"LC_meter_USB" - Multifunction Expandable Instrument controlled from PC trough USB interface

Keywords: Multifunction Expandable Instrument, LC meter, inductance, capacitance, L, C, USB interface, Microsoft Visual Studio Express, Visual Basic, ATtiny45, AVR Studio, Virtual USB, V-USB, LC_meter_USB

1. Summary

This LC_meter_USB is a universal PC controlled instrument. It is based on an alternative project named “Very Accurate LC Meter based on PIC16F84A IC”. The firmware is controlled from PC and is extended with further useful functions.

The currently implemented applications are:

- Inductance and capacitance meter,
- Frequency meter,
- Digital voltage meter,
- Frequency generator

The firmware is powered from the USB bus and controlled also through the USB bus. Instead of using simple LCD display in the firmware the graphical screen of the PC serves for display functions.

The HW design is worked out for an 8-pin microprocessor (ATtiny45-20).

The advantages and disadvantages of this implementation are as follows:

Advantages:

- No need for external power supply for the firmware. (The firmware is supplied from the USB bus.)
- The simple alpha-numeric LCD display of the original firmware is replaced with the PC’s intelligent graphical screen.
- Instead of a crystal oscillator the firmware is driven from its internal RC oscillator which is calibrated with the USB. The Start-Of-Frame packet on the USB bus is used to generate the 1ms time base for the measurements.
- Implementing complex control functions and the man-machine interface in PC, significant free code space is available for additional functions and further developments in the microprocessor of the firmware.

Disadvantages:

- Slightly increased HW complexity comparing to the original design (polar-relays, extra latch, gates, connector and USB cable).
- Only 3-4 pin out of the 8 of the microprocessor is available for the realized applications and for further developments.

The main hardware components are:

- One 8 pin microprocessor (ATtiny45) for interfacing the USB bus and for implementing all the functions of the different applications.
- LC oscillator circuit
- 3 logic IC's (1 shift register, 2 gates) and 4 latch relays
- a 3-pin connector for accepting L/C elements to be measured
- USB connector
- a 6-pin-connector and an 8 pin DIP-switch for additional extended applications and for programming the firmware through ISP interface.

2. The content of the LC_meter_USB package

The “LC_meter_USB.zip” package contains all the SW and HW documentation as follows:

- LC_meter_USB.pdf — Detailed technical documentation of the package (this file)
- README_LCmeterUSB.txt — An extract of the LC_meter_USB.pdf file
- License.txt — Free Open Source license (GPL) for the package
- CHANGELOG.txt — Documentation of changes in LC_meter_USB.zip
- LC_meter_USB_Atmel\ — SW package for the firmware (based on V-USB, ATtiny45-20)
  - default\ — Output folder: results of compilation
  - libs-device\ — Library for RC oscillator calibration routine
  - usbdrv\ — USB driver folder
  - LC_meter_USB.aps — Project file
  - lc_meter_usb.aws — Project file
  - main.c — Main program
  - usbconfig.h — USB configuration header file
  - Fuses.txt — Content of fuse bytes (for the programmer)
- LC_meter_USB_PC_VB\ — PC application for controlling the firmware (Visual Basic)
  - bin\ — Output folder: results of build
  - My Project\ — Project folder
  - obj\ — Output folder: results of build
  - Service References\ — (empty folder)
  - AssemblyInfo.vb — Attributes controlling an assembly
  - Debugging.vb — Function used in Debug.Write statements
  - DebuggingDeclarations.vb — Function formatting debug messages
  - DeviceManagement.vb — For detecting devices and receiving device notifications
  - DeviceManagementDeclarations.vb — API declarations relating to device management
  - FileIODeclarations.vb — API declarations relating to file I/O
  - FrmMain.Designer.vb — Procedures required by the Windows Form Designer
  - FrmMain.resx — Microsoft ResX Schema
  - FrmMain.vb — Main program
  - Hid.vb — For communicating with HID-class USB devices
  - HidDeclarations.vb — API declarations for HID communications
  - LC_meter_USB.sln — Microsoft Visual Studio Solution File
  - LC_meter_USB.suo — Microsoft Visual Studio Solution File
  - LC_meter_USB.vb — Runs the application
  - LC_meter_USB.vbproj — Project file (Loads VB project environment)
  - LC_meter_USB.vbproj.user — Project Tools
  - readme.txt — License related information

3. License Related Issues

This project is based on several other projects (published on the Internet) having different license regulations.

The links to these other projects are listed as follows:

- Very Accurate LC Meter based on PIC16F84A IC.doc

  http://ironbark.bendigo.latrobe.edu.au/~rice/lc/ (#1)

  http://sites.google.com/site/vk3bhr/ (#2)

- Virtual USB (vusb-20100715.zip)

  http://www.obdev.at/products/vusb/download.html (#3)

  License information and other useful links:

  - http://www.obdev.at/products/vusb/license.html (#4)

- http://www.obdev.at/vusb/ (#5)
- http://www.obdev.at/products/vusb/links.html (#6)

– generic_hid_vb_50 (language: Visual Basic .NET)

http://www.lvr.com/hidpage.htm (#7)

http://www.lvr.com/files/generic_hid_vb_50.zip (upgraded version) (#8)

– LC_meter_USB (**this project:** LC_meter_USB.zip)

http://vusb.wikidot.com/hosted-projects

In the LC_meter_USB.zip package and in its different folders the "Readme.txt", “License.txt”, CommercialLicense.txt, USB-IDs-for-free.txt files, and other source files also contain license related regulations.

Read the regulations described above carefully!

Consider changing the following USB specific definitions in the LC_meter_USB_Atmel\usbconfig.h file and in the LC_meter_USB_PC_VB\FrmMain.c file according to the guidelines in the LC_meter_USB_Atmel\usbdrv\USB-IDs-for-free.txt file:

USB_CFG_VENDOR_ID

USB_CFG_DEVICE_ID

USB_CFG_DEVICE_VERSION

USB_CFG_VENDOR_NAME

USB_CFG_VENDOR_NAME_LEN

USB_CFG_DEVICE_NAME

USB_CFG_DEVICE_NAME_LEN

USB_CFG_SERIAL_NUMBER

USB_CFG_SERIAL_NUMBER_LEN

4. Hardware

4.1. The circuit diagram

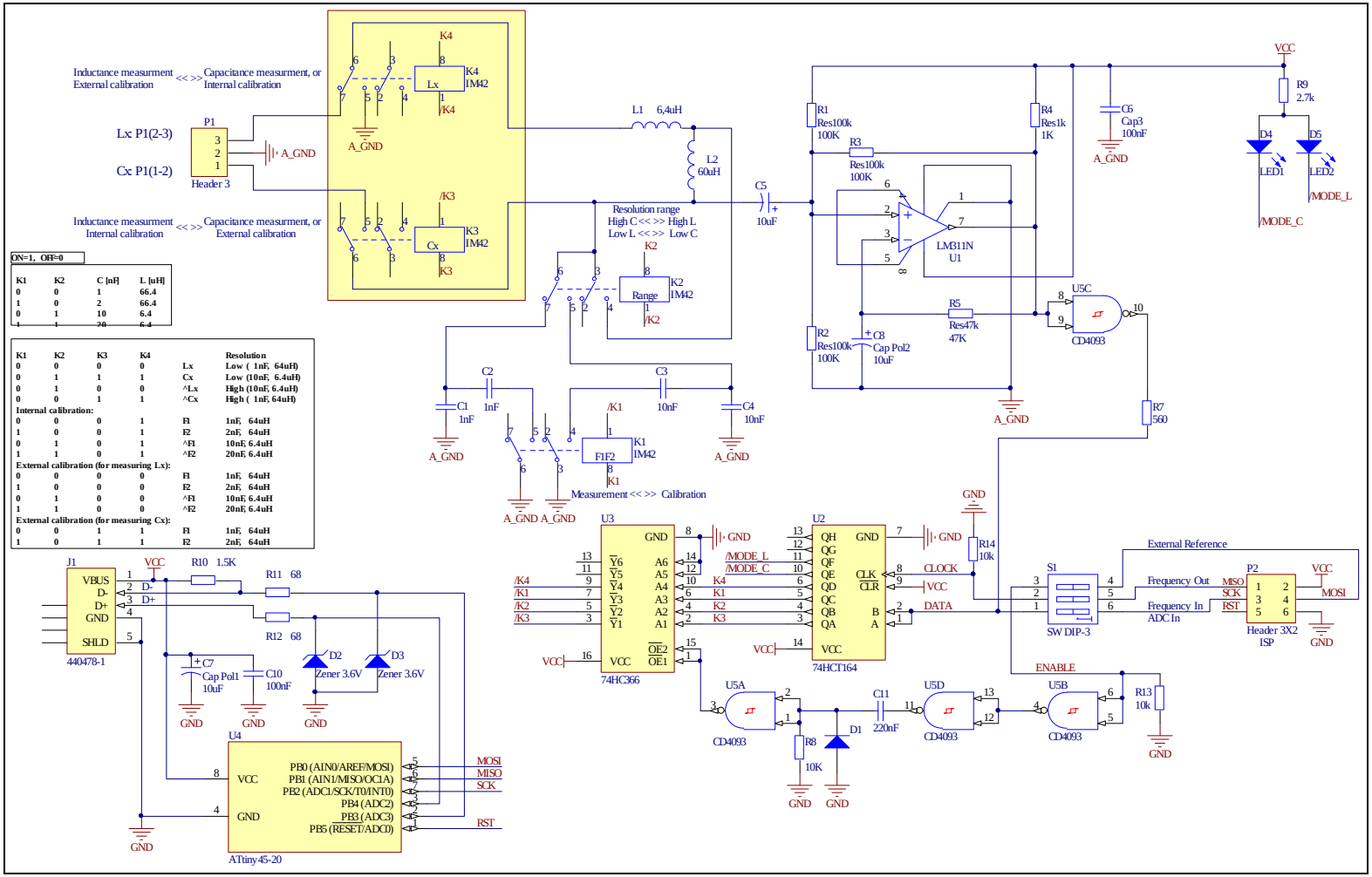

Figure 1. Circuit diagram

4.2. USB interface

Regarding to the V-USB documents (see link #3) the so called “with-zener” circuit version is adapted to interface the ATtiny45-20P to the USB bus. In order no to use the SPI control lines for interfacing to the USB, the PB3 is connected to D- and PB4 to D+ through the 68 ohm resistors. To ensure the requirements, namely the USB driver must have the highest priority interrupt in the application, the pin-change interrupt is used for the USB driver and no other interrupts are used. For precise timings in the application the SOF packet detection is used. This is why PB3 (D-) is configured for pin change interrupt in the software (see “usbconfig.h”).

4.3. LC meter

4.3.1. Theory of LC measurement

The theory of the operation of the LC meter is detailed in the documents of the original LC meter project (see link #2).

To measure the value of an inductance or capacitance, three frequency measurements should be carried out, then using the formulas below the result of the measurement can be delivered.

CAPACITANCE	INDUCTANCE
$F1 = \frac{1}{2\Pi\sqrt{LC}}$ ...(1)	$F1 = \frac{1}{2\Pi\sqrt{LC}}$ ...(5)
$F2 = \frac{1}{2\Pi\sqrt{L(C+Ccal)}}$ ...(2)	$F2 = \frac{1}{2\Pi\sqrt{L(C+Ccal)}}$ ...(6)
$F3 = \frac{1}{2\Pi\sqrt{L(C+Cx)}}$ ...(3)	$F3 = \frac{1}{2\Pi\sqrt{(L+Lx)*C}}$ ...(7)
$Cx = \frac{\left(\frac{F1}{F3}\right)^2 - 1}{\left(\frac{F1}{F2}\right)^2 - 1} * Ccal$ ...(4)	$Lx = \left[\left(\frac{F1}{F3}\right)^2 - 1\right] * \left[\left(\frac{F1}{F2}\right)^2 - 1\right] * \frac{1}{Ccal} * \left(\frac{1}{2\Pi F1}\right)^2$ ...(8)

Table 1. Equations

F1 and F2 is measured during a calibration process and these values are stored in memory for later calculations. The value of F1 or F2 is determined by the internal precise L, C and Ccal elements.

When calibration is over, the external Cx or Lx elements to be measured are connected to the instrument, and F3 is measured periodically and the value of Cx or Lx is delivered.

The actual L-C elements together with the operational amplifier form an oscillator. The signal periods on the output of the oscillator are counted for 0.4s gate time (Tg). In the software the resulting N1, N2 and N3 count values are used for further calculations based on the F=N/Tg formula and the equations in Table 2.

CAPACITANCE	INDUCTANCE
$Cx[nF] = \frac{\left(\frac{N1}{N3}\right)^2 - 1}{\left(\frac{N1}{N2}\right)^2 - 1} * Ccal[nF]$ ...(9)	$Lx[uH] = \left[\left(\frac{N1}{N3}\right)^2 - 1\right] * \left[\left(\frac{N1}{N2}\right)^2 - 1\right] * \frac{1}{k * Ccal} * \left(\frac{Tg}{2\Pi N1}\right)^2$ ...(10) *Where* $k = 1.00E-15$

Table 2. Equations based on counter values

With precise L and C values the resulting calibrating frequencies and counter values are:

FREQUENCY	L[uH]	C[nF]	F[Hz]	N	N (hex)
F1	66,4	1	617641	247056	3C510
F2	66,4	2	436738	174695	2AA67
^F1	6,4	10	629115	251646	3D6FE
^F2	6,4	20	444852	177941	2B715

Table 3. Calibrating frequencies

In ideal circumstances when there is no any gating or counting error, the resolution of the measurement (the possible minimal Lx or Cx value) is as follows:

N1	N2	N3=N1-1	Ccal [nF]	Cx min [pF]	Lx min [nH]
247056	174695	247055	1	0,008 (1)	0,538 (2)
251646	177940	251645	10	0,079 (2)	0,051 (1)

Notes: (1) High resolution
(2) Low resolution

Table 4. Resolutions of measurements

4.3.2. The operation of the circuit

During LC measurements the S1 switch should be in ON state, consequently the microprocessor can control the CLOCK and DATA lines of the shift register and the ENABLE line of the inverters.

The LM311 operational amplifier forms an LC oscillator. Latch relays connect the external Lx and Cx elements to be measured to the input of the oscillator (see Figure 1.) and the internal L1, L2 and C1..C4 parts also. The relays are controlled by the microprocessor through the shift register and gates. The latch relays are excited to change state by a short pulse formed by the C11-R8 circuit, consequently the relays consume no power otherwise.

