L-C Meter

History

Here is my new inductance & capacitance meter. I had took the analog oscillator part from a well know schematic found on the internet and adapt it for a AVR. It's impressive to have a very accurate result with a so small circuit. The way it's work is simple you chance a oscillator frequency with the part you want to find is value. The frequency is read and with some math you get the value. Here is how it's work, At power-up you read the oscillator frequency without any thing connected to you probe, we will call this value f1. After the calibration process via a relay put a very accurate capacitor of 1000pF, you read this new frequency f2. When the calibration is done you can place your unknown part value and read the frequency f3. Now we go in math...

For capacitor:

a = ((f1/f3)^2)-1

b = ((f1/f2)^2)-1

result = (a / b) * 0.001

For Inductor:

a = ((f1/f3)^2)-1

b = ((f1/f2)^2)-1

c = 1000000000

d = (1/(2*pi*f1))^2

result = a * b * c * d

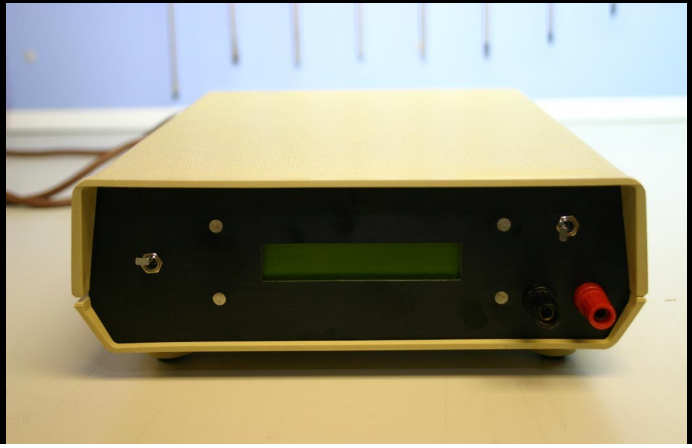
result = result * 1000000

Features

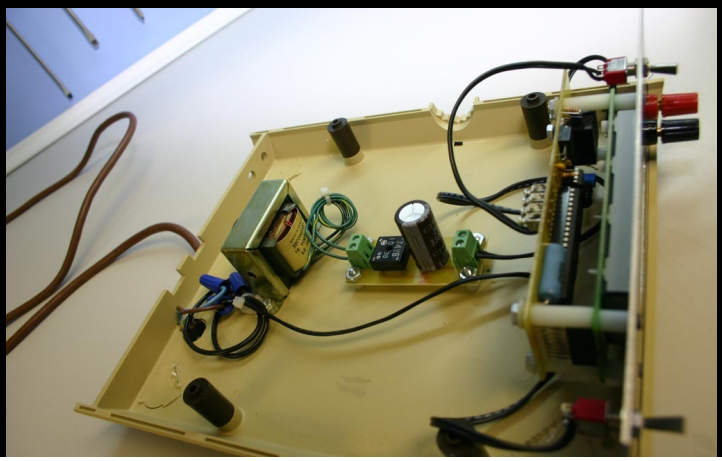
- Inexpensive
- High acuarity
- Auto scale
- Auto calibration
- Can be use as a frequency meter with only small modification
- Measure capacitance and inductance

