing any software to talk to the module we need to bring the second UART peripheral into action to be able to talk to it. The module communicates at 115200 baud, eight data bits, no parity, and does not make use of RTS or CTS signalling. Nor does it use XON/XOFF flow control thankfully. This is a slow Wi-Fi interface, and with the kind of data exchanges we are going to be performing, flow control of transmitted and received messages is unnecessary.

We will continue using UART1 for our interface over the USB-to-UART link because this will be very useful for seeing debug messages from our software and response messages from the ESP-01 module. Also, we have two UARTs, so why not use them!

First, we open the project code, named **passthrough** (unzipped from **passthrough.zip**, available for download from the November 2021 page of the *PE* website) and load it into MPLAB-X.

Open up the MCC dialogue panel in MPLAB-X again, and add the UART2 peripheral to our project. The module communicates at 115200 baud; and since our data exchange rate will be slow (we will send voltage messages every minute or so) we do not need to use more complicated-to-understand interrupt-based communication. On this basis, we configure the UART as shown in Fig.6.

Having configured the UART peripheral, we can now move on and select which pins connect to it by clicking on the Pin Module entry in the Project Resources list. After clicking on it, the lower window should change to the Pin Manager view. Scroll down until the UART2 row is visible and use the left mouse button to select PORT B pins 4 and 5 to connect to TX2 and RX2, as shown in Fig.7. Confirm that the RB4 and RB5 rows shown above it are correctly set too, as shown in Fig.8.

Next, click the Generate button to action these requests. MCC will now create new UART2 driver files in our project and update our existing code to make reference to them.

We are now ready to write some code. Let's go take a look in the header file for the new UART2 driver we've created, to see which functions will be useful, and how to use them.

Open the file **uart2.h**, by clicking on the filename in the left hand 'Files' tab. You will find it in the **mcc_generated_files** sub-directory. Unsurprisingly, it looks identical to that of **uart1.h**. The peripherals are identical, the only difference is that we have mapped them to different pins, and perhaps set the baud rate differently. This means that the code generated for the `UART2_Initialize()` function will be very different. Not that we are interested; that code has been generated for us and will be automatically called on processor reset without us needing to write that line of code – MCC added it for us. In fact, there are only three functions that are of interest to us:

```
void UART2_Write(uint8_t txData);
```

This function will send a single character out over the UART, when it is free to do so.

```
uint8_t UART2_Read(void);
```

Returns the value of the last character received over the UART.

```
bool UART2_is_rx_ready(void);
```

This will return true if there is data to read from the receive buffer.

That's it; that is all we need from the driver. It would be useful to have a 'write a string of characters' function, but this is trivial to implement. The code looks like this:

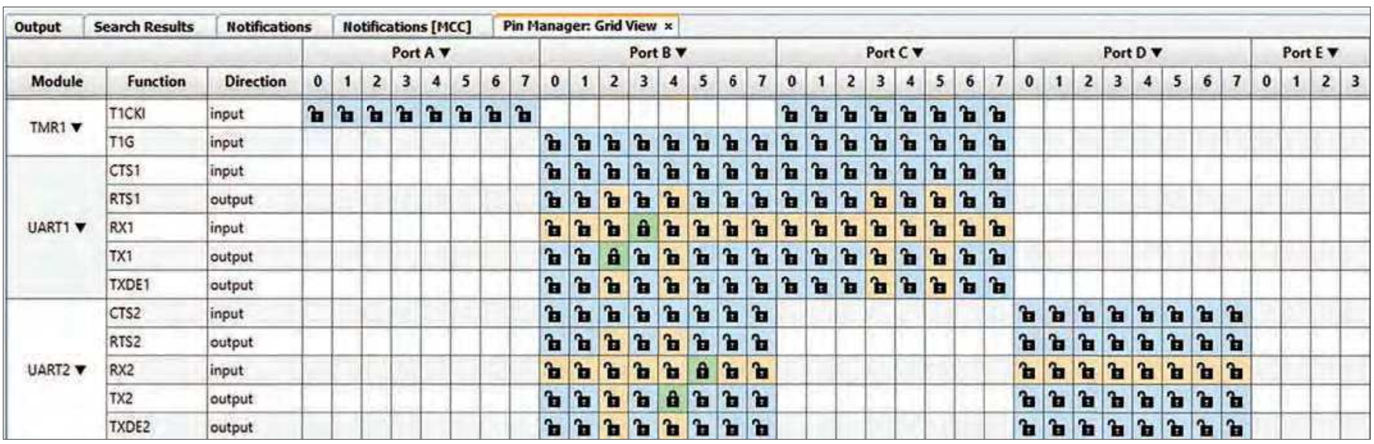


Fig.7. Completing the setup of UART2.

