

# LoWR: A Low Resistance Measurement Instrument

This m $\Omega$  (milli Ohm) meter will measure 0 to 500 m $\Omega$ 's to within 4% accuracy. It features very good repeatability (0.2% of FSR) and allows accurate comparative measurements to be made between cable and trace resistances.

It uses two dual low DC offset op-amps (LM4562) and an LPC1114 uC. The output resistance is displayed on an LCD.

Andrew C. Russell

July 2019

[hifisonix.com](http://hifisonix.com)

## *Introduction or 'Why 500 mΩs Full Scale?'*

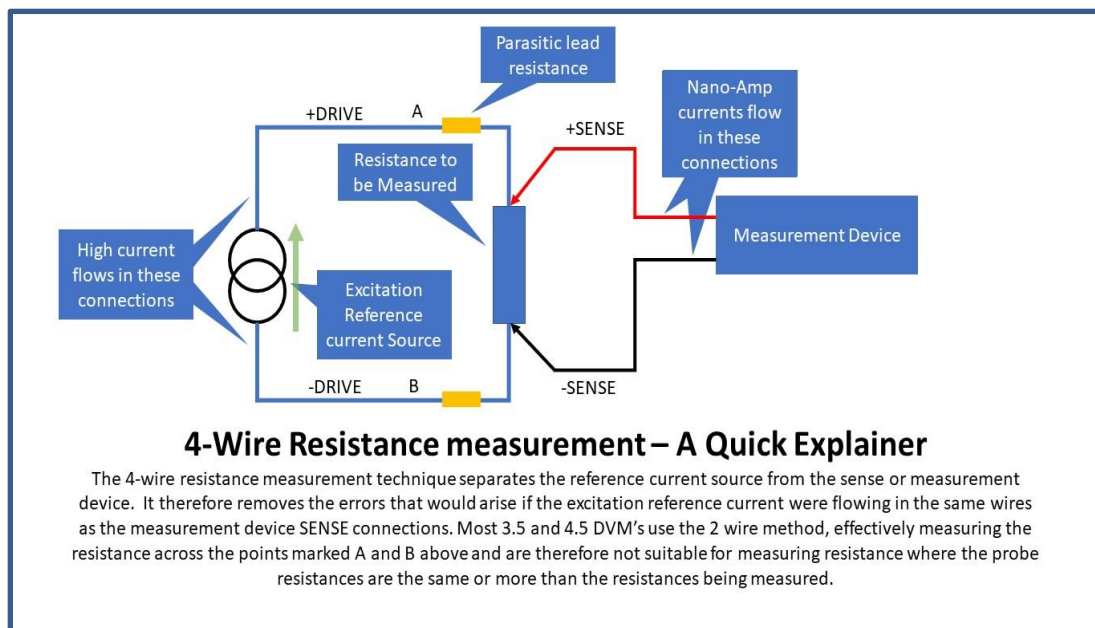
Trying to measure sub 1 Ω resistor values with a run-of-the-mill meter (3.5 or 4.5 digit) is usually an exercise in futility. The minimum FSR (full scale range) is 200 Ω's with a typical spec of 0.5 Ω's + some % inaccuracy of reading. If the resistance you are trying to measure is buried in the meter's (in)accuracy performance, there is little chance of a meaningful result. Secondly, if you want to measure resistance accurately, you would have to use the 4-wire technique (See the 'Quick Explainer' overleaf). Failure to do this means the lead resistance is added to the measurement result. Some meters allow you to calibrate out the lead resistance by shorting the leads together and then saving the measured resistance which is then automatically subtracted from subsequent readings, but they will still fail to deliver the resolution and repeatability needed to measure 0 to 500 mΩs to within 4% and 0.1 mΩ resolution which is what this instrument delivers.

One of the areas I wanted to explore, other than just accurately measuring low resistor values, was the trace and wiring resistances in some of the audio amplifiers that I build. It so happens that the resistance between the input grounds on each channel back to the central power supply has an important bearing on noise and cross channel isolation performance in unbalanced input amplifiers. The lower this resistance, the better and this device will allow this to be accurately ascertained. But, what about how low current charging and signal pulses distribute across the wiring and power supply? At (low) audio frequencies, the short cabling length inductances play little part, but the current flow difference between say a 10 mΩ and 50 mΩ branch is 5:1 i.e. huge if we are talking about capacitor charging currents for example. If we were able to accurately measure the branch resistances, we could calculate the current distribution and force currents to flow where and how we wanted them e.g. by the judicious selection of capacitor ESR and trace/wiring resistances. This simple instrument made with essentially 'junk box' parts will allow those types of ideas to be explored and questions to be answered.

## Specifications

**General Description.** A microprocessor based mΩ resistance meter using the 4-wire measurement technique allowing accurate resistance measurements over the 0-500 mΩ range to be made.

Range	0-500 mΩ
Accuracy	4% of full scale;
Repeatability	± 1mΩ on 0-500 mΩ range; 0.2% of FSR 0.1 mΩ on 0-50 mΩ range
Excitation Current	~100mA 10 ms pulse dynamically and calibrated every measurement cycle; ~1:11 duty cycle
Measurement cycle	Per measurement cycle 4 interleaved readings are each taken 200 times (total 800 readings) and the results averaged; each reading takes 2.5 us, total ~2ms per measurement cycle excluding settling time and display delays
Controls	Measure pushbutton and Range switch. Range Calibration push buttons located on PCB
Power Consumption	30 mA on ±7.5V supply rails; 90 mA with LCD backlight active



## Principle of Operation

Figure 1 below summarizes the operating principle of the meter. When a reading is initiated by depressing the 'measure' pushbutton, the  $\mu$ Controller turns the current source  $I_{ref}$  ON ( $\sim 100\text{mA}$ ), waits for 2ms for it and the amplifiers  $A_{vs}$  and  $A_{vz}$  to settle, and takes 4 readings in quick succession (typically about  $2.5\text{ }\mu\text{s}$  per reading – more about this a bit later), namely  $V1$ ,  $V2$ ,  $V3$  and  $V4$ . By measuring the reference current and ground resistance every measurement cycle, these errors are dynamically calibrated out every reading. This means that the current source need not be particularly stable despite the fact that we are building a reasonably accurate instrument and ground resistance changes, as they may exist, are also removed (more about this point a little later).

