Tiny LED Christmas Tree

Here's a brilliant little – and cheap – project for Christmas. It's a mini Christmas tree with 12 multi-colour LEDs which flash in sequence. It runs off a button cell which can last for weeks or even months. You can use whatever colour LEDs you want, or even mix them. You could build several of these, or even dozens, and arrange them around your tree (or anywhere else) for a spectacular light show!

by Tim Blythman

Our huge Stackable LED Christmas Tree from last year was a big hit, at least in part because you could use it to build a vast tree, up to a metre high – or even taller.

This project is at the other end of the spectrum; you can build trees so tiny that you could wear them as a badge, attach them to Christmas presents . . . or even use them to decorate a larger Christmas tree. You could even hang them on the giant LED tree (you could call it "Tree-ception").

PCBs and kits for last year's expandable tree (November 2019; **siliconchip.com.au/Article/11297**) are still available. So if you want a big, illuminated tree, go for it. But if you want to try something a little different, read on.

This idea came about when my wife said she wanted to create some decorative baubles for the festive season. They had to be small to be practical, and as I was involved in the design, naturally they would need flashing lights.

Thus was born the idea of the Tiny LED Christmas Tree.

The electronics are not extraordinary, except perhaps in their simplicity. An 8-pin microcontroller running from a single lithium cell drives 12 LEDs.

It's using a multiplexing method that we've used before, known as 'Charlieplexing', to allow the twelve LEDs to be driven from just four I/O pins. See the side panel for more information on this.

We've used a PIC12F675 as the microcontroller primarily because it has a low sleep current and comes in a modestly-sized 8-pin SOIC package, with enough spare I/Os to drive the 12 LEDs.

As the board is shaped like a tree, the obvious choice of

Front and back of the PCB, shown here life size. The LEDs are mounted on the front with their cathodes to the left, (indicated by a small green mark on the LEDs we used). We used a mix of LEDs on our tree; the different colours are hard to tell apart once they have been removed from their package, although white LEDs can sometimes be discerned by their yellow phosphor. On the back of the PCB are the rest of the components: the PIC IC, five resistors and the button cell battery. The orientation of the IC and the cell holder is important. While not visible from directly above, the holder has tabs on its left-hand side that prevent a cell passing out this side. If the holder is installed backwards, one resistor gets in the way of inserting the coin cell.

solder mask colour is the default green, and you can use whatever colour LEDs you want on top of that: red, green, yellow, orange, blue, white or a mixture.

But as ornaments are best when they're bright and cheerful, we're also offering boards with red and white solder masks, along with the green mask shown here.

You could build a mix and use different colour LEDs with each board colour.

Circuit details

Fig.1 shows the full circuit (not much to it, is there!). Two of microcontroller IC1's eight pins are dedicated to its power supply, and these are connected directly to the terminals of a button cell.

We've found that in this application, no bypass capacitor is necessary. These power pins and three other pins required to program the chip in-circuit are also connected to programming header CON1.

This header is mounted on a part of the PCB that's separated from the rest by a row of holes, allowing it to be snapped off if it isn't needed (eg, if you purchased a pre-programmed PIC, or you've already programmed the chip).

Pins 6, 2, 5 and 3 are used to drive the LEDs via 1kΩ current-limiting resistors, leaving pin 4 (GP3/$\overline{\text{MCLR}}$) and pin 7 (GP0/PGED), both of which can be used as I/Os but in this case, are only used for programming the chip.

Part of the reason we aren't using pin 4 to drive the LEDs is that during programming, a high voltage is applied to this pin, which could damage the LEDs.

