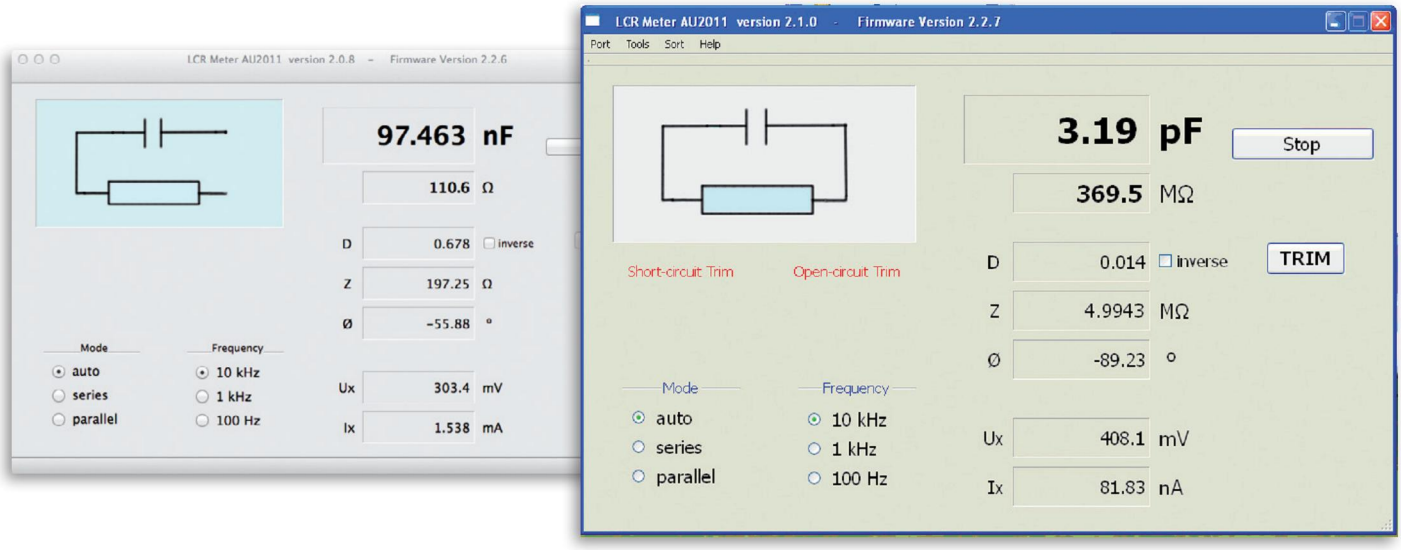
500 ppm LCR Meter (2)

Part 2: standalone use, functionality, menus, and software



By
Jean-Jacques Aubry
(France)

In this second article, you're going to be discovering:

- the display & keypad extension for using the device without a PC
- the device's functions via its menus and the display
- the software, or at least certain of its more noteworthy aspects, as it's impossible to go into all its intricacies here (the description, source code, and executables are available in full from our website [1]).

To get the most out of this Part 2 article, it's best to have digested Part 1 and have the circuit diagram of the device in front of you.

Due to lack of space in the first article [1], we haven't yet shown the LCR meter's second (optional) board, which turns it into a self-contained device. But it's not very big and we'll soon get through it.

Display & keypad extension

At the outset, this LCR meter only worked when connected to a PC. Opinion is divided about the restrictions that entails. With some in favor of a stand-alone unit and some who aren't put off by being dependent on a computer, to keep everyone happy I added the extension at the request of Elektor Labs. The task was (relatively) easy, as not only did the previous AC powered version of the device already use a 128×64 pixel graphic display, but also the code for managing this display and the associated mini-keypad was 90% reusable.

The reduced number of port lines available (5 instead of 13 in the earlier version) obliged me to abandon parallel drive for the display in favour of serial drive. In addition, I needed to be able to automatically detect the presence of the extension and the type of powering (via the PC's USB port or via an external supply).

The circuit diagram for the extension is modest (**Figure 1**). The 128×64 pixel liquid crystal graphic display U1 (Displaytech) is managed internally by a Sitronix ST7565R controller.