The user can choose between two calibration modes: internal (auto) or external (manual).

In internal mode the Lx and Cx external elements are disconnected form the internal circuit, K3=0 (OFF), K4=1 (ON), and internal Ccal elements are used for calibration.

Selecting external calibration mode more precise measurement can be achieved.

In external mode, calibrating the inductance measurement Lx should be replace with a short circuit, while calibrating the capacitance measurement Cx should be removed.

In order to measure a wide range of inductors and capacitors, two range of measurement is implemented, HIGH resolution range and LOW resolution range.

Selecting HIGH resolution range for measuring small inductances, Lx and L1 are connected in series (while L2 is short-circuited) then C4 in parallel to Lx-L1. Selecting LOW resolution range for measuring greater inductances, Lx, L1 and L2 are connected in series, then parallel to C1.

Selecting HIGH resolution range for measuring small capacitances, L1 and L2 are connected in series, then Cx and C1 in parallel to L1-L2. Selecting LOW resolution range for measuring greater capacitances, Cx, C4 and L1are connected in parallel (while L2 is short-circuited).

LED1 and LED2 diodes indicate C or L measurement state respectively.

When the microprocessor configures the relays and LEDs, it drives the data line of the shift register through its low impedance SCK output line. During the signal sequence the output of the oscillator has not influence on the SCK output as the R7 resistor value is high enough. When the shift register configuration is finished, the SCK line is switched to input mode.

4.4. Extended applications

Before selecting other then the LC meter application, it is recommended to disconnect the LC measuring circuit from the microprocessor by placing the S1 switch to OFF state. External equipment can then be connected thru the P2 header to the LC meter box and controlled by the I/O lines of the processor appearing on the header pins.

The external equipment can be powered also through the P2 header if it does not consume too much current. In the USB configuration of the software 500mA total consumption is defined for the complete hardware. As the current consumption of the internal LC meter circuit is ~20mA (not considering the transients when the latch relays change state) the maximum allowable current consumption of the external circuit from P2 is around 400-480mA.

In the USB specifications 10uF is the highest value defined for power line filter. As the HW already contains such a 10uF capacitor, it is suggested to use maximum 100nF additional capacitor on the power line of P2 in the external equipment.

Interfacing an external equipment consider the I/O specification of the U4 microprocessor.

4.5. Interface connector for programming the microprocessor

As the P2 header pin-out conforms to the pin-out of the Atmel In-Circuit-Programmer (ISP) interface, an ISP programmer can be connected to it in order to program the ATtiny45 microprocessor of the instrument.

It is not necessary, but suggested to place the S1 switch to OFF state while programming.

Select the appropriate Atmel AVR programmer in the AVR Studio for programming.

4.6. HW implementation

The HW is implemented in a 100x50x25mm sized plastic box.

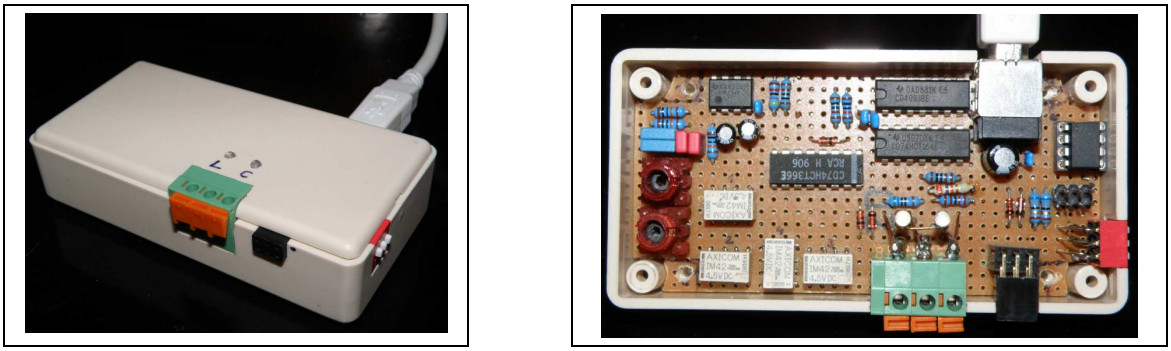

Figure 2. The implemented HW

5. Software

The host control SW in the PC was written in Microsoft Visual Basic 2010 (using the Microsoft Visual Studio Express 2010 Version 10.0.30319.1 RTMRel) under the .NET Framework V4.0. (Version 10.0.30319.1 RTMRel). These are freely downloadable form:

http://www.microsoft.com/express/Windows/

The SW in the firmware was written in C using AVR Studio development environment (AVR Studio 4.18.716, GUI Version 4.18.0.685, AVR Simulator 1.0.2.1. Plugins: AvrPluginAvrAsmObject 1.0.0.48 AvrPluginavrgccplugin 1.0.0.11 Stk500Dll 1.0.1.15). This also can be freely downloaded form:

http://www.atmel.com/microsite/avr_studio_5/default.asp?category_id=163&family_id=607

In order to program the microprocessor in the hardware device you need a programmer and a program like AVR Studio to control the programmer.

To run the LC_meter_USB.exe host application, download and install at least the .NET Framework for the PC, otherwise the application will not run on your computer. For modifying the LC_meter_USB host application, use the Visual Studio Express, or its enhanced version.

5.1. Communication between host and device

The USB HID communication is configured for Feature Report Exchange. The host (PC) controls the communication by sending a Feature report (a command to the device) then retrieving one (an answer). In this application the report length is limited for 4 bytes in each direction. The command and answer structure is detailed in the main.c file of the device.

5.2. The ATtiny45 SW

In the “LC_meter_USB.zip” package the LC_meter_USB_Atmel\ folder contains the SW for the firmware. In this folder the usbdriver folder is the unmodified copy of the same folder of the V-USB package (see: link #3).

Only the files “LC_meter_USB_Atmel\main.c”, “LC_meter_USB_Atmel\usbconfig.h”, “LC_meter_USB_Atmel\libs-device\osccal.h” and “LC_meter_USB_Atmel\libs-device\osccal.c” files had been modified for implementing the required applications.

The modifications in the “usbconfig.h” file define the required HW specific USB configuration and the “myUSB_SOF_HOOK” macro. The “myUSB_SOF_HOOK” macro starts and stops the HW timers of the microprocessor in phase with the Start Of Frame (SOF) packet on the USB. As the SOF packet is sent by the PC periodically in every 1ms, it is used as the time base of the different applications in the firmware.

5.2.1. The main.c file

The main.c file implements the different applications in the firmware.

It defines the length of the data exchange between the host and the firmware (4 byte in both direction: REPORT_COUNT 4), the data structure, commands received from the PC and the structure of answers sent by the firmware to the PC.

After power up the main() subroutine gets control. It initialize the USB, configures the HW and SW. During USB initialization the RC oscillator of the microprocessor is calibrated to 8.25MHz using the 1ms periodic rate of the SOF packet on the USB bus. Consequently the PLL circuit of the microprocessor generates 16.5MHz CPU clock from the calibrated 8.25MHz (4x8.25MHz).

The 4 byte commands received from the PC are analysed and interpreted by the InterpretCommands() subroutine, which is called from the usbFunctionWrite() subroutine. Some of the received commands are completely executed by the command interpreter, others initiate further tasks which are executed in the main() routine. If answers are required by the PC for a command, the command interpreter finishing its task prepares the required content of the 4 byte answer.