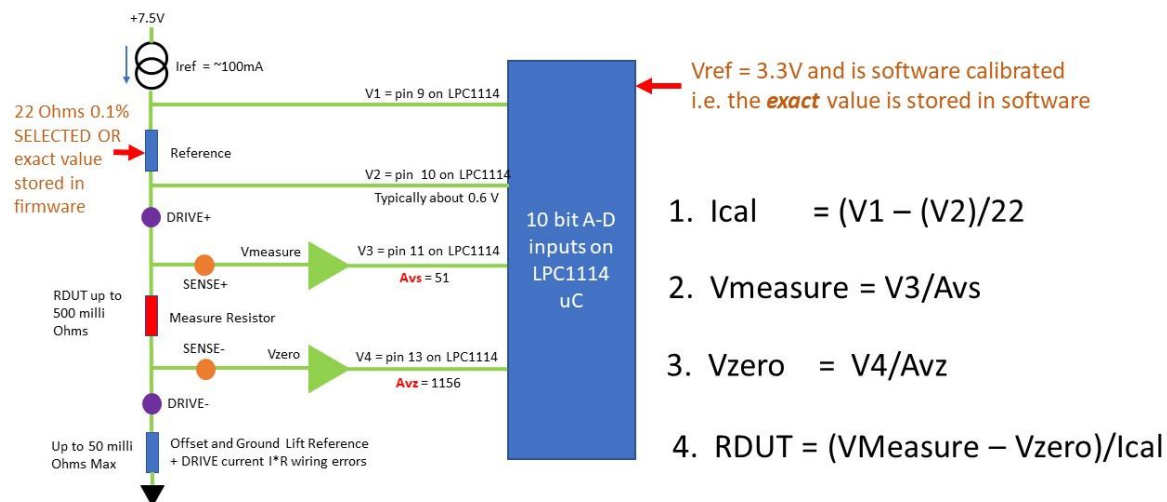


FIGURE 1 - OPERATING PRINCIPLE OF THE LOWR METER

The exact current is computed by subtracting  $V1$  from  $V2$  and dividing it by the accurately measured 22 Ohm resistor.

The value of the 22 Ohm resistor is selected to within 0.1%, or its exact value entered into the software at compile time as a constant (which is what I did).

Every time a measurement cycle is initiated, the calibrated current is stored as a local variable called  $I_{cal}$ .

Next,  $V_{zero}$  is calculated by dividing  $V4$  by the zero amplifier gain,  $A_{vz}$ , which is fixed at 1156.

Finally, we can calculate the value of RDUT by subtracting  $V_{zero}$  from  $V_{measure}$  and dividing by the calibrated current  $I_{cal}$ .

Each measurement cycle takes 5 ms and with display time and delays to limit the integral power dissipation, the total cycle time is in the region of 110ms – so a worst-case c. 1:20 duty cycle. The actual conversion time for each reading is about 2.5  $\mu$ s. When a measurement cycle is initiated, the four readings are taken in quick succession and the whole process repeated 200 times, with each of the 4 readings being summed so that the calculations are all done effectively with each reading  $\times 200$ . This acts to reduce the effect of rogue noisy readings and the result is remarkably stable readings despite an A-D resolution of only 10 bits in the presence of power supply and radiated noise.

Since the  $\mu$ Controller A-D resolution is only 10 bits, by amplifying the very small resistances – and thus sense voltages – before feeding them into the A-D, the resolution of the system is much improved over just feeding the resistor divider inputs directly into the A-D. On a 3.3V FSR with a 10 bit A-D, each count has a bit weight of 3.22mV or 32.2 m $\Omega$ s given we are measuring resistance with a 100mA current, which is far too coarse to be of any use. The amplifiers directly improve the effective bit weight by their respective gains and maximize the available measurement resolution. So, on  $V_{measure}$ , the 500 m $\Omega$  range bit weight is 32.2 m $\Omega$ s/51 or 631 micro Ohms.  $V_{zero}$ 's bit weight will change according to the magnitude of  $V_{zero}$ , but this cannot be more than 50 m $\Omega$ s otherwise an error is reported on the display, so the bit weight is no more than 92 micro Ohms.

A word about the offset and ground lift resistor. On my iteration of this instrument, I used two dual opamp [LM4562](#)'s because this is what I had available in my junk box. With much cheaper devices like the [TL072C](#) for example, the offset voltage can be as much as 13mV over temperature. If  $V_{zero}$  was < the worst case offset of the opamp and the offset was -ve,  $V_3$  would be < 0 and we would therefore not be able to calibrate the zero error out. The only way to guarantee this would not be the case would be to null the offset – this would require a pot - but this can be easily avoided by forcing  $V_{zero}$  to be positive by using a higher value offset and ground lift resistor. In the case of the TL072, this would require 130 m $\Omega$  (from 13mV/0.1Amps) which is too large a fraction of the desired measurement range, and limits the range of resistances we can measure. To avoid this problem, a low offset opamp is required and the LM4562 fits the bill perfectly. By using this offsetting and ground lift technique we can conveniently remove any opamp offset error and the wiring  $I \cdot R$  errors on the PCB through a simple software calibration cycle.