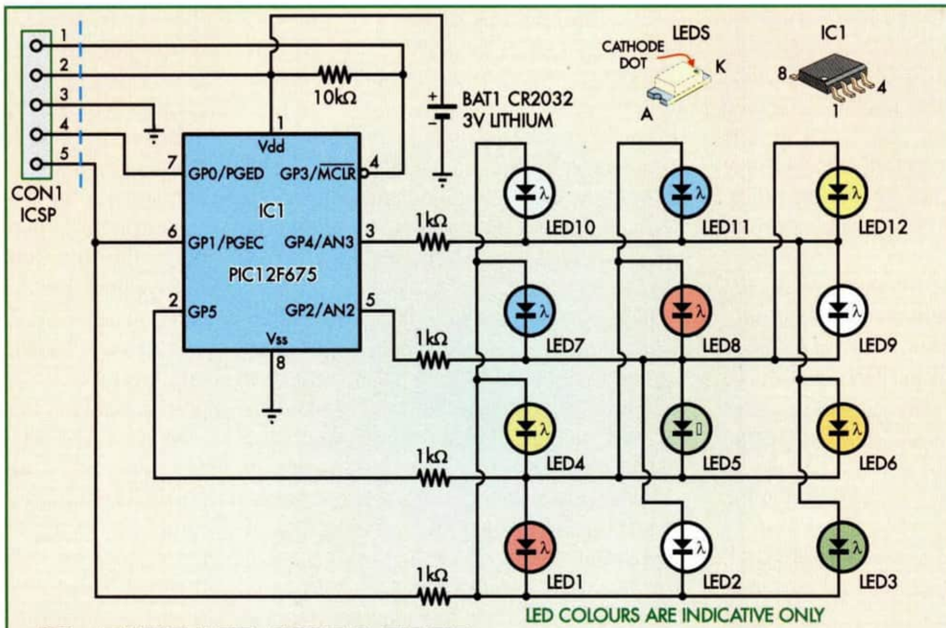


Fig.1: the circuit is simplicity itself, involving just one IC, five resistors, twelve LEDs and a lithium cell. Each LED is connected across a different pair of pins, via two 1kΩ series current-limiting resistors. The optional in-circuit programming header is on a section of the board that snaps off in case you don't need it. You can also fit it and snap it off after you have finished programming IC1.

In its role as $\overline{\text{MCLR}}$, pin 4 needs a pull-up for normal operation (to avoid 'random' resets). So we've connected a 10kΩ resistor between $\overline{\text{MCLR}}$ and Vdd.

Pin 7 is also not used for the LEDs as this might interfere with the programming of the chip. Pin 6 (GP1/PGEC) is used for both programming and driving the LEDs. We've gotten away with this as it is the only programming pin that connects to the LED array.

The circuit is designed to drive one LED at a time. While it is possible to give the illusion of multiple LEDs being illuminated by multiplexing them fast enough, we've found that we can get a nice display by flashing the LEDs in sequence, and therefore that is not necessary.

With this configuration, the current for each LED passes through two of these resistors. That's because, to drive an LED, one of the four connected I/O pins is driven high and another of the four low. The two remaining I/Os are left in a high-impedance 'floating' state.

This forward-biases one of the twelve LEDs. The other eleven are either reverse-biased, or they have a floating anode or cathode, so no current can flow.

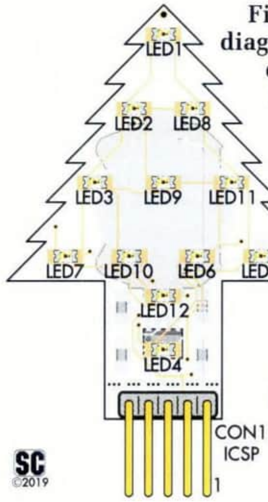
Table 1 shows which combination of pins is used to light each LED in turn.

To simplify the layout of the PCB, the LEDs are not arranged in numerical order. The mapping of logical to physical location is handled in the software programmed into the PIC. The physical layout can be seen in the top side PCB overlay diagram, Fig.2(a).

