# Touchscreen Appliance

## Part 2 – By JIM ROWE & NICHOLAS VINEN


**Last month we introduced our new Appliance Energy Meter. It uses a 2.8-inch touchscreen to display energy usage data and has handy features such as cost calculation based on time-of-day tariffs, graphing and logging. This second instalment will take you through the process of building the PCBs and assembling the whole unit, as well as describe some of the interesting features of the software.**

The Appliance Energy Meter consists of two modules. The larger PCB hosts the custom circuitry for this project while the smaller one is used to build the Micromite LCD BackPack, which provides all the control, display and user interaction functions.

The BackPack module and main PCB fit into a UB1 jiffy box along with a mains fuseholder and two cable glands to secure the mains wiring.

The February 2016 article introducing the LCD BackPack has the full construction details, although it's pretty self-explanatory.

We can supply a kit with all the parts for the BackPack (including some of the mounting hardware you'll need later), and you just fit the kit components to the PCB where indicated on the silkscreen.

Once you've assembled the BackPack, including the display, check that it works if you can but don't go any further. We'll program it after building the main PCB. If you're programming the PIC32 chip on the BackPack PCB yourself, now would be a good time to do that.

### SMD parts

The main PCB has just three SMD ICs plus about 20 passive components. Refer to the PCB overlay and wiring

# nergy Meter

diagram, Fig.3. IC2 has a relatively fine pitch while IC3 and IC4 are easier to solder. So fit IC2 first. This can be done with a standard soldering iron. The only extra tools you need are a good light, some flux paste (available from Jaycar, among other stores), solder wick, flux cleaner (eg, methylated spirits or pure isopropyl alcohol) and some sort of magnifier for checking the solder joints.

There are a few different techniques but unless you happen to have a hot air or infrared reflow set-up, they're pretty similar. Start by depositing a little solder on one of the corner pads – try not to get any on any of the other pads. Then you have two options, depending on which you think will be easier.

You can either place the LTC1863 in position, check that all its pins are properly aligned over its pads and that pin 1 (indicated with a dot or divot) is at upper left as marked on the PCB and shown in Fig.3. Then, while gently pressing the IC down onto the PCB, heat the solder on the pad that you deposited earlier so that the associated pin sinks down into it. Then re-check the positioning and solder the diagonally opposite pin.

Alternatively, you can position the IC next to its pads with pin 1 in the correct orientation and, while heating the solder on that one pad, slide it into position using tweezers or a couple of fingers. Then check that all the pins are correctly located over the associated pads. If not, reheat that solder joint and gently nudge it into position before soldering the diagonally opposite pin.

*We've had to make a minor circuit change since the first article on the Appliance Energy Meter was published last month.*

*We've added a 100nF capacitor between the Earth terminal on CON8 and the VREF pin (pin 10) of IC2. This reduces the effect of noise from switching regulator REG1 on the operation of the analog-to-digital converter.*

*As a result of this change, we've decided not to supply the RevF PCB as mentioned in the parts list last month, which needed an adaptor board, and instead wait for the RevG boards to arrive which incorporate both changes and thus do not require the extra assembly work.*

Either way, you should now have the IC located properly and pinned down so it's just a matter of soldering the remaining pins. You can attempt to do this one at a time, by first applying flux along all the pins and then touching the tip of the soldering iron, loaded with a little solder, onto the very ends of the PCB pads. Alternatively, simply solder the pins two or three at a time, then apply flux paste and use solder wick to remove the excess solder.

Regardless of which method you use, make sure to refresh the solder on those first two pins and use the flux paste and solder wick to clear any bridges between pins. Finally, clean off the flux residue using your solvent of choice and a lint-free cloth, then inspect the IC under a bright light and high magnification to ensure all solder joints are good. If any do not look 100% or you find any bridges, apply some flux and heat (and if necessary, solder wick) until it all looks good.

