MAX32, SainSmart I2C Serial 2004 20x4 LCD Module and the Speed of Sound

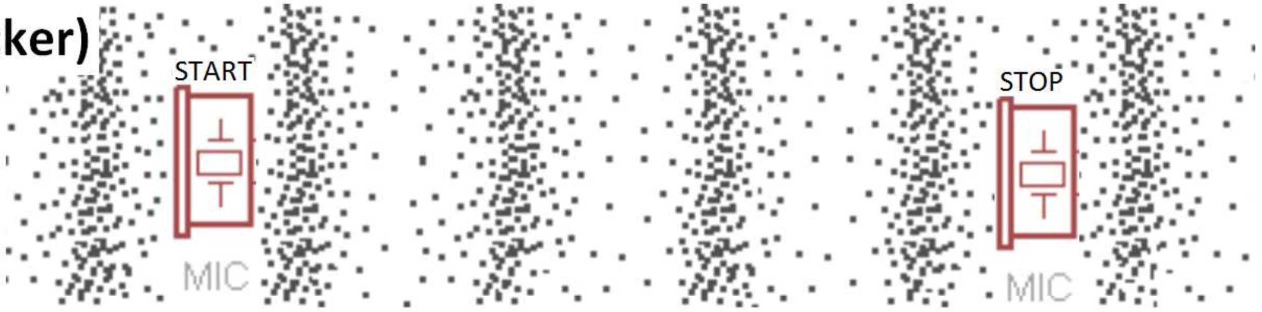
We measure the speed of sound in air by using the chipKIT Max32, an I2C-LCD and two electret microphones to start and stop (read out) 32-bit timer 23.

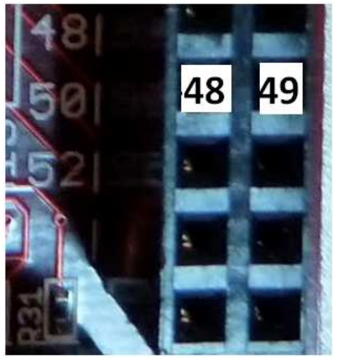
The idea …

MAX32
J9-48

MAX32
J9-49

Click (Frog Clicker)





```
void __ISR(_INPUT_CAPTURE_1…
 {
  if(n == 0) {
    n++;
    TMR2 = 0;//WriteTimer23(0); is TMR2 = 0;
    dt = 0;
  }
  mIC1ReadCapture();   mIC1ClearIntFlag();
 }
```

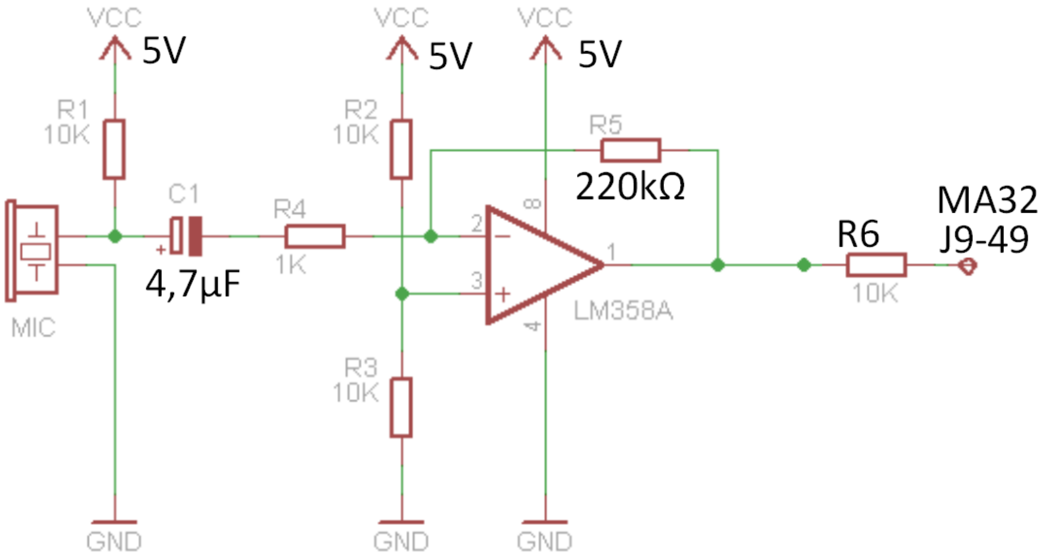
```
void __ISR(_INPUT_CAPTURE_4…
{
  if(n == 1){
    dt = mIC4ReadCapture();
    n++;
  } else mIC4ReadCapture();
  mIC4ClearIntFlag();
}
```