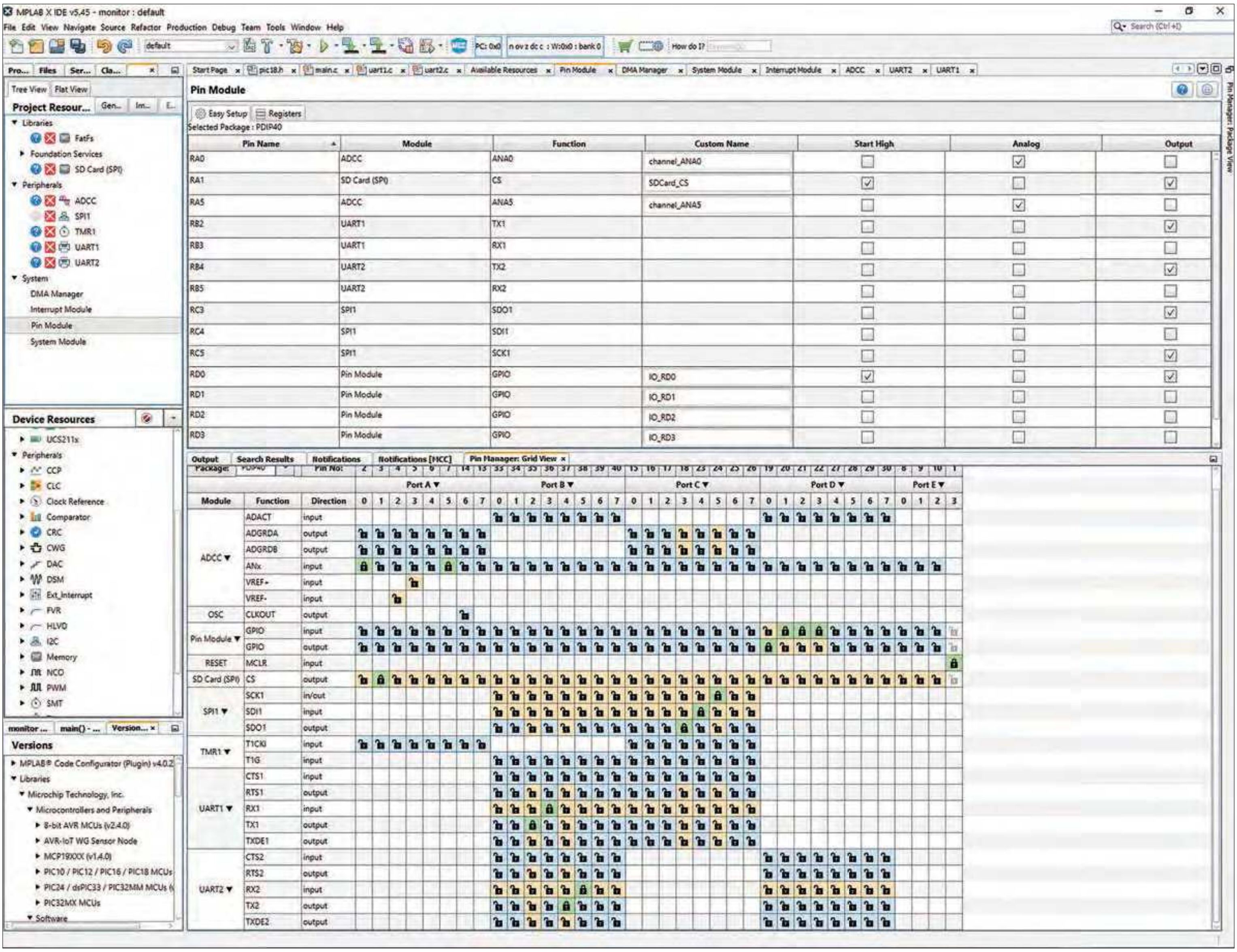


Fig.8. Connecting the GPIO pins to UART2.

```
void uart2_putstr(uint8_t *str)
{
  uint8_t val;

  while (*str) {
    UART2_WriteUART2_Write(*str++);
    __delay_ms(5);
    if (UART2_is_rx_ready()) {
      val = UART2_Read();
      UART1_Write(val);
    }
  }
}
```

The short delay is introduced to allow the ESP-01 module time to process the data we sent to it. We implemented this function in our own **main.c** file. (Do not edit code automatically generated by MCC – it may become overwritten should you use MCC to make other configuration changes.)

Now we are ready to start talking to the ESP-01 module. Let's begin by writing a very simple program that allows you to use a terminal program such as Putty to communicate directly with the ESP-01 module. We refer to this kind of program as a 'passthrough' application. The code can be seen in Listing 1. Its operation is simple, and it's a piece of code we have used many times in the past. If a character is received from the module, then retransmit it over the USB UART interface. If a character is received from the USB UART interface, then retransmit it to the module. There is one additional step, implemented by these four lines of code:

```
if (val == '\r') {
  do {} while (!UART2_is_tx_ready());
  UART2_Write('\n');
}
```

`\r` is a special character, known as 'Carriage Return'. This is the character the Putty program will send when you press the Enter key on the keyboard, to send a command. The module expects two characters: Carriage Return and then 'Line Feed', represented in the C programming language by `\n`. So, if we detect a Carriage Return character, we send it and then immediately send a Line Feed character. We run this sequence of code in a loop to provide a simple yet effective communication link between our PC and the module.