There is a second important issue here, and it is to do with the A-D zero offset error of  $\pm 3.5$  LSB's. Its true zero *may* lie below 0V (see Figure 4 and 5) such that a positive input voltage of up to 3.5 LSB's may be required *before* the A-D can detect any input and produce 1 LSB output code. Conversely, it may indicate up to 3.5 LSB's with 0V input. With the bit weight of 3.22 mV, the worst case A-D offset error is  $\pm 11.28$ mV. As in the case with the opamp offsets, we could circumvent this issue by ensuring that the -SENSE input is raised above 0V by at least a further 11.28mV. Again, this would require a 110 or 120 m $\Omega$  resistor which would eat up dynamic range of the  $A_{vz}$  amplifier and be > 2x the minimum measurement range of the instrument. Luckily, on this specific LPC1114, the zero error was nothing like  $\pm 3.5$  LSB's, so

I did not have to raise/lower the zero reference. To cater for the LM4562 worst case offset (0.7mV plus additional  $V_{os}$  drift over temperature), I used a value of 2mV ( $20\text{ m}\Omega \times 100\text{ mA}$ ). The  $20\text{ m}\Omega$  resistor is a 0.25W 1206 SMD device.

As this point, you may well ask why not use a higher resolution data converter like the TI [ADS1220](#) (about £8/US\$10)? Simply, because I had some old 28 pin DIL LPC1114 uC's in my junk box – and this project was all about using what I had.

## *Circuit description (Refer to Fig 2)*

The analog front end of the circuit is located on the RHS of the diagram. Q2 and Q3 form a standard two transistor feedback controlled current source with the output current defined by the  $V_{be}$  of Q3 and the 3  $18\text{ Ohm}$  (culled from the junk box) sense resistors (R4, R17 and R18) wired in parallel to give  $6\text{ Ohms}$ , which defines the output current at Q2's collector at  $\sim 100\text{ mA}$ . R22 ( $1\text{ k}$  in base of Q3) is included as a 'base stopper' as this type of configuration does have a tendency to occasionally oscillate without it. The current source is turned on by driving Q1 ON from the  $\mu$ Controller, U9, pin 1. The output current flows through R16 ( $22\text{ Ohm}$  reference resistor) out of U4 which is the +DRIVE output to the resistor under test and then back in through U8, the -DRIVE output and then to ground via R23 ( $20\text{ m}\Omega$  resistor – the 'offset and ground lift reference' in Figure 1). The voltage drop across the RDUT is picked off by the high impedance +SENSE and -SENSE inputs. D2 and D3 and D8 ensure that the inputs to the measurement opamp are clamped to safe levels and remain within the supply rails which are  $\pm 7.5\text{ V}$ .

The Zero amplifier, Avz in Figure 1, is configured around U3. Rather than a single very high gain stage, I elected to split the gain of 1156 ( $34 \times 34$ ) over two stages. C11 ( $330\text{ pF}$ ) and C13 provide some HF roll off on both gain stages. The output of the Avz amplifier is fed into the uC (U9) A-D channel 5 on pin 13 via R7 ( $10\text{ k}$ ) which provides protection to the uC when the amplifier output falls outside its supply rails ( $0\text{ V}$  and  $+3.3\text{ V}$ ). All of the A-D inputs are fed like this, with each input decoupled to ground via a  $10\text{ nF}$  capacitor – this arrangement provides some HF filtering above  $1.6\text{ kHz}$  ( $-3\text{ dB}$ ).