Capacitors C1–C9 are used to convert the LCD drive voltage. The built-in backlight is driven by the main board via transistor Q1; it will be activated only in self-contained mode. The state of pin 6 on J16 (LCD_RES/) indicates the presence (0) or absence (1) of the extension.

LED D5, driven by transistor Q2, is just a clone of D6 on the main PCB: it flashes at the end of each measuring sequence.
The mini keypad comprises the five buttons K1–K5 and the matrixing diodes D1–D4. Three port lines offer eight different codes, of which seven are usable if we exclude the code 111 (= no button pressed). That's not many, but the LCR meter software will differentiate between a short and long press, which will increase the possible combinations.
The connection to the main PCB (via J16) is made using ribbon cable with a female connector on this end, but soldered at the extension end with J1, which is a transition connector soldered into the PCB, for reduced vertical clearance.
The functions of the buttons, which we'll come back to in detail later, are:

	Short press	Long press
K1	**S/P/A**	**Menus**
K2	**Param.**	**Q-D**
K3	**Freq-**	**Sorting**
K4		**RT**
K5	**Freq+**	**Trim**

So that's about it for the description of the hardware. Now it's time to discover how the LCD meter works, whether or not it is associated with a computer. The procedures for setting up and alignment are covered in two additional copiously-illustrated documents that you can download from the Elektor website [3]. We're not going to go here into details of all the adjustments and menus, but just say enough to make readers want to take the plunge and launch into building this unrivalled device.

PC mode

If the measuring head (for which the circuit diagram was published last month in Elektor) is connected to a computer via a USB link, PC mode is selected by default. The interface of the AU2011 program [1] installed on the host PC is in English by default, but supports translation files.

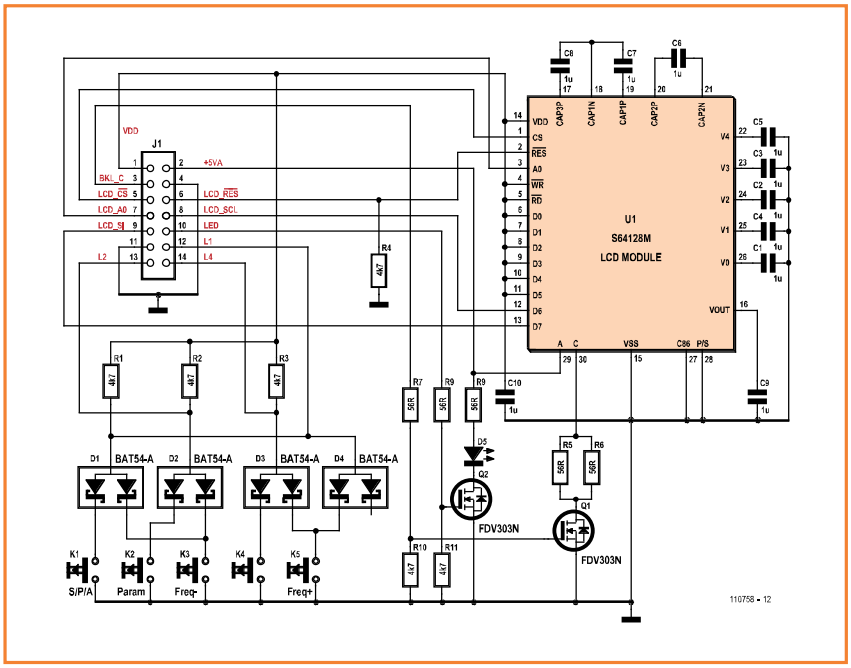


Figure 1.
The extension board includes five matrixed buttons and a graphics display.

As the device is fitted with the FTDI USB-UART communication IC (FT232R), we need to install the driver for it. This operation is described in the LCR meter | Set-up document where the whole calibration procedure is given in careful detail.