With a terminal programming running on the PC (Putty in our case) we can type `AT` followed by the Enter key, and get the response `OK` back from the module.

If you see this, all is good. If not, check that the terminal program is connected to the virtual serial port created when you plugged in the USB cable between the *Development Board* and the PC, and check the code you typed.

Fig.9. A complicated Thingsboard dashboard displaying a bus fleet monitoring system.

Listing 1: Talking to the ESP-01.

```
void main(void)
{
   uint8_t val;

   // Initialize the device
   SYSTEM_Initialize();
   do {
      // UART1 is USB serial cable, 115200 baud
      if (UART1_is_rx_ready()) {
         val = UART1_Read();
         do {} while (!UART2_is_tx_ready());
         UART2_Write(val);
         if (val == '\r') {
            do {} while (!UART2_is_tx_ready());
            UART2_Write('\n');
         }
      }

      // UART2 is ESP-01 module
      if (UART2_is_rx_ready()) {
         val = UART2_Read();
         UART1_Write(val);
      }
   } while (1);
}
```

You can now refer to the list of AT commands available and experiment; it's not possible to damage the module by sending incorrect commands.

Having explored the module's functionality, we found that integrating the module into our project consists of two tasks:

1. Configure the module to connect to your Wi-Fi access point
2. Use commands to connect to a website, post data and receive responses.

The configuration setup in step 1 gets written into Flash memory, and so needs to be done only once. This is ideal because one of the commands requires you to enter the password for your Wi-Fi access, and that is not something we would want visible in source code. Just a few commands, entered while the *Development Board* is running the 'passthrough' program, completes step 1).

```
ATE0
AT+CWMODE=3
AT+CIPMUX=0
AT+CIPMODE=0
AT+CWJAP="HomeWifi","WiFiPassword"
```

If these were entered correctly, the module should return the message:

```
WIFI CONNECTED
WIFI GOT IP
```

We can now move on to sending actual data – but, to where?

Internet dashboards

Internet-based dashboard services are websites that provide an online database to store telemetry data from you Internet-connected devices, and which display that data on graphs, updated in real time, with the ability to manipulate and filter the data being displayed on those graphs. The graphs are displayed on a webpage called a dashboard, which you configure yourself, selecting from a range of predefined graph types, and you can also select what data from which devices to display on those graphs. You can sign up with a monthly subscription to have your own private system hosted by a third party, but some companies offer free hosting for hobbyists. Thingsboard is one such company, and we will use their free service, available at: **https://demo.thingsboard.io**

The Thingsboard service can enable you to create very complex dashboards, an example of a bus fleet monitoring system can be seen in Fig.9. To build dashboards like this requires some skill and experience, but a very simple dashboard displaying two voltage levels from our single device will be simple to configure. Creating an account, configuring a virtual device and creating a dashboard is better described by a video, see here: **https://youtu.be/gX1DMkdYgfg**

The key point is that when you create a 'device' on Thingsboard you are provided with a unique 'Access Token' for that device. This is effectively a password for writing data into the Cloud-based database.

To write data, your device must connect to the Thingsboard website at **demo.thingsboard.io**, and then send a 'post' command, which is a text string providing the Access Token password and a JSON-formatted string containing your

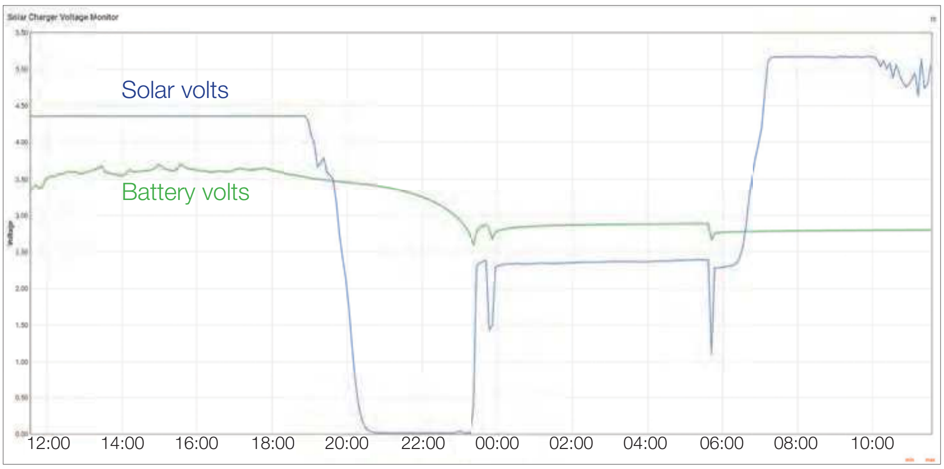


Fig.10. Example data over 24-hours, on a cloudy day for the Solar Voltage Charger Monitor.