Then solder IC3 and IC4 using the same technique although you should find them significantly easier due to

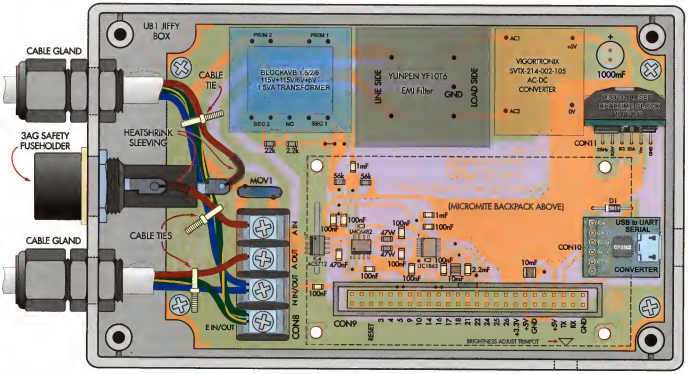



**Fig.3: this diagram shows not only the component layout on the PCB but also its connections and placement within the UB1 Jiffy Box. Take care when identifying (and then soldering) the surface-mount components onto the board – all SMDs should be in position before mounting the transformer, EMI filter, AC-DC converter, serial converter, CON8, 9, 10 and 12.**

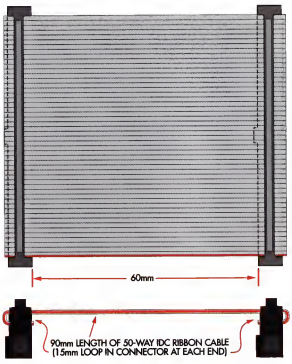



**Fig.4: here's how the IDC cable is made up with its two connectors – note the loop and direction of the cable.**

the larger pins and wider spacing.

The next step is to fit the six SMD resistors and 15 SMD capacitors using a similar technique. Basically, you just tack them in place at one end, then solder the other end before refreshing the initial joint. Make sure that solder flows onto both the PCB pad and the leads of each device.

The resistors will be marked with value codes on the top (eg, 223 or 2222 = 22k). Capacitors will be unmarked so you will need to take care not to get them mixed up after removing them from their packaging. None of these parts are polarised.

## Through-hole components

Now fit through-hole diode D1 with the cathode stripe positioned as shown and then solder a standard 6-way pin header in place for CON10.

Now fit the box header in place for CON9 with the notch towards the top as shown in Fig.3 and on the PCB silkscreen. Follow with the 1000µF electrolytic capacitor, with its longer lead through the pad indicated with the plus sign. You can now fit the 4-way terminal barrier for CON8 along with the USB/serial adaptor board, which is soldered to the pin header already in place on the PCB, with the microUSB socket on the top. MOV1 can then be fitted and it can go in either way around.

That just leaves transformer T1, the EMI filter, the Yunpen AC/DC adaptor and real-time clock modules. Ensure that each one is pushed down fully onto the PCB before soldering its leads. Only the real-time clock module can be installed the wrong way around – it's mounted vertically but make sure that the main body of it sits next to the AC/DC converter; see Fig.3. If unsure, check the labelling on this module's pins and line them up with the corresponding labels on the PCB before soldering.

Note that you will want to fit the button cell to the real-time clock module before soldering it to the board and that, if you are using a non-rechargeable (primary) cell, you will need to first desolder the surface-mount, glass-encapsulated diode from the module so that the module won't try to charge it when power is applied.

## Making the connecting cable

The LCD BackPack and main PCB are joined by a 50-wire IDC cable that's around 60mm in length. You will need to crimp the two 50-way IDC sockets onto either end of the cable as shown in Fig.4. You can either use a vise, with protective pieces of timber on either side of the assembly or an IDC crimping tool such as the Altronics T1540. We don't suggest you use a different tool such as pliers since this is likely to result in the plastic connector fracturing.