Running the program and menus

Connect the LCR meter to the PC, run the user program on the PC and power up the measuring bridge, then follow the instructions in the set-up guide. Then come the **Preferences, Tools, Sort** menus (**Figure 2**), which we're not going to discuss in detail here. Let's instead take a closer look at the **User Interface** (**Figure 3**). The program's main window displays the measurement results, and a number of buttons let you modify certain parameters or initiate certain actions.

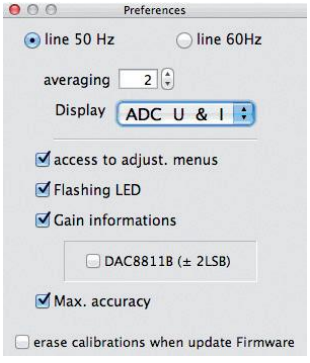


Figure 2a.
You have to start by certain inescapable choices in the Preferences.

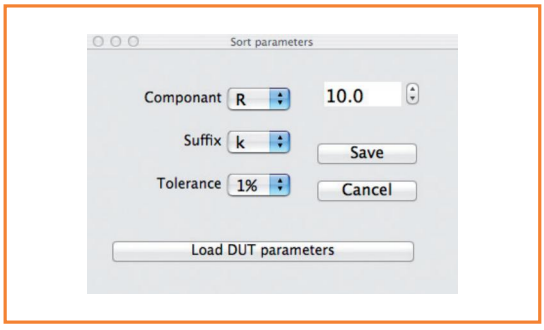


Figure 2b.
The Sort function requires prior configuration.

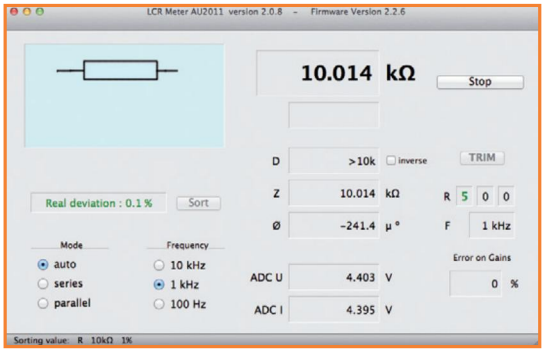


Figure 3.
The main window, here while sorting a resistor (set value displayed at bottom left).

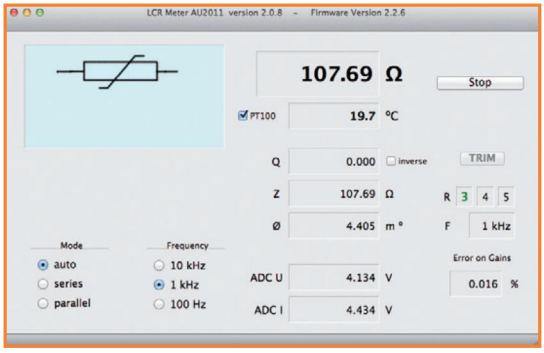


Figure 4.
A platinum probe allows accurate temperature measurement over a range from –80 °C to +600 °C.

Figure 5.
The buttons and display in standalone mode.

At the top left, a pictogram shows the equivalent circuit of the component under test.
When the mouse pointer passes over a display field, a bubble appears giving information about what is displayed in this field.
At first time use, the **Open the port** button starts the procedure for connecting to the serial port defined (and memorized) in Preferences.
Once the connection has been established, the button is no longer active. From then on, the procedure is automatic and the button is not visible any longer.
As its name indicates, the **Start** or **Stop** button starts or stops the measurements.

TRIM button

Runs the OPEN – SHORT compensation.
Compensation, which involves measuring the parameters of the measuring device itself (leads, clips), is carried out for each frequency. In the event of an error, the value is not used. A red symbol indicates when compensation has not yet been performed.

The **Sort** button starts the sort procedure. The reference value appears on the left in the strip across the bottom of the main window. The actual difference is displayed in green if the component tested is within tolerance or in red if it is not.