Fig.11. The test system, in a real-world location – my Dublin garden!

variable name and a value. This is performed with just three ESP-01 commands, AT+CIPSTART to connect to the website, AT+CIPSEND to tell the ESP-01 how many bytes you are going to send to the website, and then the string of bytes to send, starting with the text POST. That's it, and you can repeat this in an infinite loop to complete the monitoring application, as seen in Listing 2. **solarsensor,** the full code for this application, can also be download from the *PE* website. Unzip **solarsensor.zip**, open the project in MPLAB-X and then open the **main.c** file. You can add your unique access token on the two lines near the end of the file, replace ACCESS-TOKEN with your unique string.

Having run the system outdoors for a weekend, the results were fascinating. You can see a 24-hour period covering an overcast Irish September day in Fig.10, and the test setup in Fig.11. In this graph we can see that the solar panel appears to have four different stable output states, and with some thought it become clear what is going on. At first, under daylight conditions, the system is running, the solar panel under load is charging the battery and supplying power to the device under test. After sunset the solar panel is attempting to do the same, but with no sunlight, the output is clamped to zero volts. At around midnight the battery expires and the device under test turns off. Now, with no current drain, the solar panel can float up to a few volts. At sunrise, the device is still turned off, so the solar panel can rise even higher, to its open-circuit voltage of around 5.2V. Finally, at 10AM the battery has charged sufficiently to turn the device on, presenting a larger load to the solar panel, causing its output voltage to drop again. These are valuable insights that would be impossible to gain by looking at a list of numbers. And the conclusion – don't start your system with a flat battery!

Listing. 2 Reading voltages and sending them to Thingsboard.

```
ADCC_StartConversion(channel_ANA0);
   while(!ADCC_IsConversionDone());
   solarPanel_volts_adc = ADCC_GetConversionResult();
   solarPanel_volts = (double)solarPanel_volts_adc;
   // Convert ADC value to volts, compensate for divide by 2
   solarPanel_volts = (solarPanel_volts * 3.3 * 2.0) / 4095.0;

   ADCC_StartConversion(channel_ANA5);
   while(!ADCC_IsConversionDone());
   battery_volts_adc = ADCC_GetConversionResult();
   battery_volts = (double)battery_volts_adc;
   // Convert ADC value to volts, compensate for divide by 2
   battery_volts = (battery_volts * 3.3 * 2.0) / 4095.0;

   parse_response(2000);

   uart2_putstr("AT+CIPSTART=\"TCP\",\"demo.thingsboard.io\",80\r\n");
   parse_response(1000);

   uart2_putstr("AT+CIPSEND=180\r\n"); parse_response(500);

   sprintf(esp01_buff, "POST /api/v1/access-token-string/telemetry
HTTP/1.1\r\nHost: demo.thingsboard.io\r\nAccept:

application/json\r\nContent-Length: 23\r\n

Content-Type: application/json\r\n\r\n{\"

SolarVolts\": %1.3f}\r\n", solarPanel_volts);

   uart2_putstr(esp01_buff); parse_response(1000);

   uart2_putstr("AT+CIPSEND=182\r\n"); parse_response(500);

   sprintf(esp01_buff, "POST /api/v1/access-token-string/telemetry
HTTP/1.1

\r\nHost: demo.thingsboard.io\r\nAccept:

application/json\r\nContent-Length: 25\r\n

Content-Type: application/json\r\n\r\n{\"

BatteryVolts\": %1.3f}\r\n", battery_volts);

   uart2_putstr(esp01_buff);
   parse_response(1000);

   __delay_ms(10000);
```

Both Listings are available for download from the November 2021 page of the *PE* website.

This has been a challenging stage in the article series as there were many tricky issues to solve when working out how to post data in the correct format to the Internet. It involved a lot of trial and error, which thankfully you will not have to endure. With the addition of a Wi-Fi interface and the use of freely available online IoT platforms, we no longer need to use the SD Media card for storage of data locally. The capability is there however, and can work in parallel with Internet communications if desired. This is one of the benefits of using a PIC microcontroller with a large number of GPIO pins.

In our next article we will add an LCD to the project, giving local status information and showing (again) how easy it is to add complex peripherals to the *Development Board*. We will also talk further on formatting and sending data to the Thingsboard platform and review other freely available Internet dashboard services.

PIC n' Mix files

The programming files discussed in this article are available for download from the *PE* website.

Practical Electronics PCB SERVICE

NOVEMBER 2021

PROJECT	CODE	PRICE
Dual Battery Lifesaver	11111202	6.95

OCTOBER 2021

PROJECT	CODE	PRICE
Mini Wi-Fi LCD BackPack	24106201	8.95

SEPTEMBER 2021

PROJECT	CODE	PRICE
USB SuperCodec PCB	01106201	14.95
Audio DDS Oscillator PCB	01110201	5.95
Audio DDS Oscillator rotary encoder	01110201-ENC	6.95
Programming Adaptor Board for Audio DDS Oscillator	01110202	5.95
High-power Ultrasonic Cleaner main PCB	04105201	£14.95 (both)
High-power Ultrasonic Cleaner front-panel PCB	04105202	
Night Keeper Lighthouse PCB	08110201	5.95