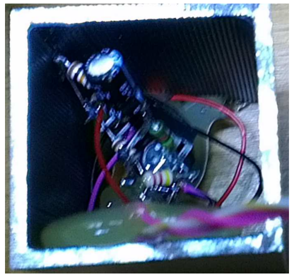
|<-------------------------------------ds--->|

Speed: $v = \frac{ds}{dt}$

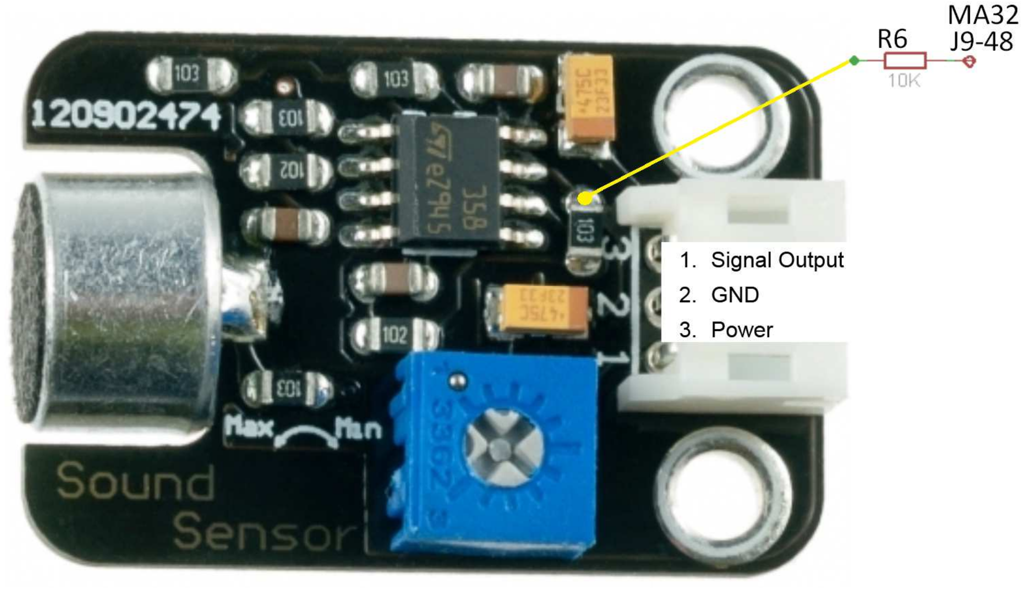
There are other ways like standing waves of longitudinal waves to measure the speed of sound. Take it as an exercise.

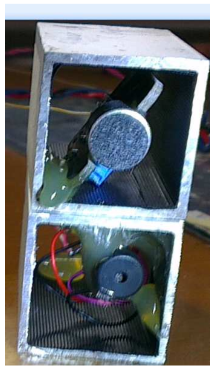
We assembled and soldered one sound sensor according to that schematic.
The MIC is an electret microphone.