The **Mode** buttons let you select the mode for representing the component under test:
Auto: selection between series or parallel is performed automatically depending on the impedance of the component under test.
Series: forces series representation.
Parallel: forces parallel representation.
The test frequency is selected using the **Frequency** buttons. The lowest is twice the AC line frequency selected in Preferences.
The first **Gain** field indicates the measuring range:
in green for ranges 3–6 where the error due to the main amplifier is zero.
in magenta for ranges 2–7 (error due to the main amplifier ±0.02 %).
in red for ranges 1 and 8 (error due to the main amplifier ±0.04 %).
The other two fields give the values, between 0 and F, of the steps of the multiplying DAC for the *voltage* and *current* measurements.
If the **Max** check box is visible (here it is not), the gain error value is calculated using either typical values or using the Max value (for the PGA103), depending on whether it is checked or not.
The **Settings** menus let you perform a number of internal settings: offsets, calibrations, etc. The first three don't require you to take the lid off. Please refer to the document LCR meter | Set-up.

Measuring a PT100 platinum probe

There are lots of devices around offering a tem-

What could be more satisfying than building your own tools?

perature display, but only with an accuracy of several degrees! I thought it would be interesting to have a true reference available. Thus a platinum probe lets us measure temperature in a range from −80 °C to +600 °C with an accuracy of ±0.3 °C (class B) or ±0.1 °C (class A). When the value of a resistor lies between 70 Ω and 300 Ω, the frequency is ≤ 1 kHz, and the *Q* (quality factor) is practically zero, a box marked **PT100** appears to the left of the secondary parameter display field (**Figure 4**).

If you check this, the temperature value is shown for a PT100 platinum probe with this resistance. The symbol changes to a thermistor one. Within the measuring range used (−75 °C to +557 °C), the accuracy of the resistance/temperature conversion is < 0.05 °C.

Standalone mode

Fitted with the display & keypad extension, our LCR meter becomes a standalone unit. No need for a PC (except for performing the preliminary settings, which cannot be done in standalone mode). In this mode, all the texts displayed are in English.

The five buttons (**Figure 5**) let you select the device functions and configure certain parameters. The buttons have different functions depending on whether your press them quickly, or hold them in for more than two seconds (indicated by the green LED). The primary functions are in black (above the buttons on my prototype), while the secondary functions (long press) are in blue. In the menus, the (variable) functions of the buttons are indicated at the bottom of the liquid crystal display.

Primary functions

S/P/A: Choice of model between series or parallel equivalent circuits for calculating the parameters, or the automatic model mode.

Param.: Display options:

modulus of the impedance $|Z|$ and phase angle Φ.

equivalent series resistance R_s and equivalent series reactance X_s.

device's rms voltage V_x and rms current I_x.

Freq− & Freq+: To select measurement frequency.

Figure 6.
Sort function.

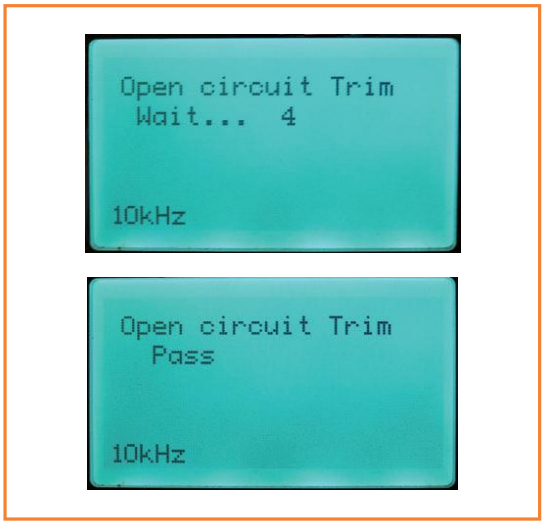


Figure 7.
Trim function.

Secondary functions

Menus: Hold this key pressed in to go into **Menu** mode.

Q_D: Holding this key in toggles the automatic display of **Q**uality → **D**issipation or **D**issipation → **Q**uality.

Sorting: You first have to define the sort parameters via the Sorting Parameters menu.

Holding this key in lets you enable or disable the **Sort** mode.

The value of the component being sorted is displayed with its symbol (**Figure 6**).

The sort parameters are displayed in reverse order.

Pressing the [GO] key starts the comparison, with momentary display of the actual tolerance and the result (OK or BAD!)