AUGUST 2021

PROJECT	CODE	PRICE
Ol' Timer PCB	19104201	£11.95
Ol' Timer 8x8 RGB LED module using WS2812B	19104201-88	£8.95
Ol' Timer set of acrylic case pieces and spacer	19104201-ACR	£8.75
Ol' Timer DS3231 RTC IC wide SOIC-16	19104201-RTC	£5.95
Wideband Digital RF Power Meter	04106201	£9.75
Switchmode 78xx regulators ***(PACK of 5!)***	18105201	£7.95
Cool Beans SMAD display	CB-AUG21	£11.95

JULY 2021

PROJECT	CODE	PRICE
ATtiny816 Breakout / Dev Board with Capacitive Touch	24110181	£9.75
IR Remote Control Assistant (Jaycar version)	15005201	£8.95
IR Remote Control Assistant (Altronics version)	15005202	£8.95
PIC18F Development Board	PNM-JUL21	12.95
Microphone Preamplifier	AO-JUL21	11.95

JUNE 2021

PROJECT	CODE	PRICE
Roadies' Test Signal Generator (surface-mount version)	01005201	£8.95
Roadies' Test Signal Generator (through-hole version)	01005202	£9.95
Touchscreen Wide-range RCL Box (Resistor module)	04104201	£18.95 (both)
Touchscreen Wide-range RCL Box (Ind/Cap module)	04104202	
KickStart Part 3 – Gyrator-based Audio Filter	KS3-2021	£7.95

MAY 2021

PROJECT	CODE	PRICE
7-Band Equaliser (Mono)	01104201	£8.95
7-Band Equaliser (Stereo)	01104202	£10.95
Car Altimeter	05105201	£7.95

APRIL 2021

PROJECT	CODE	PRICE
Reflow Oven – DSP Active Crossover (CPU)	01106193	£19.95 (all three)
Reflow Oven – DSP Active Crossover (Front panel)	01106195	
Reflow Oven – DSP Active Crossover (LCD)	01106196	
Frequency Reference Signal Distributor	CSE200103	£8.95

MARCH 2021

PROJECT	CODE	PRICE
Nutube Guitar Effects Pedal	01102201	£12.95
Programmable Thermal Regulator (Peltier Interface)	21109181	£18.95 (both)
Programmable Thermal Regulator (Peltier Driver)	21109182	
Tunable HF Preamp	CSE190502	£8.95

FEBRUARY 2021

PROJECT	CODE	PRICE
4G Remote Monitoring	27111191	£9.95

JANUARY 2021

PROJECT	CODE	PRICE
Nutube Valve Preamplifier	01112191	£12.95
Arduino DCC Controller	09207181	£10.95

DECEMBER 2020

PROJECT	CODE	PRICE
Pseudo-Random Sequence Generator	16106191	£7.95
Clever Charger	14107191	£11.95
PE Theremin Amplifier	AO-1220-01	£7.95

NOVEMBER 2020

PROJECT	CODE	PRICE
LED Christmas Tree (1 off)	16107181-1	£6.95
LED Christmas Tree (4 off)	16107181-2	£14.95
LED Christmas Tree (12 off)	16107181-3	£24.95
LED Christmas Tree (20 off)	16107181-4	£34.95
USB/SPI Interface Board	16107182	£5.95
45V/8A Power Supply PCB *plus* acrylic spacer	18111181	£14.95
45V/8A Power Supply front panel five-way display bezel	18111181-BZ	£3.95
Five-way LCD Panel Meter/Display	18111182	£7.95

OCTOBER 2020

PROJECT	CODE	PRICE
Digital Audio Millivoltmeter	04108191	£8.95
Precision Signal Amplifier	04107191	£6.95

SEPTEMBER 2020

PROJECT	CODE	PRICE
PE Theremin PSU	AO-0920-01	£5.95
PE Theremin PSU transformer	AO-0920-02	£7.95
Micromite Explore-28	07108191	£5.95
Ultrabrite LED Driver	16109191	£5.95

AUGUST 2020

PROJECT	CODE	PRICE
Micromite LCD BackPack V3	07106191	£9.95
Steering Wheel Audio Button to Infrared Adaptor	05105191	£7.95

JULY 2020

PROJECT	CODE	PRICE
AM/FM/CW Scanning HF/VHF RF Signal Generator	04106191	£11.95
Speech Synthesiser with the Raspberry Pi Zero	01106191	£5.95
PE Mini-organ PCB	AO-0720-01	£14.95
PE Mini-organ selected parts	AO-0720-02	£8.95
High-current Solid-state 12V Battery Isolator – control	05106191	£6.95
High-current Solid-state 12V Battery Isolator FET (2oz)	05106192	£9.95

JUNE 2020

PROJECT	CODE	PRICE
Arduino breakout board – 3.5-inch LCD Display	24111181	£6.95
Six-input Audio Selector main board	01110191	£10.95 (both)
Six-input Audio Selector switch panel board	01110192	