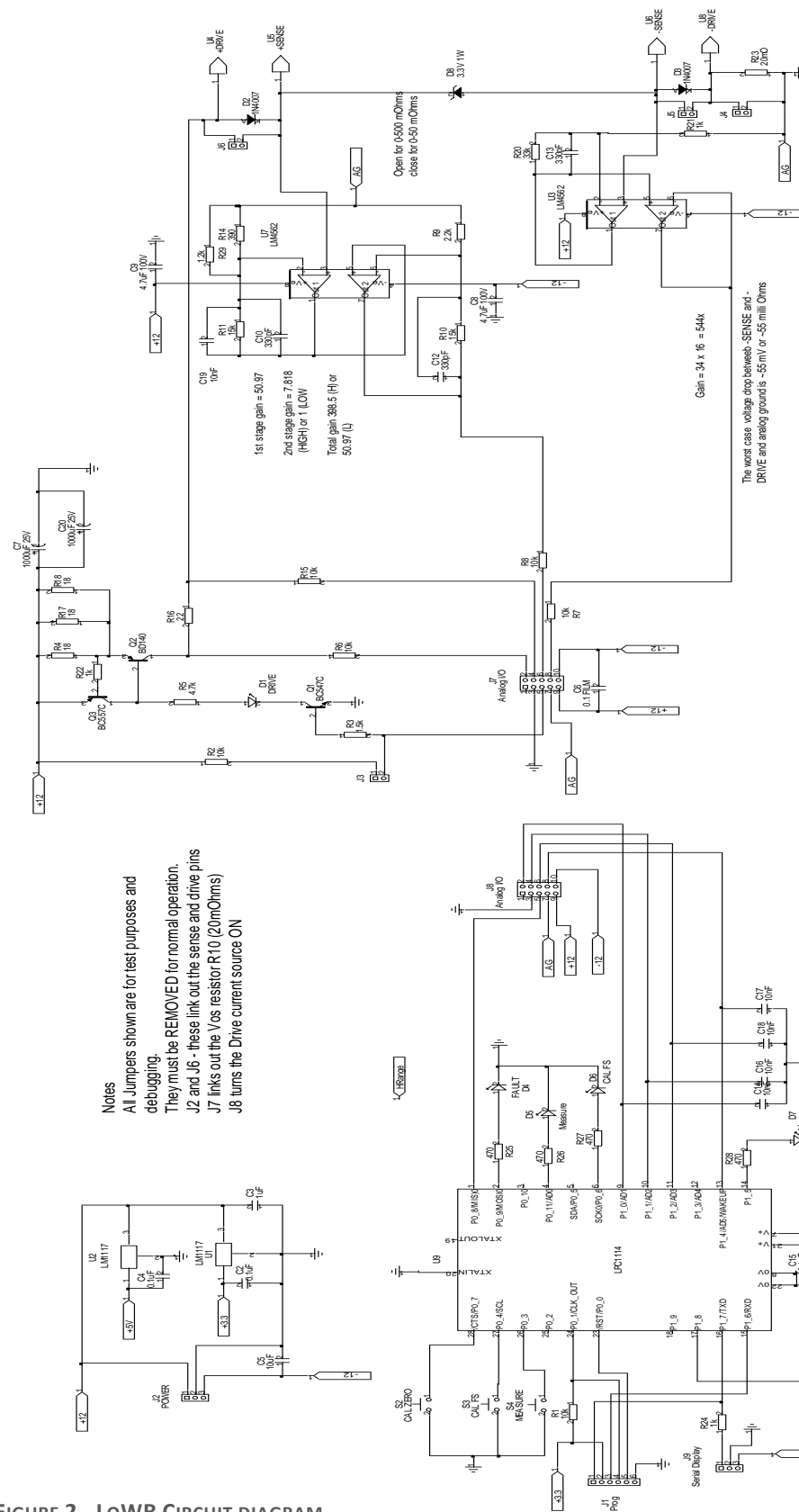


FIGURE 2 - LoWR CIRCUIT DIAGRAM

The sense amplifier ( $A_{vs}$  in Figure 1) is configured around U7, another dual opamp. The first stage (U7/1) provides a gain of 51 and the second stage (U7/2) a gain of 1 – i.e. just buffered. As in the  $A_{vz}$  amplifier, C10 (330pF) provides some bandwidth limiting.

The analog section rails are decoupled with 4.7uF 100 V electrolytic capacitors (again, culled from the junk box), while C6 (0.1uF 63V film) provides cross rail decoupling. C7 and C20 (1000 uF each) provide a local charge reservoir and decoupling for the current source.

The analog board is connected to the uC board via a 15 cm (6”) ribbon cable (J7 to J8).

The analog inputs, as discussed in Figure 1 are fed directly into the  $\mu$ Controllers A-D ports.

A single pushbutton, ‘Measure’ is fed into pin 27 (P0\_4) of the  $\mu$ Controller.

The operating mode of the instrument is conveyed by means of 2 LED’s – D4 and D5 representing Fault and Measure respectively. The ‘Measure’ LED will illuminate for about a quarter second while a reading is in progress while the fault LED will remain ON so long as there is an over-range condition – ‘Error’ will also be displayed on the LCD as well. The Fault LED will extinguish when the next reading without an overrange condition is made.

The ‘Cal Zero’ pushbutton allows just the total connection resistance between the RDUT (i.e. resistor under measurement) and the instrument to be tested. It is not used in normal operation and the associated pushbutton is not brought out to the front panel.

J1 (Prog) is the programmer interface that allows the binary code from the compiler to be written to the devices on-board flash memory.

During development, I simply dumped the raw A-D (after scaling and adjusting for the various amplifier gains) to the PC terminal. In the final instrument, I used a [Sparkfun ADM1602U 5V serial input LCD display](#). This was the most expensive item at c. £26 (available in the USA for US\$22), but it saved me the time and effort of having to write my own LCD driver had I used the very common and low cost 4 bit Hitachi standard interface. The display is fitted with a backlight (60 mA when ON) which can be illuminated but to save power, I do not use this.



Power is provided by a RS components 9-0-9 VAC [3.2 VA transformer](#) (c. £4/\$6 from RS components) as shown in Figure 3. After rectification and regulation using LM317's the  $\pm 7.5V$  rails are fed to the  $\mu$ Controller board, where a 5V regulator (U2) provides power for the LCD display, and U1 the 3.3 volts required for the  $\mu$ Controller.