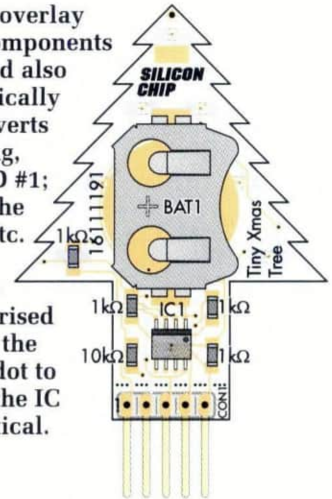
By the way,

LED	High pin	Low pin
LED1	GP5	GP1
LED2	GP2	GP1
LED3	GP4	GP1
LED4	GP1	GP5
LED5	GP2	GP5
LED6	GP4	GP5
LED7	GP1	GP2
LED8	GP5	GP2
LED9	GP4	GP2
LED10	GP1	GP4
LED11	GP5	GP4
LED12	GP2	GP4

Table 1 - LED drive pin combinations



Figs.2a (left) & 2b (right): these overlay diagrams show where to fit the components on both sides of the board, and also show how the LEDs are physically arranged. The software converts this into a simple mapping, numbering the top-most LED #1; the second row #2 & #3; the third row #4, #5 and #6 etc. This makes designing patterns in the software easier. All the LEDs are polarised and must be installed with the orientation shown (cathode dot to the left). The orientations of the IC and cell holder are also critical.



while it is possible to run LEDs directly from I/O pins in some cases, we decided to use series current-limiting resistors.

Each I/O pin can source or sink up to 25mA, and if the button cell had to supply this much current (even briefly), its voltage would sag quite badly, possibly leading to microcontroller glitches which would interfere with the pattern. This would also likely lead to a short cell life.

Software

The software is designed with low power consumption in mind, so the processor spends much of its time in sleep mode, only being woken occasionally by the "watchdog" timer. This is an essential part of our recipe for minimal power usage.

The software initialises the I/O pins and assigns its internal prescaler to the watchdog timer, allowing us to alter the prescaler and thus change the watchdog delay.

It then sets up an array of values containing the numbers of the LEDs that should be lit in sequence. The program then steps through this array, lights one LED and then puts the processor into sleep mode for 18ms. After this, the LED is switched off, and the processor sleeps for another 72ms.

The processor cycles through the array and repeats this sequence as long as it has power. Changing the pattern is as simple as changing the array in the code.

By having the processor sleep nearly all the time, the vast majority of the power used is consumed by the LEDs, with a small amount being dissipated in the series resistors and an even smaller amount by the microcontroller during the brief periods that it is active.

LED Charlieplexing

The technique we use for driving the LEDs in our Tiny Christmas Tree is called Charlieplexing, named for Charlie Allen of Maxim Integrated. Maxim is known for their LED driver ICs, such as the MAX7219 which we described previously in the LED Matrix Display Module article from June 2017 (siliconchip.com.au/Article/10680).

Charlieplexing, as the name suggests, is a variation of traditional multiplexing. We used Charlieplexing in our Digital Up/Down Timer project in the August 2010 issue (siliconchip.com.au/Article/240).

Traditional multiplexing arranges the LEDs in a grid, with one set of pins to drive the anodes and one set to drive the cathodes.

If you have 100 LEDs in a 10x10 grid, 20 output pins are required to drive them. Or 12 LEDs in a 3 x 4 grid would require a total of seven output pins; a bit difficult when you are using an 8-pin micro!

With Charlieplexing, each I/O pin can effectively be used to drive both a row and a column (except that it can't be both at the same time), so the number of LEDs that can be driven from the same number of pins is greatly increased. In our Tiny Christmas Tree, the four I/O pins can now drive 12 LEDs.

The biggest disadvantage of Charlieplexing is that you can't illuminate as many LEDs at the same time. With regular multiplexing, you can light up a whole row or column at once, whereas with Charlieplexing, you can only really light up one at a time.

Also, planning the wiring for such a Charlieplexing arrangement is tricky, especially as large arrays of LEDs are typically arranged in a grid, which lends itself well to the row/column principle of basic multiplexing. This also leads to increased software complexity.