Pictures

Outside



Inside



Sources codes & Schematics

-Schematic in PDF format

-Layout in PDF format

```
//*****************************************************************************
// LCMeter
// Version 1.0 Nov 2005
//
// 1.0 -> First Release
//
// Sylvain Bissonnette
//*****************************************************************************
// Editor : UltraEdit32
//*****************************************************************************
//
//                   R E T U R N   S T A C K   2 4
//                         X T A L  16 MHZ
//
//*****************************************************************************
//                          P I N   U S A G E
//
// PA0 -> LCD Data bit 7
// PA1 -> LCD Data bit 6
// PA2 -> LCD Data bit 5
// PA3 -> LCD Data bit 4
// PA4 -> LCD E
// PA5 -> LCD RS
// PA6 -> n/c
// PA7 -> n/c
//
// PB0 -> n/c
// PB1 -> Input
// PB2 -> n/c
// PB3 -> n/c
// PB4 -> n/c
// PB5 -> n/c
// PB6 -> n/c
// PB7 -> n/c
//
// PC0 -> n/c
// PC1 -> n/c
// PC2 -> n/c
// PC3 -> n/c
// PC4 -> n/c
// PC5 -> n/c
// PC6 -> n/c
// PB7 -> n/c
//
// PD0 -> n/c
// PD1 -> Use serial port as timer
// PD2 -> n/c
// PD3 -> Relay
// PD4 -> n/c
// PD5 -> n/c
// PD6 -> n/c
// PD7 -> n/c
//
//*****************************************************************************
//                         T I M E R   U S A G E
//
// Timer 0 not use
// Timer 1 used for frequancy counter
// Timer 2 not use
//
//*****************************************************************************

//*****************************************************************************
//                       I N C L U D E
//*****************************************************************************
#include <iom32v.h>
#include <shortnametype.h>
```

```
#include <macros.h>
#include <stdlib.h>
#include <stdio.h>

//***************************************************************************
//                      D E F I N E
//***************************************************************************
#define TRUE              1
#define FALSE             0
#define XTAL              16000000

// LCD
#define LCD_D7            0x01
#define LCD_D6            0x02
#define LCD_D5            0x04
#define LCD_D4            0x08
#define LCD_E             0x10
#define LCD_RS            0x20

#define LCD_PIN           PINA
#define LCD_DDR           DDRA
#define LCD_PORT          PORTA

#define LCD_DATA          0x01
#define LCD_CTRL          0x00

#define SWITCH_PIN            PINC
#define SWITCH_DDR            DDRC
#define SWITCH_PORT           PORTC
#define SWITCH_BIT            (1<<PB7)

#define RELAY_PIN             PIND
#define RELAY_DDR             DDRD
#define RELAY_PORT            PORTD
#define RELAY_BIT             (1<<PB3)
#define CAP_REF                  (1<<PB7)

#define CAP                     0
#define IND                     1
#define UNKNOWN                 3

//***************************************************************************
//                  P R O T O T Y P E
//***************************************************************************
void main(void);

// LCD
void LCDInit(ushort x,ushort y);
void LCDClrSCR(void);
void LCDGotoXY(ushort,ushort);
void LCDWriteString(char *);
void LCDWriteConstString(flash char *);
void LCDWriteData(ushort,ushort);
void LCDWait(void);
void LCDDelay50us(int Delay);

// Serial Port
void SerialInit(void);

// Main Function
void Calibration(void);
ulong MesureLC(void);
void CounterInit(void);

//***************************************************************************
//           G L O B A L   V A R I A B L E
//***************************************************************************
// LCD
ushort MaxX, MaxY;

// Software
ulong CountLow = 0;
ulong CountHigh = 0;
ulong Count = 0;

float Value;
float f1,f2,f3,a,b,c,d;

char Buffer[20];

//***************************************************************************
//                      M A I N
//***************************************************************************
void main(void)
{
```

```
   uint i,j;
   short State = TRUE;

   LCDInit(20,2);
   SerialInit();
   CounterInit();

   SWITCH_DDR &= ~SWITCH_BIT;
   SWITCH_PORT |= SWITCH_BIT;

   RELAY_DDR |= RELAY_BIT + CAP_REF;

   SEI();

   LCDGotoXY(1,1);
   LCDWriteConstString("LC M e t e r   V:1.0\0");
   LCDGotoXY(1,2);
   LCDWriteConstString(" Sylvain Bissonnette\0");
   for (i=0;i<65000;i++) for (j=0;j<50;j++) WDR();

   State = UNKNOWN;
   while(TRUE)
   {
      if (SWITCH_PIN & SWITCH_BIT)
      {
         RELAY_PORT &= ~RELAY_BIT;
         if (State != CAP)
         {
            State = CAP;
            Calibration();
         }

         f3 = (float)MesureLC();
         a = ((f1/f3) * (f1/f3)) - 1;