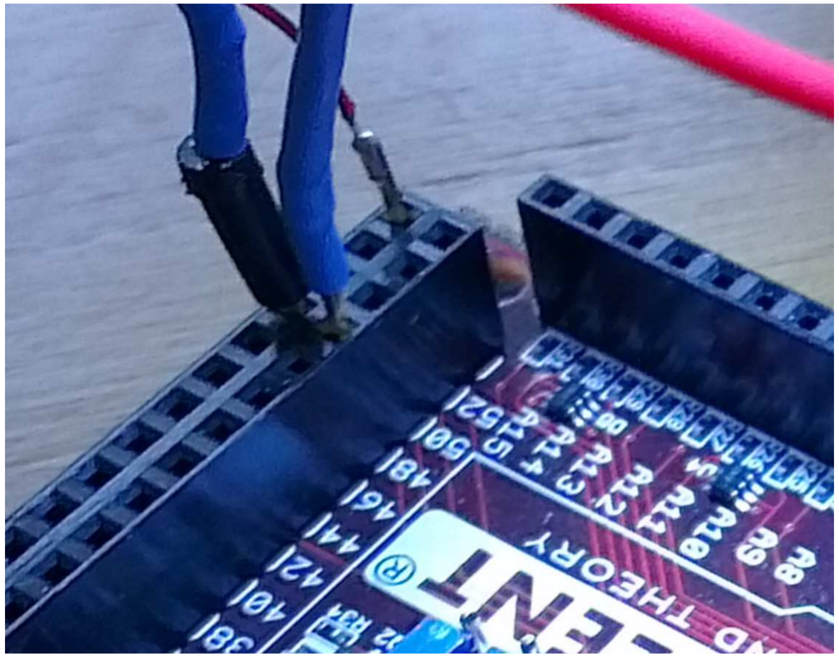




The other sound sensor is store-bought. (SKU: DFR0034)
The breakout board couples the electret microphone with the same opamp LM358 to amplify the sound. But we use another output.

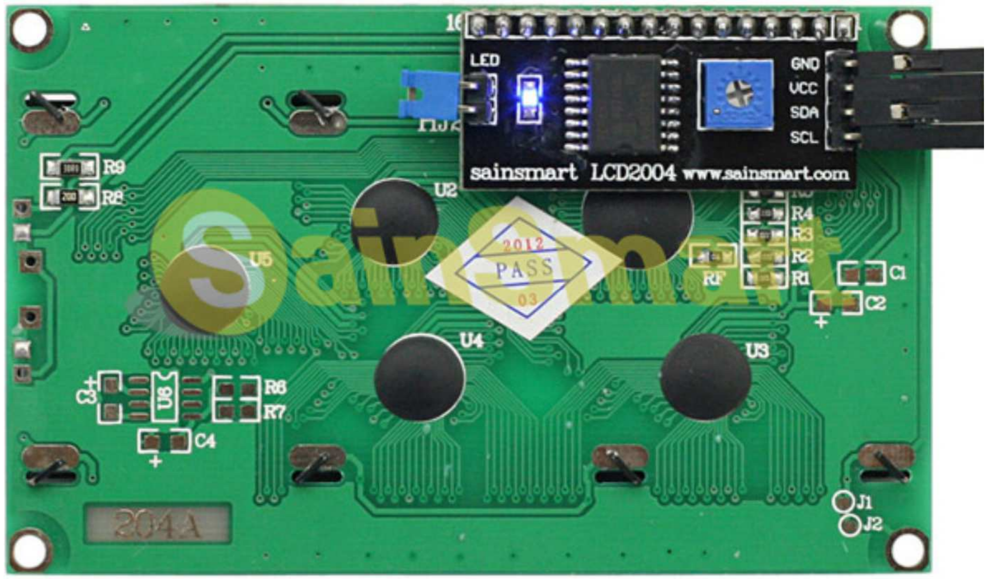




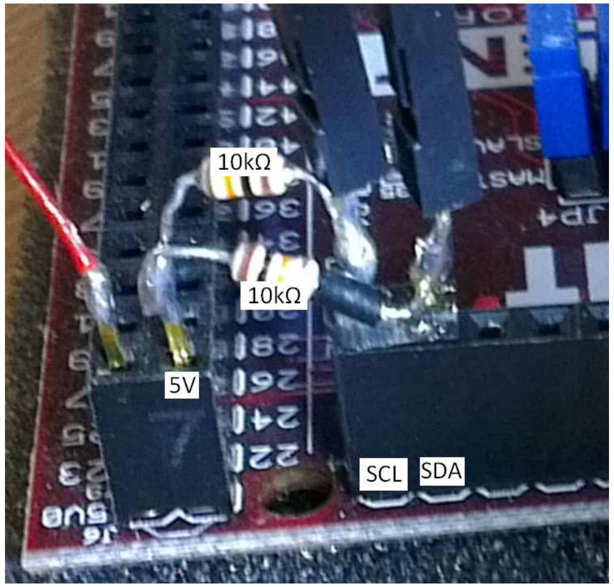


Input Capture part: IC1(48) and IC4(49) and a 32-bit timer (Timer 2, 3)