Once you've made the cable, taking care that the ribbon is properly aligned and the connectors are fully clamped down, the next step is to do some basic checks to make sure everything is working before putting it in the case.

## Initial testing

First, plug a microUSB cable into the socket on the main board and plug the other end into your computer. The red LED on the real-time clock module should light up.

If you are using Windows 10, Mac OSX or Linux, the serial port should be automatically identified. If using an older version of Windows, download and install the Silicon Chip USB serial driver. Verify that a new serial port is available, eg, by running a terminal emulator such as TeraTerm Pro and checking the list. You can open it and type some characters into the terminal but all that will happen is the LED on the USB/serial adaptor should flash.

Once you've verified that the USB serial port is working, unplug the cable and connect the BackPack module to the main PCB, using the ribbon cable you prepared earlier. This can only go into the socket on the main PCB

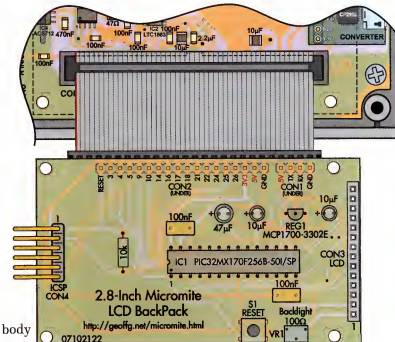



**Fig.5: how to connect the LCD BackPack to the main PCB for testing (the TFT touch screen is not shown for clarity).**

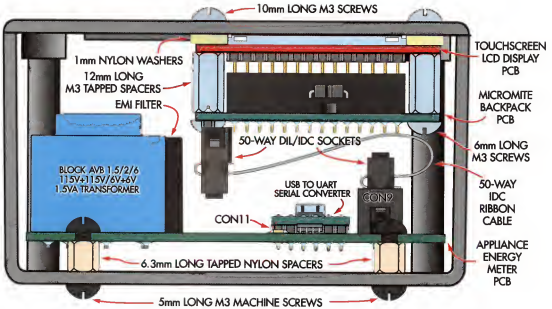



**Fig 6: it's a snug fit but all the components mount inside the UB1 Jiffy box, as shown here. Compare this and the photo on page 88 when assembling. Case drilling details are shown on page 95. Note: this diagram is shown oversize, for clarity.**

one way but you will need to be careful to plug it into the BackPack with the correct orientation; refer to Fig.5. For the moment, rest the BackPack module on your bench top as shown. Note that the TFT is not shown fitted on top of the module, for clarity.

The trick here is to make sure that the GND pin of the BackPack goes into either of the right-most holes on the IDC socket. You can then check for continuity between GND points on the two boards to confirm that it is located correctly; for example, place one probe on the via just to the right of the 10uF capacitor immediately to the left of the USB/Serial converter on the main PCB and the other probe on pin 3 of the BackPack ICSP header (CON4).

Now plug the USB cable back in. If you've used a microcontroller that was pre-programmed with the Appliance Energy Meter firmware, almost immediately you should

## Uploading the code to the Micromite chip

Most constructors will simply purchase a pre-programmed PIC or download and install the HEX file which includes MMBasic along with all our code, so that the micro is ready to go. But some readers may wish to modify the code and because we had to resort to some tricks to make it fit, here is the multi-step procedure used to load it.

First, program your PIC32 with the MMBasic 5.1 firmware and establish a serial console connection using the USB converter. You will need to set up the display and touch panel as detailed in the February 2016 article on the LCD BackPack. Note that the BackPack (and, if attached, the main board) are powered from the PC during the programming process.

The first step is to load the **SCAppEnergyMeter_Library.BAS** into the Micromite. First, download the code from the SILICON CHIP website, then grab a copy of Jim Hiley's Windows/Linux "MMEdit" program. It is freeware and available from **www.c-com.com.au/MMEdit.htm** For Windows, download the setup file called MMEdit.exe and run it. It works on any Windows version since XP.