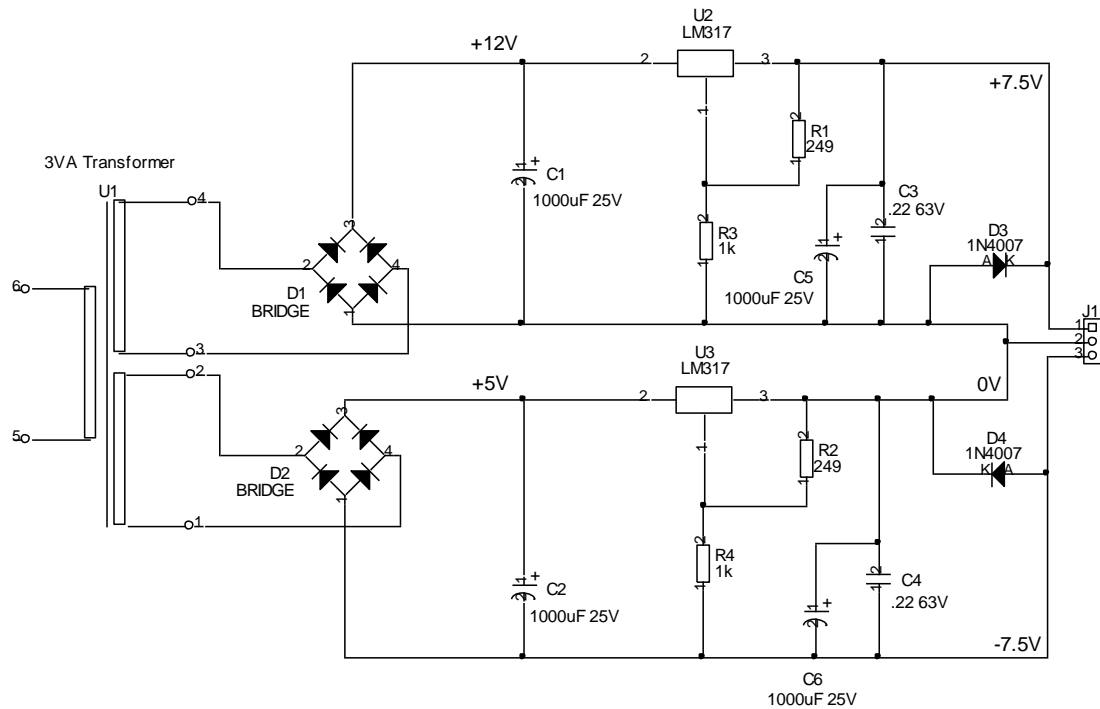


FIGURE 3 - LoWR POWER SUPPLY

## Calibration and Accuracy Discussion

The main purpose of this instrument is to measure  $m\Omega$  resistance levels ( $0\sim 500\ m\Omega$ ) with a resolution that is high enough to enable accurate determination of typical PCB trace and wiring/cable loom resistances and for fault-finding and investigation into high current LF current flow distribution.

As such, it is not designed to deliver laboratory grade resolution and accuracy. Aside from the A-D's absolute performance, the two parameters that have the largest bearing on the LoWR's performance are the A-D reference – in effect the  $\mu$ Controller's supply voltage, and the 22 Ohm current sense reference resistor R16. For the A-D reference, I simply measured the supply voltage about 30 minutes after the unit was powered up, and then entered this into the software as a constant (see the software listing in Addendum 1 – Vref). The 3.3V supply was surprisingly stable over time and temperature (note that the instrument runs cool – so the only real temperature effects are ambient after a few minutes warm-up). I did the same for the 22 Ohm resistor, measuring it accurately with a 4.5 DVM and entering it as a constant called Rref into the software at compile time.

The amplifier gains are used to 'back calculate' (i.e. divide the amplifier output voltage by the exact gain to arrive at the exact input voltage) the input voltages from the Azs and Avz amplifiers. The gain setting resistors are all 1% so the worst-case error in both the Avz and Avs is 2% and if they happen to be in exactly the opposite direction in each amplifier, about 1.5x that (because the Avz and Avs gains are different). But, whatever the absolute accuracy of the amplifiers are, the resistances being measured will all have the same errors so it will not affect the *resolution of differences in resistance*, only *absolute accuracy* and in that case as long as it is within a few percentage points, we are good to go.

Figures 4 and 5 overleaf are snapshots from the LPC1114 data sheet.

### 10.3 ADC static characteristics

Table 18. ADC static characteristics

$T_{amb} = -40\text{ }^{\circ}\text{C}$  to  $+105\text{ }^{\circ}\text{C}$  unless otherwise specified; ADC frequency 4.5 MHz,  $V_{DD} = 2.5\text{ V}$  to  $3.6\text{ V}$ .

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IA}$	analog input voltage		0	-	$V_{DD}$	V
$C_{IA}$	analog input capacitance		-	-	1	pF
$E_D$	differential linearity error	[1][2]	-	-	$\pm 1$	LSB
$E_{L(nonl)}$	integral non-linearity	[3]	-	-	$\pm 1.5$	LSB
$E_O$	offset error	[4]	-	-	$\pm 3.5$	LSB
$E_G$	gain error	[5]	-	-	0.6	%
$E_T$	absolute error	[6]	-	-	$\pm 4$	LSB
$R_{vsi}$	voltage source interface resistance		-	-	40	k $\Omega$
$R_i$	input resistance	[7][8]	-	-	2.5	M $\Omega$

[1] The ADC is monotonic, there are no missing codes.

[2] The differential linearity error ( $E_D$ ) is the difference between the actual step width and the ideal step width. See [Figure 17](#).

[3] The integral non-linearity ( $E_{L(nonl)}$ ) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset errors. See [Figure 17](#).

[4] The offset error ( $E_O$ ) is the absolute difference between the straight line which fits the actual curve and the straight line which fits the ideal curve. See [Figure 17](#).

[5] The gain error ( $E_G$ ) is the relative difference in percent between the straight line fitting the actual transfer curve after removing offset error, and the straight line which fits the ideal transfer curve. See [Figure 17](#).

[6] The absolute error ( $E_T$ ) is the maximum difference between the center of the steps of the actual transfer curve of the non-calibrated ADC and the ideal transfer curve. See [Figure 17](#).

[7]  $T_{amb} = 25\text{ }^{\circ}\text{C}$ ; maximum sampling frequency  $f_s = 400\text{ kSamples/s}$  and analog input capacitance  $C_{IA} = 1\text{ pF}$ .

[8] Input resistance  $R_i$  depends on the sampling frequency  $f_s$ :  $R_i = 1 / (f_s \times C_{IA})$ .