For Charlieplexing, the I/O pins used need to be capable of being driven high, driven low and also being set to high-impedance. This makes it harder to use discrete transistors to implement such a scheme. Regular multiplexing requires the pins only to switch between high and low, or active and high-impedance, which is easier to do with discrete transistors.

Other restrictions apply to Charlieplexing. In particular, the forward voltage of the LEDs must be within a certain range. If the forward voltage of one particular LED is more than the forward voltage of some other pair of LEDs in the Charlieplex' matrix', current will pass through the other pair, leading to some LEDs lighting up when they should and some when they shouldn't.

Fortunately, our simple circuit does not suffer from this; any combination of red, yellow, green, blue and white LEDs can be used.

You might have trouble if you try to use an infrared LED, but we aren't sure why you would want to do that!

While it is possible to illuminate multiple LEDs at a time with Charlieplexing, we have chosen not to do so in this project. Simplicity is the first reason; there are restrictions on which LEDs can be lit together. By lighting one at a time, we do not need to consider that.

Lighting one LED at a time also reduces the peak current needed from the cell, which is vital for getting the most life out of the button cells. The usable mAh rating of button cells is considerably less at higher currents.

Tests on our prototype measured a typical average current draw of 70μA, which should allow weeks of operation from a fresh coin cell. That's well and truly enough to run the ornaments for the 12 days of Christmas, and beyond!

Our best estimate is that a new CR2032 cell with a nominal capacity of 240mAh will last from the start of December until around the middle of March, although we haven't tested this. That will mean you can take them down just before Easter!

We've chosen 1kΩ current-limiting resistors to give a long battery life and sufficient brightness for indoor use. If you want the LEDs to be brighter, possibly bright enough to be used in sunlight (but out of the rain!) then you can reduce these values.

We tested resistors as low as 100Ω, and the Tree worked fine, although we would expect its battery life to be proportionally reduced (to around a week for 100Ω).

Construction

To ensure that the Tiny Christmas Tree is, well, Tiny, we are using surface-mounted components. We have also done away with markings on the front of the PCB to give the Tree a more presentable appearance.

Thus there are components on both sides of the PCB. Fig.2(a) shows the top side component overlay, with Fig.2(b) showing the components fitted on the opposite side.

The SMD parts are mostly a large size, ie, 3216 metric (1206 imperial). IC1 is in an 8-pin SOIC package. All these parts are quite manageable, even with a fairly large-tipped iron.

Having flux paste and tweezers will make this much easier. Solder braid (wick) will also be handy if you end up bridging any pads. You might also use a small piece of adhesive putty (such as Blu Tack) to hold the PCB in place as its small size means it could move around easily while you're trying to line up the components.

Start by fitting IC1. Apply flux to the pads and note the orientation of the pin 1 marking on the IC. It needs to align with the notch in the silkscreen on the PCB.

Place the IC onto the pads and align it as best you can. Gently hold it in place with tweezers, then apply a small amount of solder to the iron tip and touch it to one pin of the IC. The flux will help pull the solder onto the pin and its pad.

Check that the other pins are correctly lined up with their pads. If they are not, grasp the IC with the tweezers and move it into place while using the iron to remelt the solder. Once the IC is correctly aligned, touch the iron to the remaining pins. You may need to apply more solder as it is sucked from the tip onto the pins and pads.

If you have a bridge between two pins, solder the remaining pins before attempting to remove it. Having all the pins soldered will help to keep the IC in the correct location.

To remove the bridge, apply some more flux to the top of the pins and press the end of a piece of braid against the pins with an iron. This should absorb any excess solder onto the the braid, leaving just enough to maintain a good joint.