Run MMEdit and open the BASIC file mentioned above. Next, ensure the "Auto crunch on load" option in the Advanced menu is selected and set up the COM port to communicate with the Micromite by selecting the "New..." option under the Connect menu. You can then click the "Load and run current code" button, right-most in the toolbar under the menu (with the icon that looks like a blue stick figure). You should get a progress dialog and the upload will take around 30 seconds.

If it fails, close this window and re-check the COM port settings (make sure you don't have this open in another program).

Once the upload is complete, the MMChat console window should automatically appear. You can then execute the "LIBRARY SAVE" command (note: if you have previously done this, you will need to run "LIBRARY DELETE" first). After a brief delay, it should display the MMBasic prompt, "> ". You can verify that the code was saved by issuing a "MEMORY" command, which should yield a response like:

```
> memory
Flash:
  0K (0%) Program (0 lines)
 18K (31%) Library
 42K (69%) Free
```

Now open the **SCAppEnergyMeter_Main.BAS** file (which is supplied in the same ZIP as the BASIC file loaded earlier) and, again ensuring that the "Auto crunch on load "option is enabled, upload that to the PIC32. The MMChat window should appear once this is complete. You can then type in "OPTION AUTORUN ON", press enter, then execute the "RUN" command to start the program.

Note that this will fail, with a real-time clock error, if the BackPack is not yet plugged into the main board. Regardless, you can now unplug the USB lead and proceed with the remainder of construction/set-up.

see the main screen come up. The readings may not all initially be zero but they should drop to zero after a few seconds (you may get a current reading of around 60mA since the unit has not been calibrated yet). Touch one of the elements on the screen and verify that it takes you to a different screen.

If your microcontroller has been programmed for the Micromite Mk2 but you do not yet have the Appliance Energy Meter software installed, connect to the USB serial port with a terminal emulator set to 38400 baud and press the reset button on the LCD BackPack. You should see the Micromite prompt in your terminal emulator. You can then use the multi-step procedure detailed in the side-panel to load the firmware.

Once the software is running, it's a good idea to check that the real-time clock and ADC are working. Checking the real-time clock is quite easy; press on the time and date in the lower-right corner of the screen to set it, then once it has been set, pull out the USB plug and then plug it back in. Once the unit restarts, it should retain the date and time. That indicates the real-time clock and its backup battery are OK.

Testing the ADC is a bit more tricky. If you're getting a zero voltage reading, that's a good sign. However to be sure, the easiest way is to pass some current between the "A IN" and "A OUT" terminals on the PCB (eg, from a DC supply with a current-limiting resistor) and check that it registers on the display.

You can reverse the polarity and you should get a similar reading but note that it won't be exactly the same, as the unit has not been zeroed yet.

If you have a fully programmed BackPack but get a blank display, there are a few things that might be wrong. Firstly, check that the ribbon cable has been made properly and correctly plugged in at both ends. Check also that the red LEDs on the real-time clock module and USB/Serial modules are lit. Run a terminal emulator, connect to the USB serial port at 38,400 baud and press the reset button on the LCD BackPack. See if you get any error messages which may give you a clue.

For example, if the micro can't communicate with the real-time clock it will issue an error message and halt. You would then need to check that the clock module is soldered properly and orientated correctly.

If you're getting nothing on the display and no error messages over the serial console, there is likely something wrong with the BackPack module itself, possibly in the TFT connections or a bad component or solder joint, so check it carefully.

## Case preparation

The next step is to prepare the case. First, drill the holes for mounting the main PCB in the base. You can either use the diagrams on P95, use the PCB as a template, positioned as far to the right as possible (see Fig.3) or download the drilling diagram from the SILICON CHIP website and use that as a template. The four mounting holes are drilled to 3mm.