FIGURE 4 - LPC1114 DATA SHEET (1)

NXP Semiconductors

**LPC1110/11/12/13/14/15**

32-bit ARM Cortex-M0 microcontroller

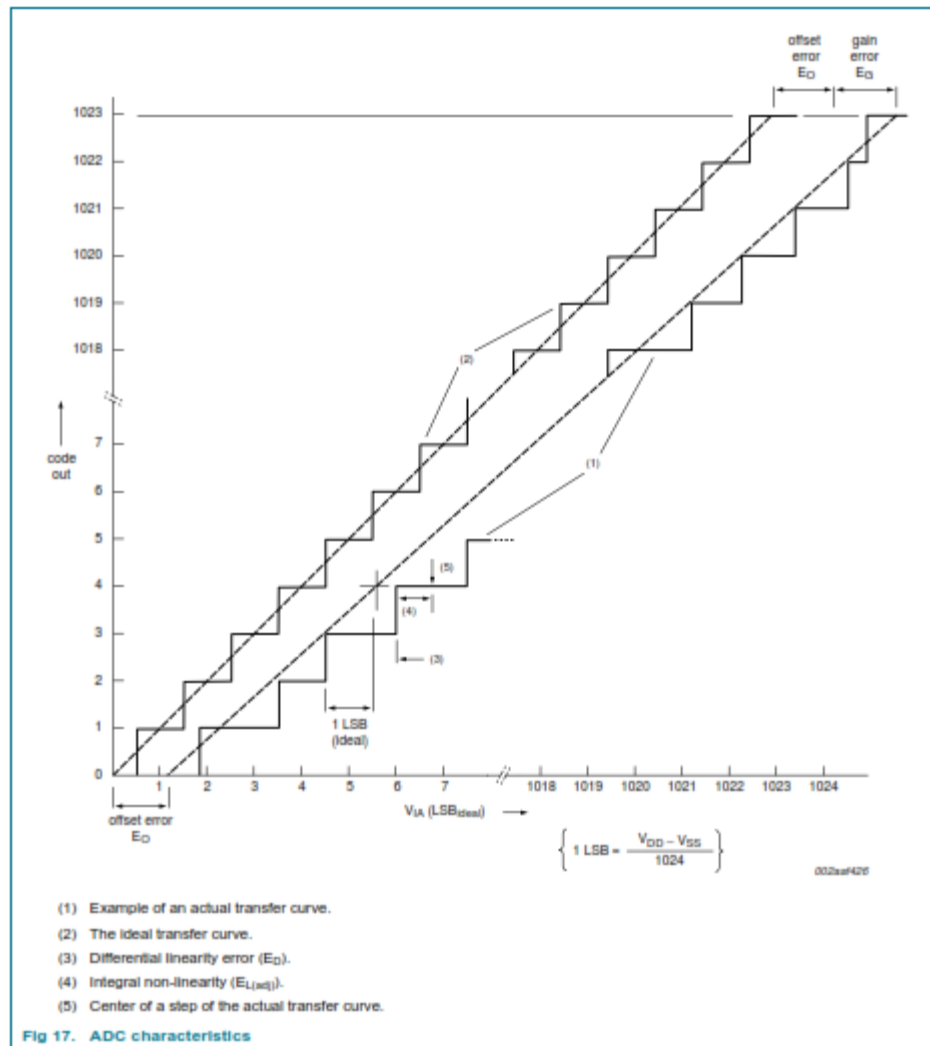


FIGURE 5 - LPC1114 DATA SHEET (2)

## *µController Platform and Firmware Development – a few comments*

I have been using ARM's 32bit µController devices for about 10 years now, starting out a few years after ARM launched its [mbed development platform](#). The IDE is easy to use and there are a host of [development boards](#) that can be used to try code out with – and many will plug directly into Arduino base boards.

If you are used to using 8 bit devices like the Atmel's or the PIC's, as good as they are, they are no match for the speed and power of an ARM µController (or PIC's 32 bit devices for that matter) which also include a host of peripherals (I2C, CAN Bus, SPI, A-D, D-A, Ethernet, Bluetooth etc). Often on 8-bit devices, to save memory or speed things up, look-up tables are used and a shortage of interrupt sources means you have to fiddle around with code and hardware on multi-interrupt systems. On a 32-bit device, complex equations can be used directly and *almost all pins* on the ARM devices can be set as interrupt sources – it's quite normal to have 5~10 interrupts on a real time operating system and it therefore makes writing code for real-time applications a cinch – no need to poll pins or use any external interrupt monitoring circuits. Since the code executes >1000 times more quickly than most 8-bit devices, the power consumption can be very low as the device can be put into sleep mode when not active and the result often is that these 32 bit devices consume less power than 8 bit devices.

Software is written in C++ and all of the µController peripherals are driven by common high level library objects (written and maintained by the mbed platform team and community) *that are the same right across all of the devices* from NXP, the original supplier of GP ARM µControllers through Freescale, ST, Nordic, Silicon Labs and many others.

So, if you are a budding embedded programmer/developer, I would earnestly recommend you go the ARM 32 bit route and the best platform out there is the mbed one.