The main() routine periodically calls the usbpoll() routine, and executes tasks initiated by the interpreter, if exist any.

In the present applications the only such task in the main() routine is to count the signal changes on the T0 input of the microprocessor with the timer/counter0 when MeasureLC or MeasureFrequency application is selected and started. To do this, the main() routine interacts with the USB driver through the myUSB_SOF_HOOK macro defined in the “usbconfig.h” header file. Through this interaction counter0 is gated (started and stopped) at the required SOF detection tickle time without delay. The resulting accuracy of the gate time is acceptable.

This interaction works as follows:

When the main() routine detects a Start condition set by the command interpreter, it waits till the usbSofCount changes. This change signals the 1ms tick time, generated from the USB bus. Then it waits further for a fraction of 1ms doing nothing. (This second wait cycle is necessary, because double SOF detection is noticed at the beginning of most of the frames, and we want to use the first one only!) Then the GateControl variable is set to 0, signalling to the myUSB_SOF_HOOK macro to start the counter0 when the first SOF packet of the next

frame is detected. If the counter0 is started, the main() routine decrements the gate time counter (msCNT) once in each frame using usbSofCount change, till gate time counter value equals 1, and at this time sets GateControl variable to 0x83, signalling to the myUSB_SOF_HOOK macro to stop counter0 when the first SOF packet of the next frame is detected.

5.2.2. Installing the SW package for the firmware

Unpack the "LC_meter_USB.zip" package then start the AVR Studio by double-clicking to the LC_meter_USB_Atmel\LC_meter_USB.aps file. Use AVR Studio to modify or extend the application as you wish. The result of the compiled application (LC_meter_USB.hex and the other files) is placed in the LC_meter_USB_Atmel\default folder. To program the microcontroller in the HW select the appropriate programmer in AVR Studio, start the programmer, check the fuse bits, and change them if necessary using the information in the LC_meter_USB_Atmel\Fuses.txt file. Then program the fuse bits. Finally program the code memory by selecting LC_meter_USB_Atmel\ default\LC_meter_USB.hex file for programming.

5.3. The host SW

The host control SW in the PC is based on the USB test program "generic_hid_vb_50.zip" available on the Internet. Downloading this compressed file (or its updated version if exists) using link (#7) and (#8), first read all the license related files in it (readme.txt, etc.).

Comparing the resulting LC_meter_USB_PC_VB host software package to the generic_hid_vb_50.zip package, the FrmMain.vb frame file differs significantly.

Many routines had been completely deleted and some of the remaining ones slightly modified. The remaining part of the original file is placed to the first half of the new FrmMain.vb file, where the small modified lines are commented with **'---** or **'+++** comments, indicating the deleted or new lines.

On the other hand the second half of the new FrmMain.vb file contains the new declarations and routines.

The routines deleted from the original FrmMain.vb file are as follows:

```
Private Sub cmdContinuous_Click
Private Sub cmdInputReportBufferSize_Click
Private Sub cmdFindDevice_Click
Private Sub cmdOnce_Click
Private Sub ExchangeInputAndOutputReports
Private Sub GetVendorAndProductIDsFromTextBoxes
Private Sub InitializeDisplay
Private Sub tmrContinuousDataCollect_Tick
Private Sub txtProductID_TextChanged
Private Sub txtVendorID_TextChanged
```

Furthermore in the Hid.vb file one line is commented out (Imports GenericHid.FileIO), while the HidDeclarations.vb file is untouched.

5.3.1. The screens

5.3.1.1. LC measurement mode

Starting the **LC_meter.exe** program on the PC the following application window opens.

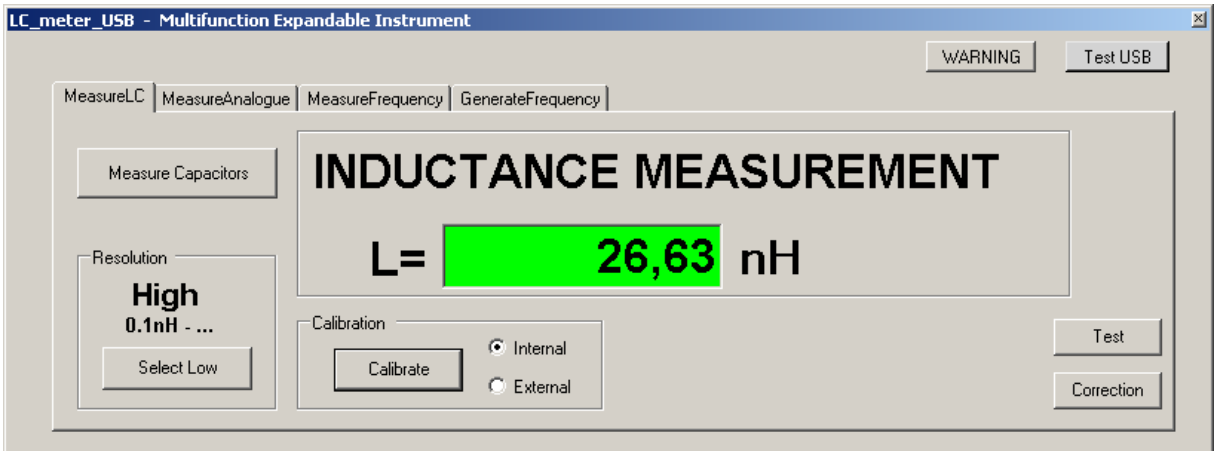

Figure 3. LC meter tab page

Clicking the **WARNING** button a message box pops up with the license related text in it.

The **LC_meter** application and the additional applications can be controlled on different tab pages. Having decided what to measure or generate, click to the required tab.

Starting the program the first tab (**MeasureLC**) is activated, and **Inductance Measurement** state is selected. To change for **Capacitance measurement**, click the **Measure Capacitors** button.

You can select between **High** or **Low** resolution measurement modes and between **Internal** or **External** calibration modes.

Selecting **External** calibration mode for capacitance measurement, remove the capacitor to be measured (Cx) from the socket then click the **Calibrate** button to start the calibration process. After calibration, place the different capacitors to be measured into socket C, then the measured values will be displayed.

Selecting **External** calibration mode for inductance measurement, place a short-circuit into the socket L then click the **Calibrate** button to start the calibration process. After calibration place the different inductors to be measured into socket L.

Selecting **Internal** calibration, the socket for L or C components is disconnected from the internal circuit of the instrument, consequently, if the socket contains any L and/or C element to be measured, those do not disturb the calibration process.

Clicking the **Correction** button the content of the **MeasureLC** tab page changes, and the correction parameters used for the calculations are displayed then can be modified. On Figure 4 the default values are represented.