Trim: Pressing and holding this key lets you run the compensation (*OPEN – SHORT*) (**Figure 7**).

Compensations are performed for each frequency.

If $|Z|$ is < 10 Ω, the SHORT calibration is performed.

If $|Z|$ is > 100 kΩ, the OPEN calibration is performed.

If either or both of the calibrations is invalid, a flashing symbol appears in front of the primary parameter value. Re-do the incorrect calibration(s) (**Figure 8**).

Unmarked button: the function of this button is explained in the document *LCR meter | Operating Instructions* [3].

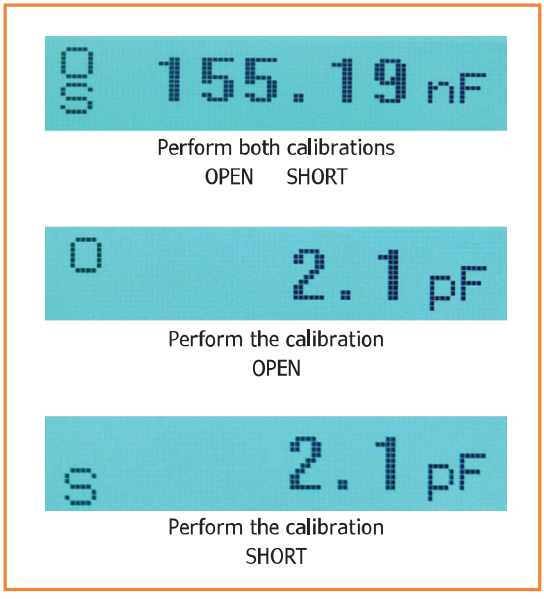


Figure 8.
Failed calibrations

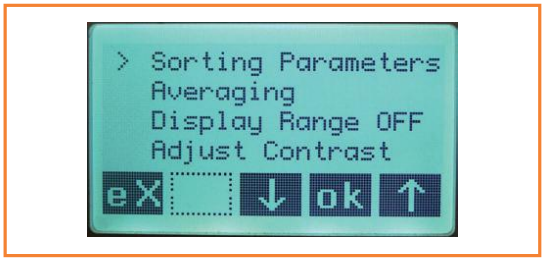


Figure 9.
The choice of menus.

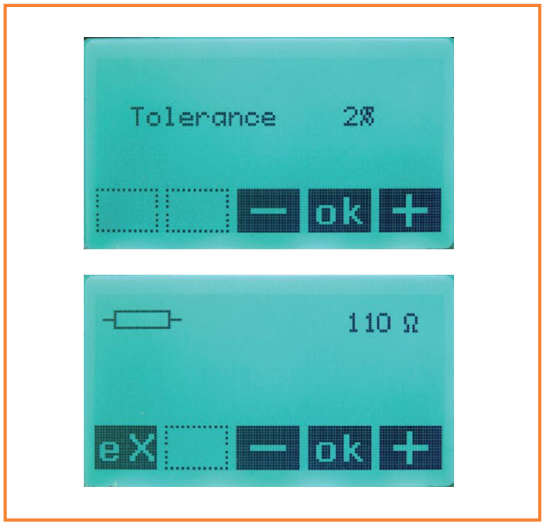


Figure 10.
Sort tolerance and primary sort parameter.

Menus (Figure 9)
Navigation is performed using the ↑ and ↓ keys. The **>** symbol indicates which menu will be selected by [ok].
Pressing the [eX] (*exit*) key takes you out of *menu* mode.

Sorting Parameters
This menu lets you select the parameters (tolerance and value) used for sorting components. The value proposed will be that of the primary parameter of the reference component connected ***before*** going into menu mode.
The first step lets you select the tolerance using the [−] and [+] keys, then confirming with [ok]. (**Figure 10**)
The next step lets you adjust the value that will be used as the sort value, starting off from the **primary parameter** of the component connected.
[−] and [+] keys, confirmed with [ok]. (**Figure 10**)

In the online operating instructions, you'll find other functions that we're not going to describe in detail here: *Averaging, Display Range, Adjust Contrast, Back Light, Line freq. 60/50 Hz.* See also the paragraph on **Measuring a PT100 platinum probe** in the PC mode section above — this function is also available in **standalone** mode.