## Construction and Assembly

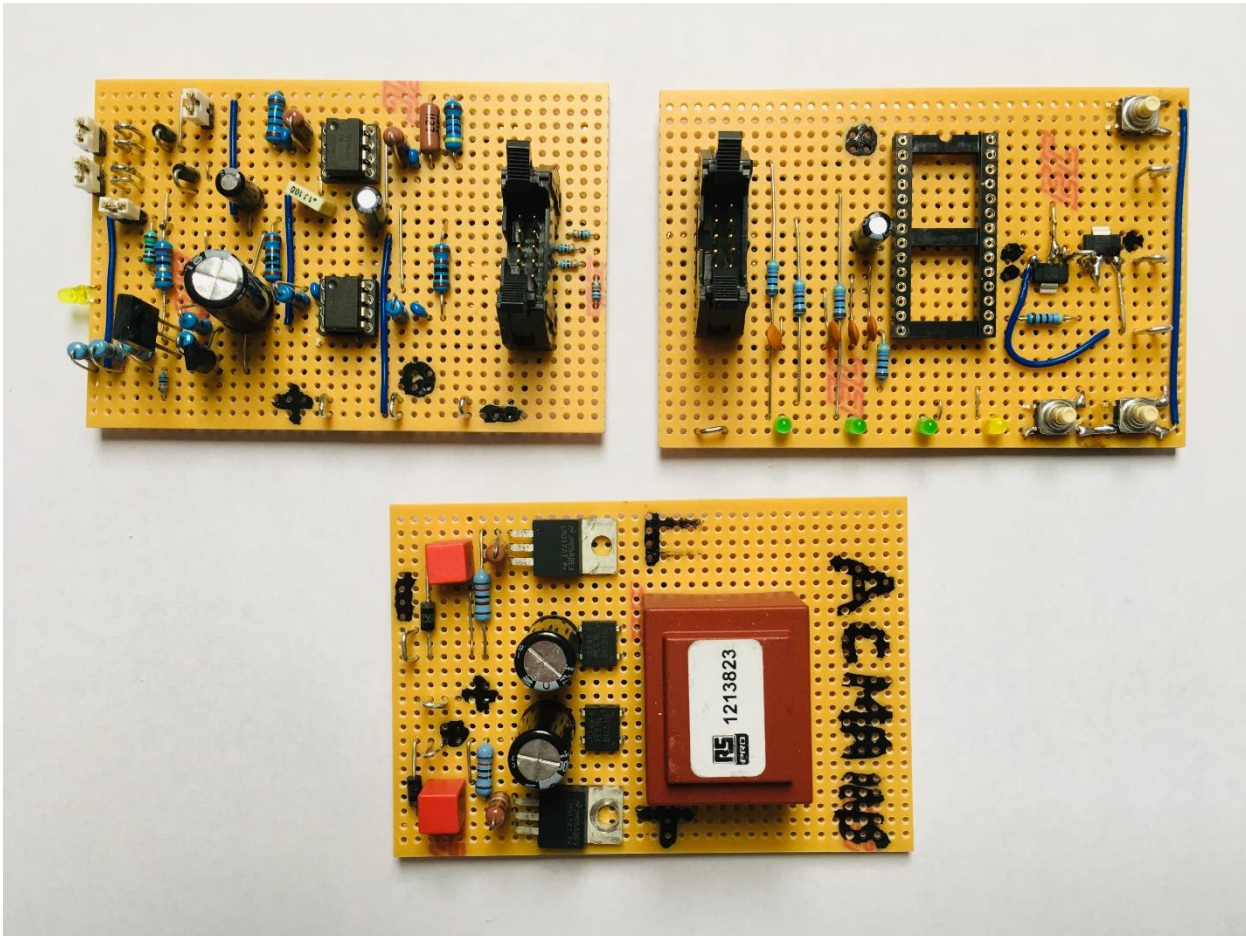


FIGURE 6 - LoWR BOARDS. TOP LEFT - ANALOG INPUT BOARD, TOP RIGHT  $\mu$ CONTROLLER BOARD, BOTTOM PSU

The complete instrument is assembled on 3 perforated ('Veroboard') boards – the analog frontend, the  $\mu$ Controller, and finally the PSU. This type of assembly is time consuming, but it saves on the cost of PCB's, and it's quite possible to get a very neat finished product. The trick is to keep all the wires and component leads orthogonal and resist the temptation – at least on the top of the board – to go point to point – that should be reserved for the underside where it cannot be seen. I managed to do this for most of the wiring but succumbed to a few 'bends' and diagonal links towards the end as you can see from the picture above.

I housed the LoWR in a black plastic Hammond case measuring 76mm x 200mm x 280mm for about £18 (\$20 in the US). The PCB's were mounted on 5mm nylon standoffs.

## Addendum 1 – Software Listing

```

/***** LoWR *****/
/***** A test instrument for measuring resistances of up to 500 milli Ohms with 4% accuracy *****/
/***** copyright Andrew C Russell May 2019 *****/

#include "mbed.h"
#include "pindef_LoWR.h"
#include "math.h"
#include "AnalogIn.h"          // standard mbed supplied analog driver header file
#include "Serial.h"

#define TRUE 1
#define FALSE 0
#define HIGH 1
#define LOW 0

#define ZOVER_RANGE 15        // a zero calibration reading of over 15 mOhms will generate an error message
#define OVER_RANGE 550       // any reading above 550m Ohms this is an overrange error
#define lo_senseamp_gain 51   // high side amplifier gain
#define zero_amp_gain 1156    // gain of the zero amplifier
#define Rref 21.8             // enter the exact reference resistor value in ohms
#define ERROR 3

// display commands
#define command1 0xFE         // send a command preamble
#define command2 0x7C
#define clear_display 0x01
#define cursor_blink 0x0D
#define backlight_on 0x9D
#define backlight_off 0x80
#define cursor_home 0x02
#define newline 0xA8
#define count 200             // this is the number of averaged readings in the A-D measurement cycle

//This is the calibration constant and must be entered in manually
#define Vref 3.337            // A-D reference voltage

int FLAG1;
int FLAG2;
int FLAG3;
int error_flag;

float Iref;                  // reference current
float reffhigha;             // absolute (i.e. measured and scaled) A-D input voltages – high side ref Voltage

float reflowa;               // as above but low side voltage
float sensehigha;
float sensezeroa;
float Irefa;
float R_DUT;
float zerocal = 0;           // this is where we store the zero offset if we do a zerocal cycle
float RAW_R_DUT;
//float gain;
float error_test;
int average;                 // this is the loop counter for averaging the readings

void get_analog(void);
void clrsv(void);

/***** Clear LCD screen ready for input *****/
void clrsv(void)
{
    putc(command1,LCD);
    putc(clear_display,LCD);
    putc(command1,LCD);
    putc(cursor_home,LCD); }