Now fit 6.3mm tapped Nylon spacers to the inside of the box using 5mm M3 machine screws and tighten them up. We recommend the use of Nylon machine screws for the attachment of the spacer at lower left (both top and bottom), which will be closest to the mains wiring when the unit is complete.

If you can't get a 5mm Nylon machine screw, consider using a longer Nylon screw fed through an untapped spacer and secure with a Nylon nut, although this will be a lot more "fiddly" to attach.

Now is a good time to attach some rubber feet to the bottom of the box. Adhesive types are the easiest, however you could use slightly longer screws to attach the Nylon spacers and also hold screw-on feet in place (but make sure they don't project any more than 3mm into the spacers), or simply drill four extra holes and attach the feet that way.

Before fitting the PCB into the box, drill the three round holes at the left end for the mains cable and fuseholder and make the rectangular cut-out on the right side for access to the USB socket. Details are in the drilling and cutting templates mentioned above and available for download from our website, as well as being shown on page 95.

The best approach for the round holes is to start with a small drill (eg, 3mm) and use either a tapered reamer, stepped drill bit or series of larger drill bits (going up by 1-2mm at a time) so that the holes are nice and round. Once they're large enough, test fit the components, then de-burr the holes using a larger drill bit or countersink tool.

The rectangular cut-out can be made by drilling a series of holes inside the perimeter with a small bit, cutting the remaining plastic to remove the inner piece, then filing the edges smooth and flat with a flat needle file.

The drilling diagrams also show a hole in the front of the box, so that you can access the brightness adjustment trimpot on the LCD BackPack board with a small screwdriver. This is optional however it may be a good idea as it will allow you to reduce the display brightness for lower power consumption during long-term power logging and then increase it again when you want to read the results.

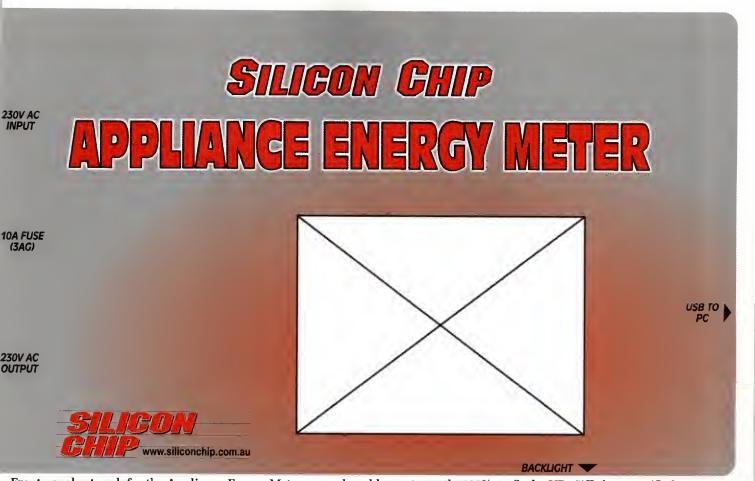
For the lid, a large, rectangular cut-out plus four 3mm mounting holes are required to suit the LCD BackPack. It's quite hard to do a neat job cutting the hole for the display. By far the easiest approach is to simply buy a replacement laser-cut black acrylic panel from the SILICON CHIP online shop.

You may need to use longer self-tapping screws than those supplied with the case, as this panel is slightly thicker than the existing lid and lacks the recessed holes for the screw heads – it depends on how long the supplied screws are as this can vary, based on case manufacturer. But it does give a neat appearance and you can still attach a lid label should you wish to. Alternatively, download the cutting diagram and make the holes in the original lid, using a similar technique as described above.

## Putting it all together

The next step is to fit the 3AG safety fuseholder into the centre hole in the left-hand end of the box, using the pliant washer and mounting nut supplied. Tighten up the nut firmly, with the body of the fuseholder positioned so that the side connection lug is in a position that allows easy access for soldering.