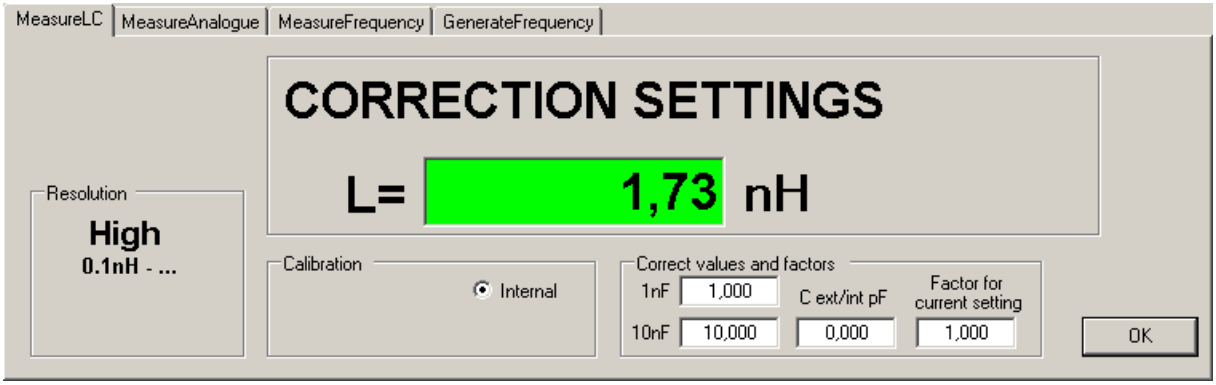

Figure 4. LC meter tab page – Correction settings

The actual value of the C2 and C3 calibrating capacitors can differ from their nominal value. In order to use the actual correct values for calculations, type these correct values in to the **1nF** and **10nF** input fields (positive floating numbers).

Using internal calibration the stray capacitance (and inductance) of the circuit of the HW is less then when using external calibration. Consequently the result of the measurement will not be the same in the two calibration modes, especially when measuring small value capacitors. With the **C ext/int pF** value (signed floating number) this difference in pF can be compensated.

The **Factor for current setting** multiplication coefficient (positive floating numbers) is applied to the result at the end of the calculations of the selected (current) measurement. Eight different setting combinations exist (based on L or C measurement mode, High or Low resolution and Internal or External calibration), and consequently one of the eight different coefficients is displayed for modification and assigned to the current measurement setting.

To return from this **Correction settings** mode and to continue measurements, click the OK button.

Exiting the program the above coefficients (2+8=10) are stored in the LC_config.txt file and loaded on the next start of the program.

In order to check the proper operation of the **LC_meter HW** (or test it after building it) clicking the **Test** button, then the content of the **MeasureLC** tab page changes again as shown below.

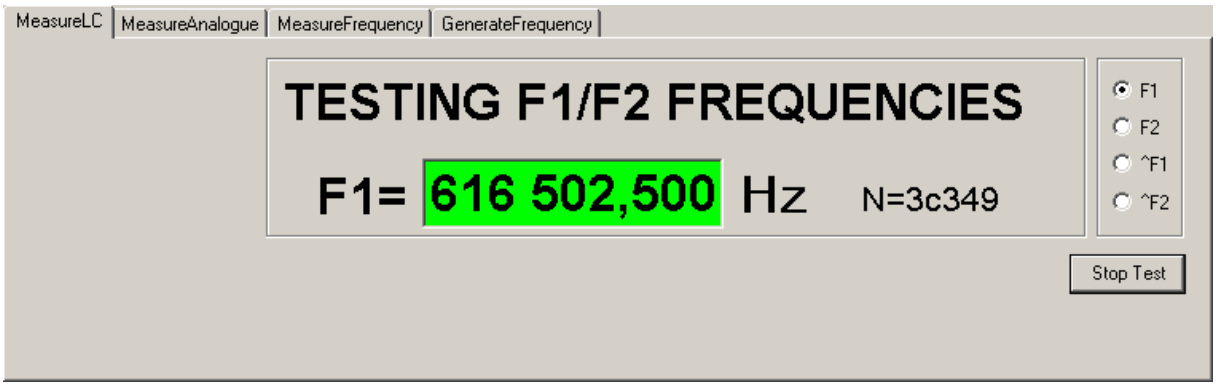

Figure 5. LC meter tab page – Frequency tests

On this screen the special frequencies of the internal LC oscillator of the **LC_meter HW** can be selected for display. These frequencies are measured with disconnected external Lx or Cx

element (Internal calibration mode). The frequency values are delivered from counter values (N) collected in 400ms, and the relevant counter value is also displayed.

5.3.1.2. Analogue signal measurement mode

To measure/log external analogue signals, or some specific parameter of the LC_meter HW or the microprocessor applied in the HW, click the **MeasureAnalogue** tab, then the screen changes to the following.

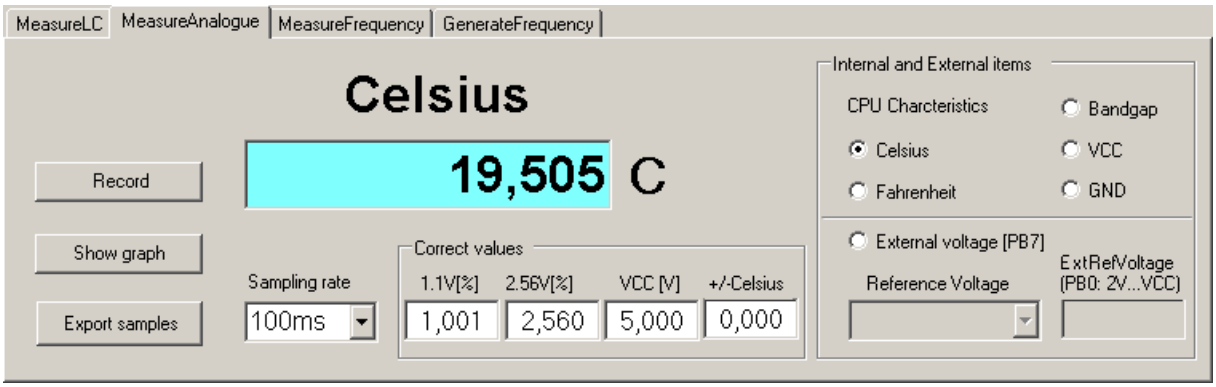

Figure 6. Analogue measurement tab page

By default the measured temperature of the microprocessor is displayed in Celsius. To measure external analogue voltages, click the **External Voltage** button, then select the appropriate reference voltage. For reference voltage either internal or external ones can be selected. Selecting external reference (**EXT**) fill the value of the **ExtRefVoltage** supplied to the HW.

If there is some error in the result of those measurements which use one of the internal voltage references, and the value of the internal voltage reference differs from its nominal value, then type the correct value in the relevant input field. To correct the measured temperature results, type the signed difference in Celsius in the **+/-Celsius** input field.

Selecting **Bandgap** or **VCC** measurement from the CPU characteristics, the displayed result will neither reflect the actual Vbg value of the microprocessor, and probably nor the actual value of the VCC.

> It is because the bandgap voltage (Vbg) is measured using the 2.56V internal voltage reference, which is generated from the Vbg in the microprocessor. Consequently the result will be always very near to 1.1V, and this result only reflects, whether the generation of the 2.56 reference voltage and the operation of the AD converter is correct.
>
> On the other hand, the result of the VCC measurement uses the 1.1V internal voltage reference. As the value of this reference voltage has a great tolerance (1.0 .. 1.2V), the measured VCC can contain 10% error. To compensate it, use a digital voltmeter for determining the VCC value, then modify the 1.1V internal voltage reference value in the input field till the displayed VCC value equals with the value measured by the voltmeter.

Selecting a measurement mode the result can be recorded in time by clicking the **Record** button. To display the recorded values in a graph click the Show graph button.