Programs
The overall performance of the LCR meter (measurement accuracy, ease of use) is very much dependent on the quality of the three programs used:
the bootloader, which takes command as soon as the measuring bridge is powered up.
the firmware, the most important part of the internal program, which performs all the work of measurement acquisition, calculating the DFT, etc. It also drives the display in standalone mode.
the external AU2011 program running on the connected computer, which in "PC" mode makes it possible to display the results, run commands, etc. This dialogues with the bootloader and then the firmware by sending and receiving messages in the form of predefined character strings.

Bootloader
The bootloader, which runs automatically at power-up, is permanently resident in memory

from the address 0x0000. It's main function is to update the main program (firmware):
if jumper J17 is in place (unconditional update, e.g. in the event of the firmware's having crashed);
during a software reset, when the user requests the update;
if the integrity check on the main program code in EEPROM is negative.
The AU2011 program can only perform this update **if the device is in PC mode**.

The first step is to erase the program memory (all bytes take the value 0xFF), then the file in IntelHEX format is received over the USB-UART link. After checking that the operation has been correctly completed, the new firmware is run with the help of a function pointer to the firmware code start address:

```
((void (code *) (void)) PROG_BEGIN_FLASH_
ADDR) () // -> jump to application code
```

Listing # 1

```
void UART0_ISR(void) interrupt INTERRUPT_UART0
{
      char SFRPAGE_SAVE = SFRPAGE;
      SFRPAGE = UART0_PAGE;
      if (RI0 == 1)    // if receive flag set, put the byte in  UART_InputBuffer
      {
      RI0 = 0;
      Byte = SBUF0;   // Read a character from UART0
      if ((Byte == '\0') || (Byte == '\n') || (Byte == '\r'))
      {

      RX_Ready = 1;                    // Reception complete
      }
      else
      if (UART_InputBuffer_Size < UART_IN_BUFFERSIZE - 1)   // room needed for string terminal '\0'
      {

      UART_InputBuffer[UART_InputBuffer_Size] = Byte;

      UART_InputBuffer_Size++;         // Update array's size
      }
      }
      if (TI0 == 1)
      // if transmit flag set, send UART_OutputBuffer datas
      {
      TI0 = 0;
      if (UART_OutputBuffer_Size > 0)    // If buffer not empty
      {

      Byte = UART_OutputBuffer[UART_Output_First];

      SBUF0 = Byte;                    // Transmit to UART0

      UART_Output_First++;             // Update counter

      UART_OutputBuffer_Size--;        // Decrease array size
      }
      else
      {

      UART_Output_First = 0;

      TX_Ready = 1;                    // Transmission complete
      }
      }
      SFRPAGE = SFRPAGE_SAVE;    // restore SFRPAGE
}
```

The update part of the firmware comes directly from Silicon Labs application note AN112, but with a modification that is required for the checksum of the memory occupied by the newly-installed program. In point of fact, since a number of 'holes' (with value 0xFF) may exist in the newly-written memory space, this can lead to a difference between the actual size of the program and the calculation performed by end_address – write start_address!

For the other tasks performed by the bootloader, you'll need to refer to the full source code, which you can download from our website [1].

Firmware

This is the heart (and brain!) of the device. It resides in memory from address 0x2000. It carries out the measurements, responds to the user's commands, and either displays the results (in standalone mode) or sends them over the USB link (in PC mode).

Only a few small sections of the code are described here; the full and copiously-annotated source code of the firmware can be found on our website [1].

In PC mode, dialogue with the AU2011 program occurs by exchanging messages. The character strings received or sent are handled by an interrupt routine of the UART0.