You can then mount the two cable glands. Tighten up the internal nuts to secure the bodies of the glands but leave the outer nut loose. Now cut the 3m 230V/10A extension cable in half. If you don't have a 60mm length of 10A brown mains wire handy, cut a 60mm long piece off the input cable (ie, with the 3-pin plug on the end) and strip its insulation off.




**Front panel artwork for the Appliance Energy Meter, reproduced here at exactly 100%, to fit the UB1 Jiffy box specified in the parts list last month. Note that holes are not shown – drilling details are in the diagram on page 95.**

Strip 6mm of insulation from one end of the 60mm brown wire and 10-12mm from the other. Solder the shorter bared end to the fuseholder's side connection lug, making sure to produce a reliable joint, then slip a 15mm length of 5-6mm heatshrink sleeving over the joint and shrink it down (eg, using a hot air gun), making sure it covers as much of the exposed metal as possible.

Now remove about 85mm of the outer sheath from the cut end of the input cable, to free the three insulated wires inside. Cut the brown (Active) wire shorter than the others, to about 40mm, and remove about 6mm of the insulation from the end. At the same time, 10-12mm of insulation can be removed from the ends of the blue (Neutral) and green/yellow (Earth) wires.

Then push all three wires into the box through the input cable gland, at upper left. You may need to remove the outer nut entirely but don't lose the rubber sleeve in the process. Slip another 15mm length of 5-6mm heatshrink tubing over the brown wire and push it all the way down, then solder this wire to the lug at the rear of the fuseholder, making sure you make a secure and reliable joint. Once this has cooled down, slip the heatshrink tubing over the joint and shrink it down.

The next step is to lower the PCB into the case and secure it to the previously installed mounting posts at each corner, using M3 x 5mm machine screws.

As mentioned earlier, use a Nylon screw in the lower-left corner. You can then secure the bare end of the wire from the fuseholder under the clamping plate of the top-most terminal of CON8, labelled "A IN". Route the wire to the side of the screw furthest from the adjacent terminal and make sure there are no loose strands of copper and that it's screwed down firmly.

You can now remove about 40mm of the outer sheath from the end of the remaining half of the extension cable, ie, with the 3-pin socket at its other end. Having done that, strip 10-12mm of insulation from each of the three insulated wires. Push this through the other cable gland and feed the two blue Neutral wires (ie, one from this cable and one from the other) into the "N IN/OUT" terminal of CON8 and clamp them down firmly.

You can now complete the wiring by doing the same with the green/yellow striped Earth wires and finally, the brown Active output wire; see Fig.3 and the photo for details.

Having completed the wiring, gently pull the two mains leads out of the cable glands until there is only a little slack left on the internal wires, then screw the gland nuts down firmly and add cable ties where shown in Fig.3.

Note that we're going to glue the gland nuts in place later, so that they won't come undone, but we want to do some more testing before making it permanent.

## Connecting the BackPack

The BackPack can now be secured to the inside of the lid; see Fig.6 for details. Remove the screws holding the LCD onto the spacers and feed four 10mm M3 machine screws through from the top side of the lid.

You may want to countersink these holes or use black screws to match the lid.

Pop 1mm-thick Nylon spacers (3mm inner diameter, 6mm outer diameter) over the ends of the screws, then feed them into the spacers through the LCD panel, with

## Fitting the software into the Micromite

During the development of this software, we struggled to fit the required functions into the available flash memory and RAM of the 28-pin Micromite Mk2. While surface-mount PIC32s have up to 512KB flash and 128KB RAM, the DIP-package versions are limited to 256KB flash and 64KB RAM, with roughly 50KB of each available to MMBasic.

### RAM limitations

Our goal was to be able to log up to one week of data to RAM, with a maximum logging interval of one minute. We managed to compact the voltage, current and power readings into 32 bits (four bytes). So one week of data requires 60 (minutes) x 24 (hours) x 7 (days) x 4 (bytes) = 40,320 bytes of RAM.