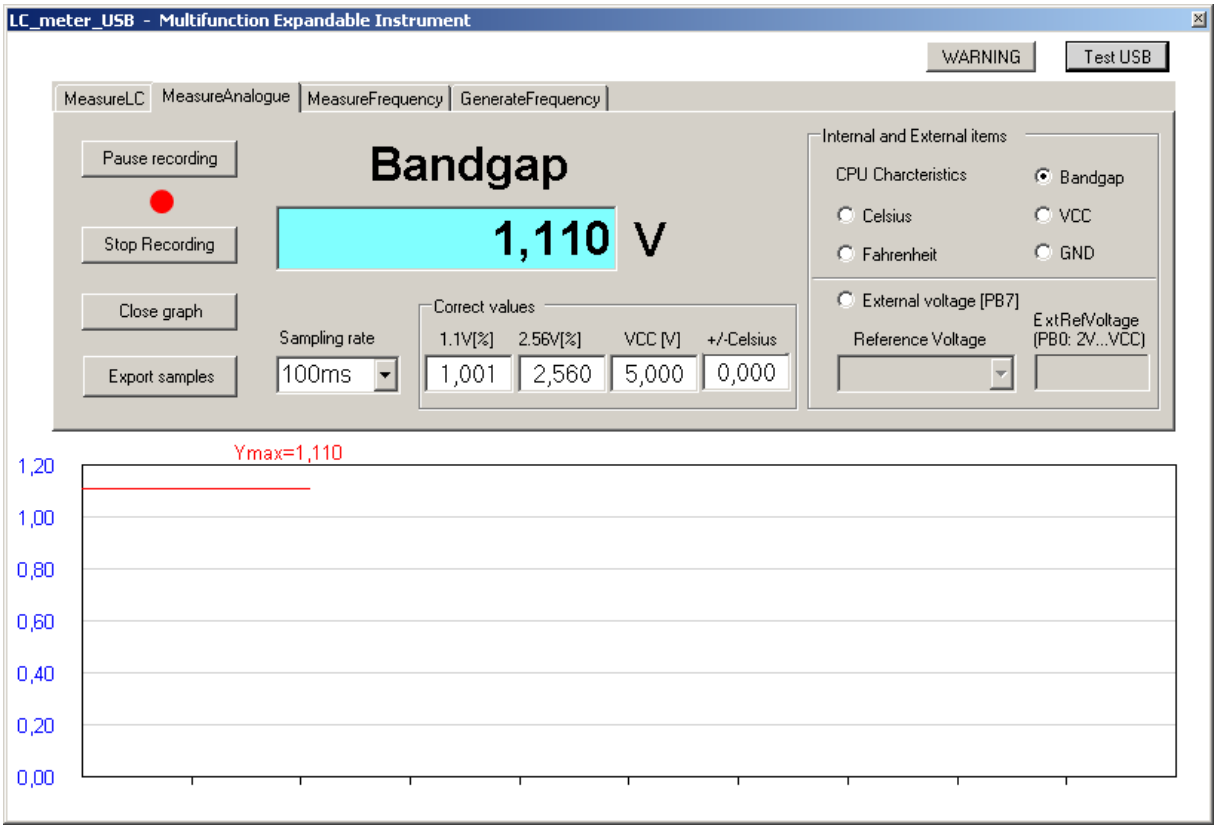

Figure 7. Analogue measurement tab page – Graph displayed

The recorded normalized values and the measured maximum value are presented on the graph in a moving window giving space for 1024 samples.

Select the **Sampling rate** as required from the predefined list.

The current content of the graph (the sampled values) can be exported by clicking to the **Export samples** button for further processing. (Excel is able to import the output file.)

5.3.1.3. Frequency measurement mode

Selecting the frequency measurement application by clicking to the MeasureFrequency button, either the frequency on an external input line, or the clock frequency of the CPU of the HW can be measured.

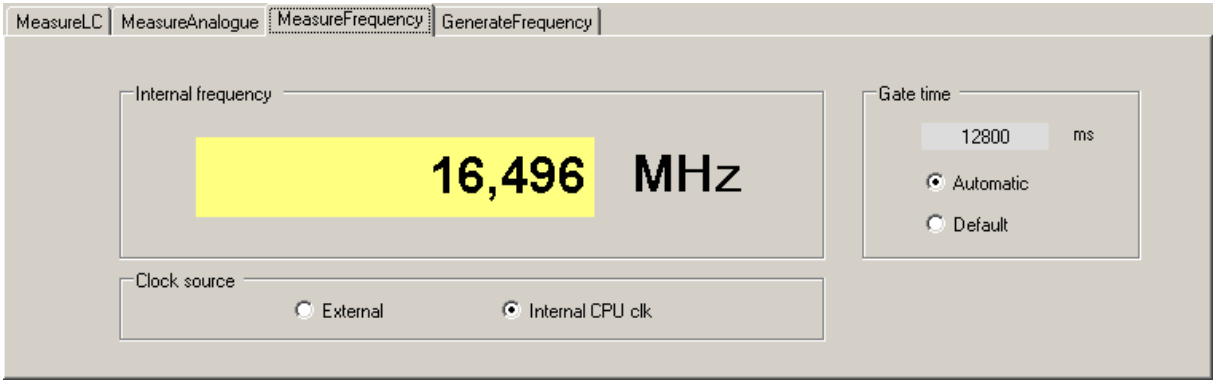

Figure 8. Frequency measurement tab page

The upper limit for measuring frequency is 6.6MHz.

If the Default radio button is active then the gate time is set to constant 400ms. Alternatively, activating the Automatic radio button the gate time is also set to 400ms, but depending on the result of the measurement it is increased or decreased automatically till adequate measurement accuracy is reached.

5.3.1.4. Frequency generator mode

With the frequency generation application a fix set of frequencies is available for generation in the range 3.965Hz to 33MHz. The set of frequency is defined by the configuration parameters of the CPU counter/timer1, which divide the 66.5MHz CPU frequency. The formula for calculating the set of selectable frequencies in Hz is:

F=66000000 / (2^S) / D where S={0,1,2, … 16} and D={1,2,3, … 255}.

For 50% duty cycle signals D can take even values only.

There are many identical frequencies in the set but the program deals with the different ones.

If signal with 50% duty cycle is required, the user can select 1151 different one out of the total 2159 frequencies in this subset. Otherwise, if signals with duty cycle different then 50% are also allowed, the number of selectable different frequencies is 2302 out of the 4318 in this subset.

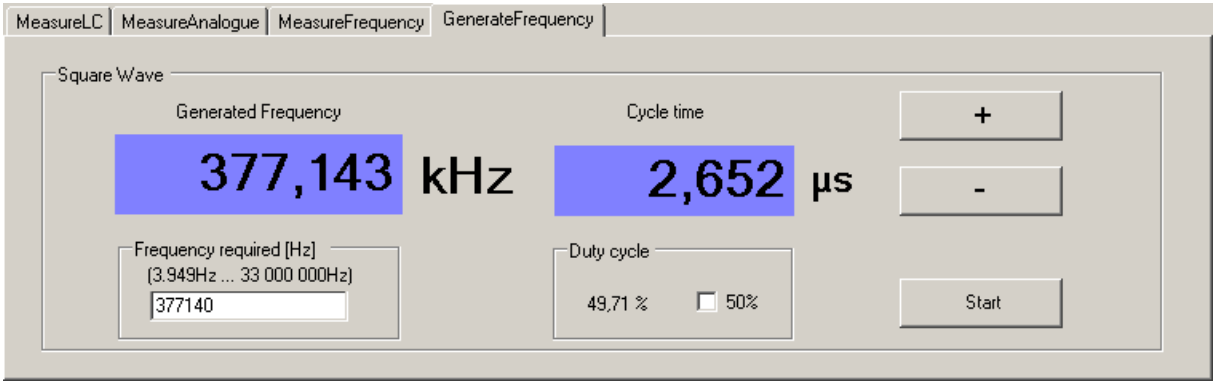

Figure 9. Frequency generation tab page

When typing in the required frequency, the application searches for a match in the set. If the required frequency exactly matches a frequency in the set then it is selected for generation immediately. Otherwise the nearest higher one is selected.