One of the most worthwhile features of the microcontroller chosen is direct storage into XRAM (**D**irect **M**emory **A**ccess) of data coming from by ADC0 and ADC1. The DMA interface is programmable for, among other things, writing the data from an address in XRAM and for a certain number of acquisitions of these data. This pro-

Listing # 2

```
void TIMER2_ISR(void) interrupt INTERRUPT_TIMER2
{
      char SFRPAGE_SAVE = SFRPAGE;
     SFRPAGE = TMR2_PAGE;
     TF2 = 0;                          // Immediately reset Interrupt flag
      if (TMR2CF & 0x04)              // if state of the output = 1
      {

         ET2 = 0;                     // Disable Timer2 Interrupt
         SFRPAGE = DMA0_PAGE;         // Switch to DMA0 Page
         DMA0EN = 1;                  // Begin Executing DMA Ops (which will enable ADC0)
      }
      SFRPAGE = SFRPAGE_SAVE;
      // restore SFRPAGE
}
void DMA0_Acquire_Samples (void)
{
      char SFRPAGE_SAVE = SFRPAGE;
      // --- reset TF2 flag just prior enabling Timer2 interrupt
      SFRPAGE = TMR2_PAGE;
      TF2 = 0;
       ET2 = 1;                           // Enable Timer2 Interrupt
       SFRPAGE = DMA0_PAGE;               // Switch to DMA0 Page
      // --- Timer2 interrupt enable DMA-ADC operation
      while (!DMA0INT);                   // wait for DMA operations are complete
     DMA0EN = 0;                          // Stop Executing DMA Ops
     DMA0INT = 0;                         // Clear DMA0INT bit
     SFRPAGE = SFRPAGE_SAVE;              // restore SFRPAGE
}
```

Comparative measurements with a professional measuring bridge

By Ton Giesberts (Elektor Labs)

Measuring inductances is a science in itself. Inductor data sheets often give the current and frequency at which the inductance value is specified. For example, if you measure at 1 kHz instead of 10 kHz as given by the manufacturer and with a different measuring current, you will obtain a noticeably different inductance value. Which is not necessarily a disaster, since chokes have a tolerance of ±20% anyway. In the Elektor labs, I compared the behaviour of our new LCR meter designed by Jean-Jacques Aubry with the one in a very big lab measuring device costing over $2,000 (£1,300): the Hameg Programmable LCR Bridge 8118. The measuring current in this reference device is adjustable; during an initial measurement, its current was ten times that of our little device. Depending on the material of the inductor, this may lead to considerable differences. But we remained comfortably within the range of ±20 %.

A first measurement made on the *Hameg* for a 100 µH/5 A choke gives 108.7 µH. Reducing the measuring current (and for conscience's sake performing a new calibration), the same device then indicated 97.6 µH. Now you're bound to be wondering what the new Elektor LCR Meter was reading: 96.8 µH. Not bad, eh?

gramming is carried out once and for all by a few instructions in certain of the registers. All that then remains is to send the start command and wait for the end signal: all the work of acquiring data and transferring them into memory is automatic and superbly optimised, making it possible to achieve an acquisition speed of 1 mega-acquisitions per second (16-bit data, i.e. two 8-bit words) using a 24 MHz clock!

This process is organized by the interrupts requested by:

- Timer2 (frequency of the sinewave signal) → starts a sequence of N acquisitions by setting the DMA0ENable flag to 1.
- N = number of samples per cycle of Timer2 × number of cycles of Timer2.
- Timer3 (frequency of Timer2 × number of samples per cycle of Timer2 / 2) → automatically starts an acquisition; the coefficient of 2 is due to the fact that there are two interrupts per cycle of Timer3.

The sequence end is indicated by the DMA0INT flag going to 1. All that remains is to read the XRAM and use the data.

We saw in the first article that it was necessary to compensate for the offsets in opamps U6 (input_offset) and U1 (sine_offset). As we have two 12-bit DACs available, this adjustment can be automated. We just need to unplug the measuring cables so the process is not upset by stray signals (among others, from the power lines).

As for all the device's other adjustments, these can only be performed in **PC mode** via the AU2011 program. They are described in the document LCR meter | Operating Instructions that you can download free and which we strongly recommend you to read [3].