After that and taking into account MMBasic's overhead, that left us with about 10KB of RAM. That sounds like a lot, given that our program requires less than 1KB of general variables. Unfortunately though, during software development, we frequently ran out of memory and had to make significant changes to the software to work around this limitation. We also had to frequently rationalise the code so that it (and the fonts) would fit into the 50KB of available flash program space.

To make matters worse, changes to reduce RAM usage would often increase flash usage and vice versa. So we had to perform iterative optimisation, reducing the memory footprint, then shrinking the flash space used, then reducing the memory footprint again and so on until we were able to get all the required functions into the device.

Our challenges included:

1) *each MMBasic variable has several hundred bytes of overhead; we're guessing a fixed, relatively large amount of RAM is allocated to store the name of each variable. Just allocating a few integers (nominally 8 bytes each) can easily use up more than 1KB of RAM.*

Solutions: minimise the number of variables used; use arrays where possible (as the name only needs to be stored once); specify a maximum length for all string variables; use local variables wherever possible so that the RAM is freed once we have finished with them; combine multiple flags into a single integer variable; pack configuration data into strings; refactor code to use fewer local variables; do not use constants (making the code messier, unfortunately).

2) *each level of MMBasic function or subroutine recursion uses around 1KB RAM. Therefore, just a few levels of call depth can exhaust available RAM.*

Solutions: "flatten" functions, ie, when a subroutine or function is only called from one place, integrate it into the "parent" - this makes the code harder to work with and read but it uses less RAM; use CFUNCTIONs where this can't be avoided, especially for code that must be called in deeply recursed subroutines, as they have much lower stack and variable overhead.

3) *complex string pasting expressions allocate many temporary strings, which can easily add up to several kilobytes.*

Solutions: split up such complex expressions, placing temporary strings into local variables with limited size to reduce RAM usage; perform complex string processing in CFUNCTIONS which don't have this limitation.

4) *fonts and CFUNCTIONs use up a lot of flash.*

Solutions: use a minimal number of fonts (two, plus the built-in font); place all fonts in the LIBRARY section where they are compressed; also place as many CFUNCTIONs as possible in the LIBRARY section (one of the two will fit).

5) *the program is too large to fit in flash.*

Solutions: place as many extra function as possible in the LIBRARY section, where they are compressed; re-factor code to reduce repetition and take advantage of subroutines, recursion and loops (possibly increasing RAM usage); use the MMEdit "crunch" feature which strips out unnecessary spaces, comments, etc from the program when uploading to the Micromite; use shorter variable and subroutine names; refactor code to use more compact expressions which perform the same operation; remove any unused or redundant code; hard-code display dimensions.

```
LOCAL INTEGER count, t
LOCAL v, a, pf
LOCAL temp$(8) LENGTH 18
FOR count = 1 TO num_datum-1
  t = (count-1) * log_interval
  v = get_datum(count, "v")
  a = get_datum(count, "a")
  pf = get_datum(count, "pf")
  temp$(1) = STR$(count)
  temp$(2) = STR$(t)
  temp$(3) = duration_str$(t)
  temp$(4) = STR$(v,0,1)
  temp$(5) = STR$(a,0,3)
  temp$(6) = STR$(v*a,0,1)
  temp$(7) = STR$(v*a*pf,0,1)
  temp$(8) = STR$(pf,1,2)
  print temp$(1)+","+temp$(2)+","+temp$(3)+","+temp$(4)+"
  ,"+temp$(5)+ ,"+temp$(6)+ ,"+temp$(7)+ ,"+temp$(8)
NEXT count
```