         b = ((f1/f2) * (f1/f2)) - 1;
         Value = (a / b) * 0.001;

         if (Value < 0) Value = 0;
         csprintf(&Buffer[0],"  Cap: %#08.6f uF  \0", Value);
         LCDGotoXY(1,1);
         LCDWriteString(&Buffer[0]);

         if (Value < 0) Value = 0;
         if (Value < 0.001)      csprintf(&Buffer[0],"  Cap:  %3.0f pF           \0", Value * 1000000);
         else if (Value < 1)     csprintf(&Buffer[0],"  Cap:  %3.0f nF           \0", Value * 1000);
         else if (Value > 0)     csprintf(&Buffer[0],"  Cap:  %3.0f uF           \0", Value);
         LCDGotoXY(1,2);
         LCDWriteString(&Buffer[0]);
      }
      else
      {
         if (State != IND)
         {
            State = IND;
            Calibration();
         }
         RELAY_PORT |= RELAY_BIT;

         f3 = (float)MesureLC();
         if (f3 > 1000)
         {
            a = ((f1/f3) * (f1/f3)) - 1;
            b = ((f1/f2) * (f1/f2)) - 1;
            c = 1000000000;
            d = (1/(6.283*f1)) * (1/(6.283*f1));
            Value = a * b * c * d;
            Value *= 1000000;
            if (Value < 0) Value = 0;
            csprintf(&Buffer[0]," Inductor: %4.0f uH        \0", Value);
            LCDGotoXY(1,1);
            LCDWriteString(&Buffer[0]);
            LCDGotoXY(1,2);
            LCDWriteConstString("                    ");
         }
         else
         {
            csprintf(&Buffer[0]," Inductor: xxx uH        \0");
            LCDGotoXY(1,1);
            LCDWriteString(&Buffer[0]);

         }
      }
   }
}

/*****************************************************************************
```

```
Name:         void Calibration(void)

Description:  Calibration

Input:        none

Output:       none

Misc:

*******************************************************************************/
void Calibration(void)
{
   uint i,j;

   LCDClrSCR();
   LCDGotoXY(1,1);
   LCDWriteConstString("    Calibration\0");
   LCDGotoXY(1,2);
   LCDWriteConstString("    Please Wait\0");

   RELAY_PORT |= CAP_REF;
   for (i=0;i<2;i++) f1 = (float)MesureLC();

   RELAY_PORT &= ~CAP_REF;
   for (i=0;i<2;i++) f2 = (float)MesureLC();

   RELAY_PORT |= CAP_REF;
   LCDClrSCR();
}

/*******************************************************************************

Name:         void SerialInit()

Description:  Initialize Serial Port

Input:        none

Output:       none

Misc:

*******************************************************************************/
void SerialInit(void)
{
   UBRRH = ((XTAL / (8 * 250000)) - 1)>>8;   // 250kbps at 16Mhz
   UBRRL = (XTAL / (8 * 250000)) - 1;
    UCSRA = (1<<U2X);
    UCSRB = (1<<TXEN);
    UCSRC = (1<<URSEL) + (1<<UCSZ1) + (1<<UCSZ0);
}

/*******************************************************************************

Name:         void CounterInit()

Description:  Initialize Counter

Input:        none

Output:       none

Misc:

*******************************************************************************/
void CounterInit(void)
{
    TIMSK |= (1<<OCIE1A);
    OCR1A = 0xffff;
}

/*******************************************************************************

Name:         ulong MesureLC()

Description:  Mesure LC Value

Input:        none

Output:       none

Misc:
```

```
*****************************************************************************/
ulong MesureLC(void)
{
 int i;

 TCCR1B = 0;
 CountHigh = 0;
 TCNT1 = 0;

 for (i=0;i<25000;i++)
 {
  TCCR1B = (1<<CS12) + (1<<CS11) + (1<<CS10);
  UDR = 0x55;
  while(!(UCSRA & (1<<UDRE)));
 }
 while(!(UCSRA & (1<<TXC)));
 TCCR1B = 0;

 CountLow = TCNT1;
 Count = CountLow + (CountHigh << 16);

 return Count;
}

/*****************************************************************************

Name:         void HighByteCount(void)

Description:  High Byte Counter

Input:        none

Output:       none

Misc:

*****************************************************************************/
#pragma interrupt_handler HighByteCount:iv_TIMER1_COMPA
void HighByteCount(void)
{
   CountHigh++;
}

//*****************************************************************************
// LCD Code
//*****************************************************************************

/*****************************************************************************

Name:         void LCDClrSCR(void)

Description:  Clear the LCD

Input:        none

Output:       none

Misc:

*****************************************************************************/
void LCDClrSCR()
{
  LCDWriteData(LCD_CTRL,0x01);      // Clear display
  LCDDelay50us(100);
}

/*****************************************************************************

Name:         LCDGotoXY(ushort x, ushort y)

Description:  Position cursor on the LCD at X & Y location