The next step is to install the resistors. There is a single 10kΩ resistor and four nominally 1kΩ resistors, none of which are polarised. Fit the 10kΩ resistor first, where shown. Apply a small amount of solder to the pads and hold the resistor in place with the

Parts list – Tiny LED Christmas Tree

(A kit is available from the SILICON CHIP ONLINE SHOP; see below right)

1 double-sided PCB coded 16111191, 54mm x 41mm (green, red or white solder mask)
1 5-way right-angle header strip (CON1) (optional, for programming)
1 surface-mount coin cell holder [Digikey BAT-HLD-001-ND, Mouser 712-BAT-HLD-001 or similar]
1 PIC12F675-I/SN 8-bit microcontroller, SOIC-8, programmed with 1611119A.HEX (IC1)
1 10kΩ 3216 (1206 imperial) SMD resistor [Altronics R8188]
4 1kΩ 3216 (1206 imperial) SMD resistors [Altronics R8116]
12 3216 (1206) SMD LEDs (any colours) [Altronics Y1041, Y1056, Y1073, Y1079, Y1085]
1 CR2032 coin cell or similar (CR2025 is also suitable)

tweezers. Load up the iron with a small amount of solder and apply it to one pad; the solder should flow into the join.

Check that the part is within the pad markings and if not, adjust it. Solder the second pad. If you have the right amount of flux, the solder between the pad and component should look smooth and shiny. Have a look at the photos of the board to see how it should look.

Do the same for the remaining resistors. There are three more around IC1 and one up and to the left amongst the branches of the tree.

Now flip the board over so that you can fit the LEDs. The markings on this side are minimal, to avoid spoiling the visual effect. You might be able to see a small line indicating the LED cathodes. All LEDs should be mounted with their cathode to the left.

This is usually marked with a small green dot on the LED body, although it's best to verify this using a DMM set on diode test mode before soldering them. When the LED lights up, the red probe is on the anode and the black probe on the cathode.

You can fit any colour LED to any location. We tested one of our prototypes with a mix of white, red, yellow, green and blue LEDs and found that they all worked fine. If you use high-brightness types, then you will get the best results from the meagre current they are supplied.

These are fitted in the same manner as the resistors, although you may find it takes a little more heat to make the solder joints.

The coin cell holder should be fitted next. Check that the cell opening faces away from the resistor, as shown in Fig.2(b) and our photos. Otherwise, you might have trouble getting the cell in later.

The holder should be mounted similarly to the other components, with a small amount of flux paste to help the solder flow smoothly. You will probably need to use more heat than the smaller components. Tack one end, check that the holder is straight and symmetrical and then solder the other end.

If you can't get the specified coin cell holder, you can substitute a 40mm length of 0.7mm tinned copper wire. We tested this by rigging one up, and it worked well enough.

We started by bending the wire into a gentle curve in the middle, with a sharp 180° bend of approximately 2mm radius at each end. The bends give a bit more springiness and help to hold the cell in place.

To place the wire at the correct height to provide sufficient tension, we placed a spare PCB (standard 1.6mm thickness) between the Tree board and the wire. Once soldered and the weight released, the wire will spring back a small amount to allow a 3.2mm thick CR2032 cell to fit underneath.

Balance the wire on the PCB and apply a good amount of solder to each end to hold it in place.

With the soldering complete, remove any excess flux with an appropriate cleaning solution to ensure that the front of the PCB presents a clean appearance.

Important!

Coin cells (like button cells and other small batteries) can be dangerous if they are ingested. The Tree and any batteries that might go inside it should be kept well away from small children and babies that might (nay, WILL) try to put such things in their mouth at the first opportunity!

We found that it was tricky to remove the coin cell without something thin to push it out of the holder. If you have substituted a piece of wire for the cell holder, the cell will not be held as tightly. Nonetheless, children will find a way. So it's best to keep the Trees well away from children.

You might consider adding a little glue or silicone sealant to the side of the cell to make it harder to remove, or even wrap the tree in a piece of large-diameter clear heatshrink tubing, which would also provide a measure of protection against being dropped, getting splashed etc.