**This code snippet from the logging output portion of the code shows how using string arrays with fixed length can be used to paste multiple values together with lower memory overhead than simply using a single, long expression.**

```
long long int main(const char* date, const char* time, const
char* tariff_times, const char* holidays) {
  unsigned int i, day, mon, year, hour, min, dow, offset;
  day = bcd2_to_int(date+1);
  mon = bcd2_to_int(date+1+3);
  year = bcd2_to_int(date+1+8);
  if( dow > 0 && dow < 6 ) {
    for( i = 0; i < 22; ++i ) {
      int holday, holmon, holyear;
      holday = bcd2_to_int(holidays+1 + i*6);
      holmon = bcd2_to_int(holidays+1 + i*6 + 2);
      holyear = bcd2_to_int(holidays+1 + i*6 + 4);
      if( holday == day && holmon == mon && holyear == year)
      {// it's a public holiday, woohoo
        dow = 0;
        break;
      }
    }
  }
...
```

**This partial CFUNCTION shows how the lower function call overhead and ability to pass pointers into strings eliminates the memory associated with temporary sub-strings.**

the brightness adjustment pot towards the edge of the lid.

If you're using the laser-cut lid, you should find that the display fits snugly through the provided cut-out although you may need to keep the screws loose initially in order to line it up. If using a self-cut lid and it doesn't fit first time, you will have to remove the display and do some filing.

Once it's secured in place, you can attach the ribbon cable as shown in Figs.5 & 6. Again, be careful to ensure that the pins on the BackPack are properly aligned with its IDC header and check for GND continuity.

Now would be a good time to attach a label to the lid. Artwork can be downloaded from the SILICON CHIP website. You have various options for producing the label:

1) print it onto plain or photo paper, then laminate it and either glue it to the lid (eg, using silicone sealant) or attach it using thin double-sided tape.

2) mirror it and print it onto overhead transparency film, then attach it to the lid using a thin smear of silicone sealant.

3) use Datapol/Dataflex printable labels (to suit laser printers or inkjet printers respectively).

Regardless of which method you use, cut out the holes for the LCD and mounting screws using a sharp hobby knife before affixing the label to the lid. One advantage of attaching a lid label is that it will cover the non-viewable area of the TFT, for a neater appearance. But since pretty much all interaction is done via the touchscreen, a label is not mandatory.

Before attaching the lid to the box, re-check the mains wiring, especially the wires going into CON8 and make sure that there are no stray strands of copper wire that could short to anything else and that all the connections are secure. Then fold the ribbon cable under the BackPack and attach the lid to the box using four black self-tapping screws.

## More testing

Now for the real test. Make sure nothing is plugged into the socket end of the mains cable and the lid is securely attached, with no loose wires. Place a fuse in the fuse holder; you can use a 1A fast-blow fuse for now and replace it with a 10A slow-blow fuse as specified later, so that it will blow faster in the unlikely event of a fault.

Place the unit in a secure location where it won't fall off under the weight of the mains cables or be knocked off, then plug it into mains and switch it on. The LCD backlight should be illuminated immediately and display should come up soon afterwards (you already tested this earlier, so all we are really testing here is the mains power supply).

Verify that the voltage reading is reasonable, ie, around 230VAC but keep in mind that you haven't calibrated it yet. The current, VA and power readings are not going to be zero for the same reason but they should drop to a low level after about ten seconds (less than 100mA, less than 10VA and under 5W). If not, that suggests something is wrong so switch off, unplug the unit and check for faults, especially bad solder joints.

## Next month

To conclude the Touchscreen Appliance Energy Meter, next month we'll go over the calibration procedure and give more information on how to use the unit. We'll also give some details on the CFUNCTIONs we used to augment the BASIC code and provide the required functions for the meter to perform well. SC

## Box drilling and cutting diagrams

Shown below are the holes and cutouts required in the UB1 Jiffy box. These diagrams are shown exactly half size, so if you enlarge them with a photocopier to 200% they can be used as drilling and cutting templates.

Alternatively, you can download them from www.siliconchip.com.au (see the downloads section) and print them at 100% to use them as templates. Colour front panel artwork can also be downloaded from this source.