The **Start**/**Stop** button activates/deactivates the output signal on the interface.

The selected frequency can be increased or decreased to the nearest higher or lower next one in the set by clicking to the "**+**" or "**—**" button respectively.

The accuracy of the generated frequency is the function of the accuracy of the CPU clock, which is calibrated to the 1ms Start-Of-Frame packet period of the USB bus at powering up the HW. The calibrated CPU clock frequency is sensitive for temperature and USB bus power changes.

5.3.1.5. Testing USB bus communication

The 4 byte two-direction communication between the host (PC) and the firmware can be tested, as USB setup and status information, commands issued by the host and answers from the firmware are logged during the operation of the applications. Clicking to the **Test USB**

button the content of the USB log buffer, the number of commands issued **(# USB exchange**) and the content of the last answer (the 4 **Bytes received**) is displayed in the application window under the selected tab page area.

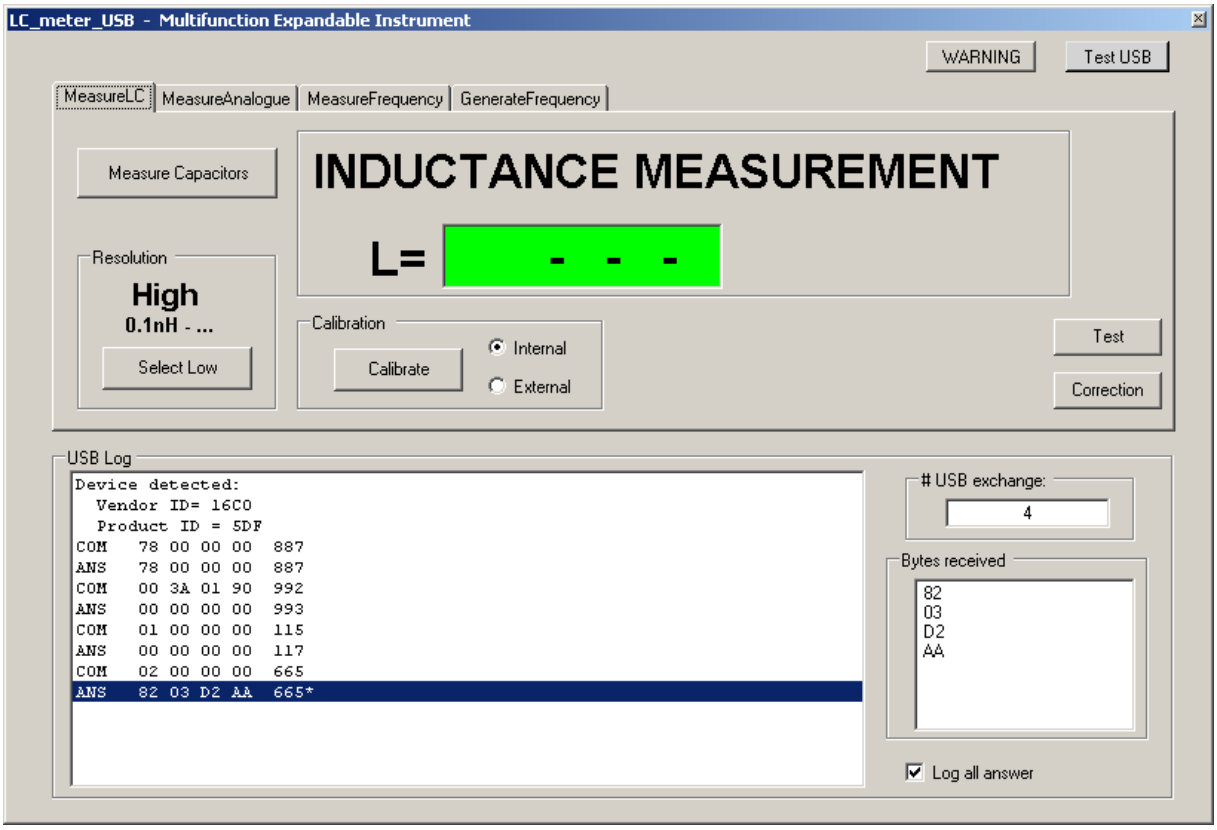

Figure 10. USB bus communication log

Command and answer information in the log buffer is displayed in individual lines. Command lines are identified with **COM**, while answer lines with **ANS** abbreviation. Following the **COM** or **ANS** identifiers the sent or received data (4 byte) is displayed in hexadecimal form. The data bytes are followed by the current time information in the host in "ten-thousandths of seconds" format. Answer-lines carrying measured values (the first byte is > 127) are terminated with a "*"character. Answers not carrying measured values are filtered out from the log buffer if the **Log all answer** checkbox is not checked.

If the **Test USB** button turns red then the host program in the PC informs the user that a communication errors happened (device removed, device not found). When the communication link works again (device attached, device detected) the **Test USB** button turns back to grey.

5.3.2. Installing the HOST SW package

Unpack the "LC_meter_USB.zip" package then start the Visual Studio Express – Visual Basic by double-clicking to the LC_meter_USB_PC_VB\LC_meter_USB.vbproj file. Use Visual Studio to modify or extend the application as you wish. The result of the compiled application (LC_meter_USB.exe and the other files) is placed in the LC_meter_USB_PC_VB\bin\Release\ or in the LC_meter_USB_PC_VB\bin\Debug\ folder, depending on whether you clicked to the "Build LC_meter_USB" in the Build menu, or to the "Start Debugging (F5)" icon in the toolbar.

To run the program press F5, or double click to the LC_meter_USB.exe file.

5.3.3. Running the application on PC having no Visual Studio installed

If you want to run the LC_meter_USB application on a PC having no Visual Studio installed, then do the following:

- Copy the content of the "LC_meter_USB_PC_VB\bin\Release\app.publish\" folder to a new folder of your PC. The content of the new folder will be:
  - folder "Application Files"
  - file "LC_meter_USB.application"
  - file "setup.exe"
- Download and install the "Microsoft VisualBasic. PowerPacks. Vs Version 10.0.0.0" runtime module from http://go.microsoft.com/fwlink/?LinkID=145727&clcid=0x804
- Start the installation of the LC_meter_USB application by double clicking to the setup.exe file in your new folder. This installation will place the LC_meter_USB application to the START menu. You can also uninstall the application later thru the Control Panel.