Input:        X -> X position on the LCD
              Y -> Y position on the LCD

Output:       none

Misc:

*****************************************************************************/
void LCDGotoXY(ushort x,ushort y)
{
  ushort address;

  x--;
```

```
  if (MaxY < 3)
  {
    switch(y)
    {
      case '\x01' :
      address = 0 + x;
      break;
      case '\x02' :
      address = 64 + x;
      break;
    }
  }
  else
  {
    switch(y)
    {
      case '\x01' :
      address = 0 + x;
      break;
      case '\x02' :
      address = 64 + x;
      break;
      case '\x03' :
      address = 20 + x;
      break;
      case '\x04' :
      address = 84 + x;
      break;
    }
  }
  LCDWriteData(LCD_CTRL,address | 0x80);
}

/*****************************************************************************

Name:         void LCDWriteString(char *ptr)

Description:  Write a string from RAM on the LCD

Input:        string pointer

Output:       none

Misc:

*****************************************************************************/
void LCDWriteString(char *ptr)
{
  ushort i;

  for (i=1;i<21;i++)
  {
    if (*ptr == 0x00) break;
    LCDWriteData(LCD_DATA,*ptr++);
  }
}

/*****************************************************************************

Name:         void LCDWriteConstString(const char *ptr)

Description:  Write a constant string on the LCD

Input:        string pointer

Output:       none

Misc:

*****************************************************************************/
void LCDWriteConstString(flash char *ptr)
{
  ushort i;

  for (i=1;i<21;i++)
  {
    if (*ptr == 0x00) break;
    LCDWriteData(LCD_DATA,*ptr++);
  }
}

/*****************************************************************************

Name:         void LCDWriteData(ushort rs, ushort ch)
```

```
Description:  Write a byte in rs of the LCD

Input:        rs -> Register select
              ch -> byte to write

Output:       none

Misc:

*******************************************************************************/
void LCDWriteData(ushort rs,ushort ch)
{
  LCDWait();

   LCD_PORT = 0x00;
  if ((ch & 0x80) == 0x80) LCD_PORT |= LCD_D7;
  if ((ch & 0x40) == 0x40) LCD_PORT |= LCD_D6;
  if ((ch & 0x20) == 0x20) LCD_PORT |= LCD_D5;
  if ((ch & 0x10) == 0x10) LCD_PORT |= LCD_D4;
  if (rs == 1) LCD_PORT |= LCD_RS;
  LCDDelay50us(1);
  LCD_PORT |= LCD_E;
  LCD_PORT &= ~LCD_E;

   LCD_PORT = 0x00;
  if ((ch & 0x08) == 0x08) LCD_PORT |= LCD_D7;
  if ((ch & 0x04) == 0x04) LCD_PORT |= LCD_D6;
  if ((ch & 0x02) == 0x02) LCD_PORT |= LCD_D5;
  if ((ch & 0x01) == 0x01) LCD_PORT |= LCD_D4;
  if (rs == 1) LCD_PORT |= LCD_RS;
  LCDDelay50us(1);
  LCD_PORT |= LCD_E;
  LCD_PORT &= ~LCD_E;
}

/*******************************************************************************

Name:         void LCDWait(void)

Description:  wait for the LCD to be ready

Input:        none

Output:       none

Misc:

*******************************************************************************/
void LCDWait(void)
{
  LCDDelay50us(1);
}

/*******************************************************************************

Name:         void LCDDelay50us(int Delay)

Description:  Delay of 50 us with a 16Mhz resonator

Input:        Delay X x 50us

Output:       none

Misc:

*******************************************************************************/
void LCDDelay50us(int Delay)
{
  int i,j;

  for (i=0;i<Delay;i++)
  {
    for (j=1;j<(XTAL/133333);j++);
    asm("WDR");
  }
}

/*******************************************************************************

Name:         void LCDInit(ushort X, ushort Y)

Description:  Initialize LCD in 4bit mode

Input:        ushort X -> X size
```

```
              ushort Y -> Y size

Output:       none

Misc:

*******************************************************************************/
void LCDInit(ushort X, ushort Y)
{
  MaxX = X;
  MaxY = Y;
  LCD_DDR = LCD_D7+LCD_D6+LCD_D5+LCD_D4+LCD_E+LCD_RS;
  LCD_PORT = ~(LCD_D7+LCD_D6+LCD_D5+LCD_D4+LCD_E+LCD_RS);
  LCDDelay50us(340);

  LCD_PORT = (LCD_D5 + LCD_D4);              // Function Set 8 bit 3 time
  LCD_PORT |= LCD_E;
  LCD_PORT &= ~LCD_E;
  LCDDelay50us(100);
  LCD_PORT |= LCD_E;
  LCD_PORT &= ~LCD_E;
  LCDDelay50us(100);
  LCD_PORT |= LCD_E;
  LCD_PORT &= ~LCD_E;
  LCDDelay50us(100);

  LCD_PORT = LCD_D5;                         // Function Set 4 bit
  LCD_PORT |= LCD_E;
  LCD_PORT &= ~LCD_E;
  LCDDelay50us(100);

  if (Y == 1) LCDWriteData(LCD_CTRL,0x20);   // 1 line
  else LCDWriteData(LCD_CTRL,0x28);          // 2 line
  LCDWriteData(LCD_CTRL,0x0c);               // Disp ON-Cur OFF-Blink OFF
  LCDWriteData(LCD_CTRL,0x01);               // Clear display
  LCDDelay50us(50);
  LCDWriteData(LCD_CTRL,0x06);               // Cursor INC Shift OFF
}
```