Programming

If you have a pre-programmed microcontroller (eg, from the SILICON CHIP ONLINE SHOP), you don't need to worry about this step and can jump ahead to the completion step.

To program the PIC requires a suitable programmer, such as PICkit 2, PICkit 3 or similar. While a five-way header can be soldered onto the pads at the bottom of the PCB, you can also press it in place for the duration of the programming if you only intend to do this once.

Not having the header soldered in place will also make it easier to break off the tab later.

Load the HEX file (available for download from the SILICON CHIP website) into your programmer application and plug the header strip into the programmer. The pin marked with the arrow symbol on the programmer should line up with pin 1 of the header. This is marked on the back of the PCB and also has a rectangular (instead of oval-shaped) pad.

Then press the button to program the PIC. The LEDs should start flashing immediately, if the programmer is set to allow the PIC to run after programming.

Unplug the programmer and fit a coin cell. Take care to avoid having a coin cell fitted while the programmer is connected, as most coin cells will not take kindly to receiving a charge from the programmer's 5V supply.

The LEDs should cycle up the Tree and from left to right. If one or more LEDs do not light up, check that they are correctly soldered and orientated.

If only three or six of the LEDs are lighting, then one of the resistors may not be connected correctly, or one of the IC's pins may not be soldered properly.

If the pattern seems to be random, then your LEDs may have a mark on their anode instead of their cathode, which unfortunately sometimes seems to be the case. In this case, all LEDs will operate, but out of sequence. The only solution is to remove and reverse them all.

If you have a different problem, remove the cell and check your construction carefully before reinserting it.

Completion

Once IC1 has been programmed and all the LEDs are operating correctly, the programming header can be removed. You may wish to leave it attached if you want to reprogram the IC later (eg, to change the pattern) or use the header to apply power to the board.

We think that the tree looks nicer without it.

There is a row of small holes across the end of the PCB so that it can be snapped off cleanly. Before snapping, gently score or file along this line on both sides of the board, to break the copper tracks. If this is not done, the traces may tear and lift off the PCB, causing damage to tracks that you need for it to operate.

Once scored, snap off the end of the PCB with a wide-jawed set of pliers. You can then file down the rough edge; it's best to do this outside and with a face mask so that you do not inhale any fibreglass dust.

If you want the tree to stand up on its own, you could instead leave the bottom tab in place and solder a short piece of wire to the centre pad. This is connected to ground, so care should be taken that this does not contact any other part of the circuit.

Mounting

There are two small pads near the top of the PCB which are designed to allow the Tree to be hung.

A loop of wire can be soldered to the small, round through-hole pad right at the top, allowing it to be hung as a tree ornament.

The square pad on the back can be soldered to a safety pin so that the Tree can be worn as a brooch.

Changing the pattern

The source code is included with the HEX file download from our website. This contains project files which can be edited with Microchip's MPLAB X V5.05 or later (a free download).

The sequence of LEDs is programmed into an array near the start of the "main.c" file, so modifying the values within is the easiest way to change the pattern.

The pattern sequence can be made up to 255 steps long by changing the contents of the pattern[] array. The numbers refer to the physical position of the LEDs, with number 1 at the top, 2 and 3 in the second row etc. The number 0 can be used to have no LEDs lit for a step.

As mentioned earlier, our first prototype used 100Ω LED current-limiting resistors instead of 1kΩ. This made the LEDs much brighter, but the button cell did not last anywhere near as long. But if you just want the ornament to run for a few days over Christmas, that would be a good option.

Alternatively, if you have a source of 5V DC power, you can opt for the brighter option and power the Tree via pins 2 (5V) and 3 (GND) of the programming header, or via the coin cell holder pads.

You can also paint the PCB if you wish to change the appearance or add some colour, although it would probably be easier to purchase some different-coloured PCBs from our Online Shop.

As noted earlier, we will have boards with green, red and white solder masks available. SC