6. Realizing new applications

In the host the FrmMain.vb program can be easily extended if there is a need to create new applications. There is space enough for creating new tab pages beside the existing ones. Just create a new tab page, then assign the necessary control and display element on it and extend the FrmMain.vb program with the required new declarations, subroutines and functions in order to control your HW extension connected to the P2 header of the LC_meter HW.

In the device there is also free space enough for adding new applications to the existing ones. Furthermore without any change in the circuit diagram, the microprocessor of the current design can be upgraded for the ATtiny85, which has 2 times more code and data memory space.

For a new application extend the command/answer structure as required.

7. Notes

Verifying the HW and the SW, and reading related documents I faced with some situation which need further considerations:

- In the current firmware the watchdog reset function is not used.
- Calibrating the RC oscillator of the ATtiny45 CPU the original calibration routine is applied which implements a binary search algorithm in order to find the required OSCCAL register value in a few steps. Executing this algorithm the OSCCAL register step value starts from 128 (0x80), then the next step is the half of the previous. As a contradiction, the ATtiny45 technical document (preliminary!) says that *the changes in OSCCAL should not exceed 0x20 for each calibration* otherwise it can lead to unpredictable behaviour. This contradiction needs further consideration.

- The circuit path from the Lx or Cx to the operational amplifier should be minimized in order to:
  - minimize the stray capacitance and inductance
  - minimize noise and other external effects (It is suggested to keep distance from the instrument during measurements.)
- A grounded metal shield inside the case of the instrument would decrease external effects.
- Connecting Cx (or Lx) to the instrument, the frequency of internal LC oscillator gets stabilized slowly, especially when the Cx value is relatively high. With frequent calibration the result of the measurement is acceptable.
- The voltage of the USB power line is noisy, changes significantly during measurements, which also has an effect on the accuracy of the instrument. One possible solution for this is the application of a low drop power stabilizer (with 4.5V .. 4.7V output) which takes power from the USB and supplies current for the CPU or even for the operational amplifier. Take into account that the ATtiny45-20 does not operate on a frequency near to its maximum if its VCC is below 4.5V. Most of the relays usually do not operate under 5V also, so operate the relay drivers from the USB bus power line.

  If the microprocessor is powered from a voltage stabilizer output (VCC_CPU < VCC) then take care of the followings:
  - Change VCC to VCC_CPU on the P2 header
  - Connect an anti-parallel diode pair in series with R7. This decreases the logic level coming from the U5C gate output to the microprocessor.
- Selecting the analogue measurement application, measurements which are based on the internal voltage references (1.1V or 2.56V) can have significant error. This is because by specification the internal reference voltage of the microprocessor can take a value in the 1.0V-1.2V interval. (The 2.56V internal voltage reference is probably generated from the 1.1V Vbg and proportional to its value.)

  As an example we want to measure the 5V CPU voltage. In this case the CPU voltage is selected for reference and the 1.1V Vbg is selected as input source, and the equation for the result is: VCC=Vbg*1024/(256*ADCH + ADCL). In our case the result will be between 4,55-5,46V. One way to eliminate this, configure the PB0 of the microprocessor for AREF output, and measure the internal reference with a digital multimeter. Then set the correction factor on the screen as required.

  For more precise measurements select external voltage measurement mode and external voltage reference.
- The complete project was tested on a PC with USB and XP operating system. Copying the LC_meter_USB.exe program to any other folder (or to other similar PC) the application will run if the proper version of the .NET Framework is installed on the machine. (The place of the .exe program is the LC_meter_USB_PC_VB_50/bin/Debug/ folder or the LC_meter_USB_PC_VB_50/obj/x86/Debug/ folder.)

- The 1ms time base derived from the USB bus is accurate enough as the USB standard defines a frame interval to be 1.000ms ±500ns. This have a limit for accuracy for the L-C meter and frequency meter applications.
- On the HW circuit diagram all the latch relays are type DPDT, whereas the 4 relays could be replaced with 1 DPDT, 1 SPST and 2 DPST ones. The reason for representing 4 DPDT relays on the circuit diagram is that I found cheep used miniature IM42 DPDT surface mounted relays in a shop. Alternatively reed relays can also be applied with small modifications on the HW and also modifying the shift register patterns in the SW. A possible solution for using reed relays is shown on Figure 11.

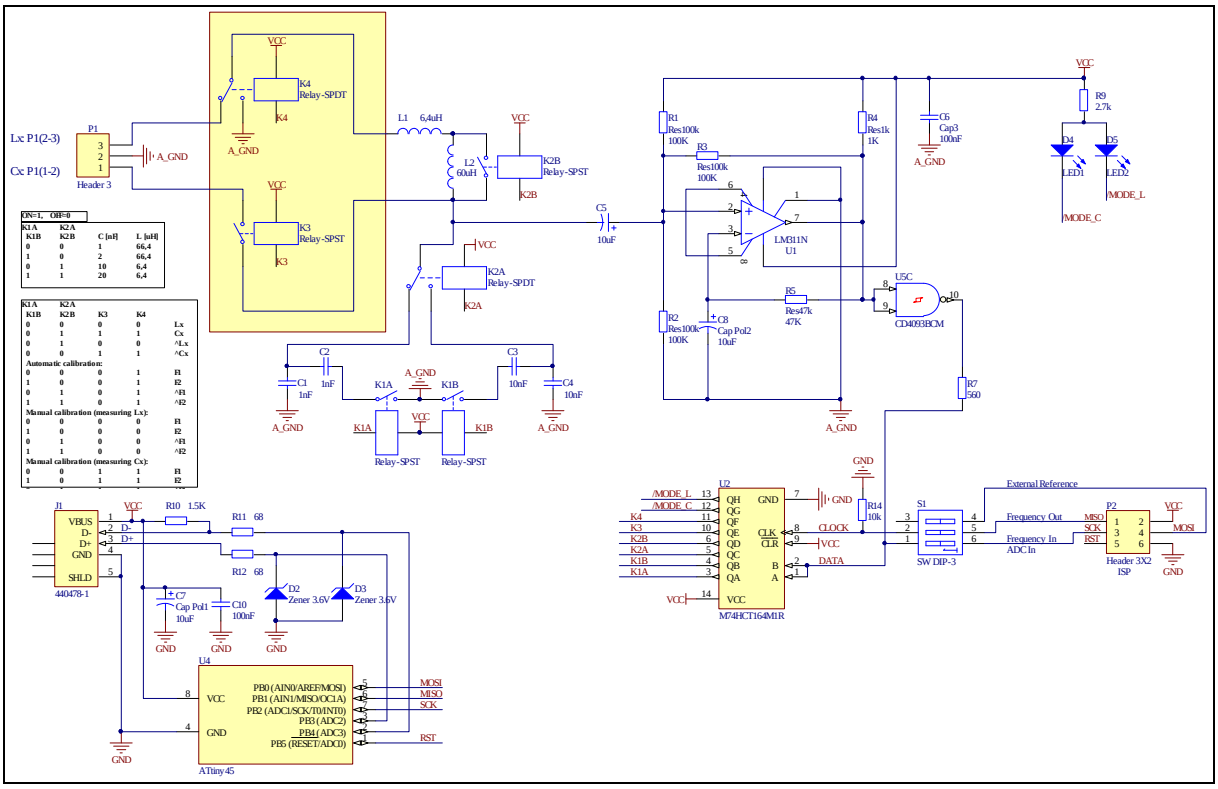

Figure 11. Alternative circuit diagram using reed relays