MAY 2020

PROJECT	CODE	PRICE
Ultra-low-distortion Preamplifier Input Selector	01111112	£11.25 (both)
Ultra-low-distortion Preamplifier pushbutton Input Selector	01111113	
Universal Regulator	18103111	£7.95
433MHz Wireless Data Repeater	15004191	£8.50
Bridge-mode Adaptor for Amplifier	01105191	£7.95
iCEstick VGA Terminal	02103191	£4.95
Analogue noise with tilt control	AO-0520-01	£7.95
Audio Spectrum Analyser	PM-0520-01	£8.95

APRIL 2020

PROJECT	CODE	PRICE
Flip-dot Display black coil board	19111181	£14.95 (all four)
Flip-dot Display black pixels	19111182	
Flip-dot Display black frame	19111183	
Flip-dot Display green driver board	19111184	

MARCH 2020

PROJECT	CODE	PRICE
Diode Curve Plotter	04112181	£10.95
Steam Train Whistle / Diesel Horn Sound Generator	09106181	£8.50
Universal Passive Crossover (one off)	UPC0320	£12.50

FEBRUARY 2020

PROJECT	CODE	PRICE
Motion-Sensing 12V Power Switch	05102191	£5.95
USB Keyboard / Mouse Adaptor	24311181	£8.50
DSP Active Crossover (ADC)	01106191	£29.95 (all six)
DSP Active Crossover (DAC) ×2	01106192	
DSP Active Crossover (CPU)	01106193	
DSP Active Crossover (Power/routing)	01106194	
DSP Active Crossover (Front panel)	01106195	
DSP Active Crossover (LCD)	01106196	

PCBs for most recent *PE*/*EPE* constructional projects are available. From the July 2013 issue onwards, PCBs with eight-digit codes have silk screen overlays and, where applicable, are double-sided, have plated-through holes, and solder mask. They are similar to photos in the project articles. Earlier PCBs are likely to be more basic and may not include silk screen overlay, be single-sided, lack plated-through holes and solder mask.

Always check price and availability in the latest issue or online. A large number of older boards are listed for ordering on our website.

In most cases we do not supply kits or components for our projects. For older projects it is important to check the availability of all components before purchasing PCBs.

Back issues of articles are available – see Back Issues page for details.

Double-sided | plated-through holes | solder mask

PROJECT	CODE	PRICE
JANUARY 2020		
Isolated Serial Link	24107181	£8.50
DECEMBER 2019		
Extremely Sensitive Magnetometer	04101011	£16.75
Four-channel High-current DC Fan and Pump Controller	05108181	£8.75
Useless Box	08111181	£11.50
NOVEMBER 2019		
Tinnitus & Insomnia Killer (Jaycar case – see text)	01110181	£8.75
Tinnitus & Insomnia Killer (Altronics case – see text)	01110182	£8.75
OCTOBER 2019		
Programmable GPS-synced Frequency Reference	04107181	£11.50
Digital Command Control Programmer for Decoders	09107181	£9.95
Opto-isolated Mains Relay (main board)	10107181	£11.50
Opto-isolated Mains Relay (2 × terminal extension board)	10107182	
AUGUST 2019		
Brainwave Monitor	25108181	£12.90
Super Digital Sound Effects Module	01107181	£6.95
Watchdog Alarm	03107181	£8.00
PE Theremin (three boards: pitch, volume, VCA)	PETX0819	£19.50
PE Theremin component pack (see p.56, August 2019)	PETY0819	£15.00
JULY 2019		
Full-wave 10A Universal Motor Speed Controller	10102181	£12.90
Recurring Event Reminder	19107181	£8.00
Temperature Switch Mk2	05105181	£10.45
JUNE 2019		
Arduino-based LC Meter	04106181	£8.00
USB Flexitimer	19106181	£10.45
MAY 2019		
2× 12V Battery Balancer	14106181	£5.60
Deluxe Frequency Switch	05104181	£10.45
USB Port Protector	07105181	£5.60
APRIL 2019		
Heater Controller	10104181	£14.00
MARCH 2019		
10-LED Bargraph Main Board	04101181	£11.25
+Processing Board	04101182	£8.60
FEBRUARY 2019		
1.5kW Induction Motor Speed Controller	10105122	£35.00
NOVEMBER 2018		
Super-7 AM Radio Receiver	06111171	£27.50
OCTOBER 2018		
6GHz+ Touchscreen Frequency Counter	04110171	£12.88
Two 230VAC MainsTimers	10108161	£12.88
	10108162	
SEPTEMBER 2018		
3-Way Active Crossover	01108171	£22.60
Ultra-low-voltage Mini LED Flasher	16110161	£5.60
AUGUST 2018		
Universal Temperature Alarm	03105161	£7.05
Power Supply For Battery-Operated Valve Radios	18108171	£27.50
	18108172	
	18108173	
	18108174	

PROJECT	CODE	PRICE
JULY 2018		
Touchscreen Appliance Energy Meter – Part 1	04116061	£17.75
Automotive Sensor Modifier	05111161	£12.88
JUNE 2018		
High Performance 10-Octave Stereo Graphic Equaliser	01105171	£15.30
MAY 2018		
High Performance RF Prescaler	04112162	£10.45
Micromite BackPack V2	07104171	£8.45
Microbridge	24104171	£5.60
APRIL 2018		
Spring Reverberation Unit	01104171	£13.95
DDS Sig Gen Lid	Black	£8.05
DDS Sig Gen Lid	Blue	£7.05
DDS Sig Gen Lid	Clear	£8.05
MARCH 2018		
Stationmaster Main Board	09103171	£17.75
+ Controller Board	09103172	
SC200 Amplifier Module – Power Supply	01109111	£16.45

For the many pre-2018 PCBs that we stock please see the PE website: www.electronpublishing.com

PE/EPE PCB SERVICE

Order Code **Project** **Quantity** **Price**

....