```

```

/***** newline cursor return *****/

void nline (void)
{
    putc(command1, LCD);
    putc(cursor_home,LCD);          // put cursor at start of LCD text
    putc(command1, LCD);
    putc(newline, LCD);             // move cursor to new positon on 2nd line
}

/***** Interrupt Service routines *****/
void measure_resistor_isr(void)      // measure interrupt
{
    FLAG1 = TRUE;
}
void calibrate_fullscale_isr(void)    // calibrate full scale interrupt - not used in normal operation
{
    FLAG2 = TRUE;
}
void calibrate_zero_isr(void)        // calibrate zero interrupt
{
    FLAG3 = TRUE;
}

/***** analog Inputs *****/
// the analog inputs are all read simultaneously and the absolute voltages are calculated and scaled
void get_analog(void)
{
    int tempcount = 0;
    refhigha = 0;
    reflowa = 0;
    sensehigha = 0;
    sensezeroa = 0;

    do {
        refhigha = refhigha + (ref_Hi*Vref);          // read top of 22 ohm sense resistor
        reflowa = reflowa + (ref_Lo*Vref);            // read bottom of 22 ohm sense resistor
        sensehigha = sensehigha + ((sense_Hi*Vref)/lo_senseamp_gain); // top of Resistor being measured
        sensezeroa = sensezeroa + ((sense_Lo*Vref)/zero_amp_gain);    // bottom of resistor being measured

        tempcount = (tempcount+1);
    } while (tempcount < count);

    Iref = (refhigha-reflowa)/(Rref);
    R_DUT = 1000*((sensehigha-sensezeroa))/(Iref);    // x 1000 to convert reading to milli Ohms
}

/***** main *****/
int main(void)
{
    // turn all LED's OFF initially
    drive = LOW;
    fault_LED = LOW;
    calzero_LED = LOW;
    calfs_LED = LOW;
    measure_LED = LOW;
    Serial pc(USBTX, USBRX);
    __disable_irq();          // just to make sure we can set up correctly without problems
    // setup all the pins as required
    measure_PB.mode(PullUp);   // pin 26
    calfs_PB.mode(PullUp);     // pin 27 - must be pulled up externally
    calzero_PB.mode(PullUp);   // pin 26
    // hgain.mode(PullUp);     // when HIGH, high gain range is selected

    // setup the associated interrupts
    measure_PB.fall(&measure_resistor_isr);
    calfs_PB.fall(&calibrate_fullscale_isr); // trigger int on rising edge - go service it at rc5dat
    calzero_PB.fall(&calibrate_zero_isr);    // input from rotary encoder or input select
    // make sure the interrupts are disabled

    // display the spash
    putc(0xFE,LCD);

```



```

putc(0x01,LCD);
wait(2);
clr();
printf("LowR Ready");
nline();
printf("hifisonix.com");
__enable_irq();

LOOP: // start of the main loop
{
    __WFI();
    if (FLAG1 == TRUE) {
        __disable_irq();
        fault_LED = LOW;
        calfs_LED = LOW;
        measure_LED = HIGH;
        drive = HIGH;
        wait_ms(1);
        get_analog();
        wait_us(100);
        drive = LOW;
        R_DUT = (R_DUT - zerocal); // here is the measured resistance

        // check here for measurement errors = eg swapped leads, overrange etc
        if ((R_DUT < 0) || (sensezeroa > sensehigha) || (reflowa > reffhigha) || (reflowa == reffhigha)) {
            clr();
            fault_LED = HIGH;
            clr();
            printf("Measure Error");
        }

        else {
            clr();
            printf("R=%3.1f", R_DUT);
        }
        nline();
        if (R_DUT > 550) {
            printf("Over Range");
            fault_LED = HIGH;
        }
    }

    /// all done, now get ready for the next read cycle
    measure_LED = LOW;
    FLAG1 = FALSE;
    __enable_irq();
}

if (FLAG3 == TRUE) { // calibrate and remove the zero error with the leads shorted
    __disable_irq();
    calzero_LED = HIGH; // turn calibrate LED ON
    clr();
    printf("Short Leads");
    wait_ms(3000);
    drive = HIGH;
    wait_ms(1); //let the analog front end settle
    get_analog();
    wait_us(100);
    drive = LOW;
    zerocal = 1000*((sensehigha)-(sensezeroa))/(Iref); //milliOhms
    clr();
    if (zerocal > ZOVER_RANGE) {
        printf("Zero Error ");
        printf("Redo test ");
        calzero_LED = LOW;
        wait_ms(500);
        calzero_LED = HIGH;
        wait_ms(500);
        calzero_LED = LOW;
        wait_ms(500);
        calzero_LED = HIGH;
        wait_ms(500);
    }
    else {
        printf("Zerocal = %3.1f", zerocal);
    }
    wait_ms(100);
    calzero_LED = LOW;
}

```

```
        FLAG3 = FALSE;
        __enable_irq();
    }
}
wait_ms(100);
goto LOOP;
}
/ ***** end of main listing *****/
```

```
/****** LoWR Pin Assignments *****/

Serial LCD(dp16,NC);      // create serial port tx only to drive the LCD

DigitalOut drive(dp1);    // turn the 100mA DRIVE current ON
DigitalOut fault_LED(dp2); // Fault LED for overrange or missing sense connection
DigitalOut calzero_LED(dp4); // Calibrate Zero LED - illuminates when Zero is calibrated
DigitalOut calfs_LED(dp6); // Calibrate FS LED - illuminates when FS is calibrated

AnalogIn ref_Hi(dp9);     // 22 Ohm reference resistor high side
AnalogIn ref_Lo(dp10);    // 22 Ohm reference resistor low side
AnalogIn sense_Hi(dp11);  // High sense connection to resistor under measurement
AnalogIn sense_Lo(dp13);  // Low sense connection to the resistor under measurement

DigitalOut measure_LED(dp14); // measure LED is on during measurement and calibration cycles

InterruptIn measure_PB(dp26); // initiate measure cycle
InterruptIn calfs_PB(dp27);   // initiate full scale calibration cycle
InterruptIn calzero_PB(dp28); // initiate zero calibration cycle

/* all unused pins left open */
```