AU2011 program

The AU2011 software runs on a personal computer. It is written in C++ and uses the Qt and qserialdevice/AbstractSerial libraries in order to obtain an executable for PC under Windows, Mac OSX, or Linux, depending on the compilation target. The source code and a number of executables are available from our website [1]. If you want to modify and/or compile the source code, the IDE to use is Qt Creator. Conditional compilation options (#ifdef) in the source code allow you to manage the various targets directly. The User Interface (UI) is very slightly different depending on the compilation target, because of the differences in the character fonts, among other things; thus each target has its own AU2011_mainwindow.ui file.

The core of the user interface is in English, but it is designed to be able to use translation files. The French version (AU2011_fr.qm) already exists. Placed in the same directory as the AU2011 program, together with the *qt_fr.qm* file (a translation file specific to Qt that is located in the *QtSDK/Desktop/Qt/4.8.0/gcc/translations* direc-

A major advantage of the microcontroller chosen is the storage of the data from ADC0 and ADC1 directly into XRAM (Direct Memory Access)

tory, along with those for other languages), this makes it possible to have all the messages and the UI in French **for a French system**.
To obtain the same thing for another **local** language, you need to:
either (in the AU2011_Projet.pro file) modify the directive TRANSLATIONS += AU2011_fr.ts to AU2011_xx.ts where xx is the symbol for the local language (de, da, cs, etc.) and recompile to produce this new file;
or else duplicate the AU2011_fr.ts file and rename the copy AU2011_xx.ts.
Then you need to use the *Linguist* program to translate all the strings, and then lastly run the command *lrelease* to produce the *AU2011_xx.qm* file.
Refer to the Qt *Linguist* documentation for more details [2].
Our readers hopefully will gradually offer new translation files!

Let's remember that the dialogue between the AU2011 software and the LCR meter takes place via the USB link, by emulating a serial link thanks to the driver offered by FTDI (115,200 baud, 8 bits, no parity bit, 1 stop bit, no flow control). The commands sent to the device or received from it are character strings defined in the *AU2011_mainwindow.h* header file [1].

After the communication port has been opened and the bootloader synchronization command has been sent, the software waits to receive the first character string containing the firmware version number; it displays this information in the window title bar. Then will come the string formed from the initialisation parameters:
power line frequency 50 or 60 Hz (**L5** or **L6**)
test frequency (**F1** or **F2** or **F3**) for 100/120 Hz or 1 kHz or 10 kHz
Trim Short performed or not (**S1** or **S0**)
Trim Open performed or not (**O1** or **O0**)
averaging value (**A1** to **A9**)

Then the firmware goes into its event loop and monitors the arrival of a command from the UART (interrupt routine). If a request to perform the measurements has been received, the resulting parameters are sent to the software in the form of a long string of characters formed using the values of:
"**Rs Xs Freq ranges Vpp Ipp ADC_Vpp ADC_Ipp**"
plus the character 'C' or 'Z' depending on whether the is component is capacitive or not.

When this string is received, read by the ***slotRead()*** interrupt routine, the ***ParseInputString()*** function breaks it down and the display functions ***Display_xxx()*** fill in the relevant fields. One important routine, among others, is *Convert_Value_to_String()*, which converts a floating numerical value into a character string with mantissa and exponent in the form of a standardized suffix, e.g. **12.05 nF** for a capacitor with a value of 1.205×10^{-8} (in farads).

You can almost smell the solder (at last!)

In the next issue, we'll at last be showing you the two boards, which are going to be available as ready-to-use tested modules. For significantly less than $275 / €200, you'll be able to get your hands on a marvel of accuracy that will never again leave your workbench. There's nothing more satisfying than building your own tools! Between now and then, we invite you to take a look at the online documentation, which will give you both an overall and a very detailed view of our automatic impedance measuring bridge.

(130022)

[1] Downloadable software (bootloader, firmware and main program):
www.elektor-magazine.com/110758

[2] Qt Linguist
http://goo.gl/fIYQh
[or]
http://qt-project.org/doc/qt-4.8/linguist-translators.html

[3] On-line documentation:
LCR meter | Set-up
LCR meter | Operating Instructions
www.elektor-magazine.com/110758