....

....

....

....

Name ....

Address ....

....

Tel ....

Email ....

I enclose payment of £ .... (cheque/PO in **£ sterling only**)

payable to: **Practical Electronics**

Card No ....

Valid From ....Expiry Date ....

Card Security No ....

You can also order PCBs by phone, email or via the shop on our website: www.electronpublishing.com

No need to cut your issue – a copy of this form is just as good!

All prices include VAT and UK p&p. Add £4 per *project* for post to Europe; £5 per *project* outside Europe.

Orders and payment should be sent to:

Practical Electronics, Electron Publishing Ltd
113 Lynwood Drive, Merley, Wimborne, Dorset BH21 1UU
Tel 01202 880299 Email: shop@electronpublishing.com
On-line Shop: www.epemag.com

Cheques should be made payable to 'Practical Electronics' **(Payment in £ sterling only)**.

NOTE: Most boards are in stock and sent within seven days of receipt of order, please allow up to 28 days delivery if we need to restock.

CLASSIFIED ADVERTISING

Practical Electronics

If you want your advertisements to be seen by the largest readership at the most economical price then our classified page offers excellent value. The rate for semi-display space is £10 (+VAT) per centimetre high, with a minimum height of 2·5cm. All semi-display adverts have a width of 5.5cm. The prepaid rate for classified adverts is 40p (+VAT) per word (minimum 12 words).

Cheques are made payable to 'Practical Electronics'. **VAT must be added**. Advertisements with remittance should be sent to: Practical Electronics, 113 Lynwood Drive, Wimborne, Dorset, BH21 1UU.

Tel 07973518682 Email: pe@electronpublishing.com

For rates and further information on display and classified advertising please contact our Advertisement Manager, Matt Pulzer – see below.

Practical Electronics reaches more UK readers than any other UK monthly hobby electronics magazine. Our sales figures prove it.

We have been the leading monthly magazine in this market for the last twenty-seven years.

MISCELLANEOUS

VALVES AND ALLIED COMPONENTS IN STOCK. Phone for free list. Valves, books and magazines wanted. Geoff Davies (Radio), tel. 01788 574774.

PIC DEVELOPMENT KITS, DTMF kits and modules, CTCSS Encoder and Decoder/Display kits. Visit www.cstech.co.uk

ADVERTISING INDEX

Advertisement offices

Matt Pulzer

Electron Publishing Ltd
1 Buckingham Road
Brighton
East Sussex BN1 3RA

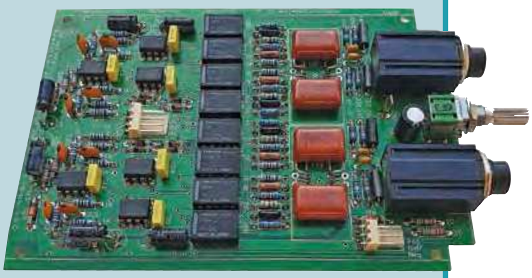
Tel 07973 518682

Email pe@electronpublishing.com

Web www.electronpublishing.com

For editorial contact details see page 7.

Next Month – in the December issue



Digital AM/FM/SW Receiver

This project uses the BK1198 digital radio chip, which is cheap and readily available, and requires only a handful of discrete components to work. The resulting radio covers the AM and FM broadcast bands, plus shortwave from 2.7 to 22MHz.

Mini Digital Volt/Amp Panel Meters

There are many low-cost digital panel meters available which can display voltage and current at the same time. Quite a few have popped up on the market in the last year or so. So let's take a look at some of the more popular models, see what's inside them and whether they're easy to use.

Balanced Input & Attenuator for the USB SuperCodec – Part 1

This compact balanced input attenuator is designed to fit into the same instrument case as the USB *SuperCodec*. It provides four attenuation settings of 0dB, –10dB, –20dB and –40dB and has performance to match the superlative *SuperCodec*. Together, they form a potent recording and/or measurement system.

KickStart – Part 6

Next month, *KickStart* will help you get started with direct digital synthesis (DDS). We've provided some practical circuit arrangements and sample code based on the popular and very inexpensive Analog Devices AD9833 DDS chip. These versatile devices can form the basis of projects ranging from simple fixed frequency sources to programmable waveform generators and FSK/PSK generators.

PLUS!

All your favourite regular columns from *Audio Out, Cool Beans and Circuit Surgery*, to *Make it with Micromite* and *Net Work*.

On sale 4 November 2021

Content may be subject to change

NEW subscriptions hotline!

Practical Electronics

We have changed the way we sell and renew subscriptions. We now use 'Select Publisher Services' for all print subscriptions – to start a new subscription or renew an existing one you have three choices:

1. Call our NEW print subscription hotline: 01202 087631, or email: pesubs@selectps.com
2. Visit our shop at: www.electronpublishing.com
3. Send a cheque (payable to: 'Practical Electronics') with your details to:

   Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Remember, we print the date of the last issue of your current subscription in a box on the address sheet that comes with your copy.

Digital subscribers, please call 01202 880299 or visit: www.electronpublishing.com

Published on approximately the first Thursday of each month by Electron Publishing Limited, 1 Buckingham Road, Brighton, East Sussex BN1 3RA. Printed in England by Acorn Web Offset Ltd., Normanton WF6 1TW. Distributed by Seymour, 86 Newman St., London W1T 3EX. Subscriptions UK: £26.99 (6 months); £49.85 (12 months); £94.99 (2 years). EUROPE: airmail service, £30.99 (6 months); £57.99 (12 months); £109.99 (2 years). REST OF THE WORLD: airmail service, £37.99 (6 months); £70.99 (12 months); £135.99 (2 years). Payments payable to 'Practical Electronics', Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom. Email: pesubs@selectps.com. PRACTICAL ELECTRONICS is sold subject to the following conditions, namely that it shall not, without the written consent of the Publishers first having been given, be lent, resold, hired out or otherwise disposed of by way of Trade at more than the recommended selling price shown on the cover, and that it shall not be lent, resold, hired out or otherwise disposed of in a mutilated condition or in any unauthorised cover by way of Trade or affixed to or as part of any publication or advertising, literary or pictorial matter whatsoever.

PEAK® electronic design ltd

15% discount!

Use code *PE15* during checkout

limited time only!

ZEN50 Zener Diode Analyser (inc. LEDs, TVSs etc)

Popular product!

The ***Atlas ZEN*** (model ZEN50) is ideal for testing Zeners, VDRs, transient suppressors, LED strings and many other components.

- **Measure Zener Voltage (from 0.00 up to 50.00V!)**
- **Measure Slope Resistance.**
- **Selectable test current: 2mA, 5mA, 10mA and 15mA.**
- **Very low duty cycle to minimise temperature rise.**
- **Continuous measurements.**
- **Single AAA battery (included) with very long battery life.**
- **Gold plated croc clips included.**
- **Can measure forward voltage of LEDs and LED strings too.**

~~£46.50~~ **£39.53**
~~£38.75+VAT~~ **with discount!**

LCR45 LCR and Impedance Meter with Auto and Manual modes

Advanced firmware!

The powerful LCR45 meter that not only identifies and measures your passive components (Inductors, Capacitors and Resistors) but also measures complex impedance, magnitude of impedance with phase and admittance too! Auto and Manual test modes allow you to specify the test frequency and component type.

- **Continuous fluid measurements.**
- **Improved measurement resolution: (<0.2µH, <0.2pF).**
- **Test frequencies: DC, 1kHz, 15kHz, 200kHz.**
- **Measure the true impedance of speakers and more.**
- **Great for hobbyists and professionals.**

~~£91.50~~ **£77.78**
~~£76.25+VAT~~ **with discount!**

DCA55 Semiconductor Analyser - Identify your semi's

Now with backlit display and AAA battery!

Connect any way round to identify the type of component and the pinout! Also measures many parameters including transistor gain, base-emitter voltages, MOSFET thresholds, LED voltages etc. Complete with a comprehensive illustrated user guide. Includes an Alkaline battery so you're ready to go straight away.

- **Transistors (including NPN/PNP, darlingtons, Si & Ge).**
- **Measure hFE, V_{BE} and leakage.**
- **Diodes and LEDs. Measure V_F.**
- **MOSFETs. Measure V_{GS}(th).**
- **Gold plated hook probes.**
- **Long battery life.**
- **Free technical support for life.**
- **Comprehensive instruction book.**
- **2 year warranty.**

~~£52.50~~ **£44.63**
~~£43.75+VAT~~ **with discount!**

ESR70 Equivalent Series Resistance and Capacitance Meter

Now with backlit display and AAA battery!

The new improved ESR70 is great for identifying capacitors (especially electrolytics) that are suffering from high ESR. ESR is a great indicator of capacitor condition, especially in applications demanding high ripple currents such as reservoir capacitors, SMPSUs, motor capacitors etc. Includes Alkaline AAA battery and a very comprehensive user guide. The ESR70 will even perform a controlled discharge of your caps!

~~£84.00~~ **£71.40**
~~£70.00+VAT~~ **with discount!**

It's only possible to show summary specifications here. Please ask if you'd like detailed data. Further information is also available on our website. Product price refunded if you're not happy.

www.peakelec.co.uk
sales@peakelec.co.uk

Atlas House, 2 Kiln Lane, Harpur Hill Business Park
Buxton, Derbyshire, SK17 9JL, UK. Tel: 01298 70012