The microphones with aluminum case



Backside of the 20x4 LCD with I2C-module



MAX32 needs power for the SCL- and SDA-wire

From timer values to microseconds …

$$dt1 = \frac{prescaler \cdot TMR}{f_{CLOCK}} \underset{example}{=} \frac{1 \cdot 163840}{80000000 \cdot \frac{1}{s}} = 0{,}002048 \cdot s = 2048\,\mu s$$

$$ds := 0.2 \cdot m \qquad dt := 604 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 331.126\,\frac{m}{s}$$

$$ds := 0.25 \cdot m \qquad dt := 746 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 335.121\,\frac{m}{s}$$

$$ds := 0.3 \cdot m \qquad dt := 895 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 335.196\,\frac{m}{s}$$

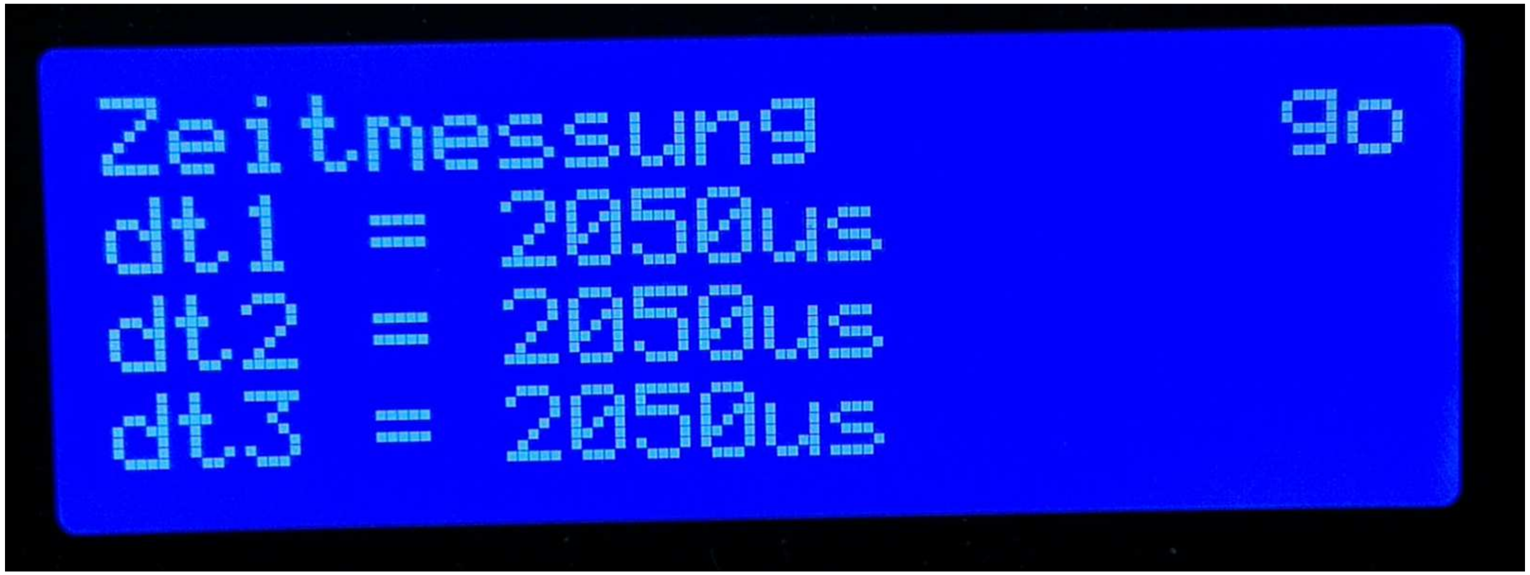
$$ds := 0.4 \cdot m \qquad dt := 1183 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 338.123\,\frac{m}{s}$$

$$ds := 0.5 \cdot m \qquad dt := 1472 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 339.674\,\frac{m}{s}$$

$$ds := 0.6 \cdot m \qquad dt := 1764 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 340.136\,\frac{m}{s}$$

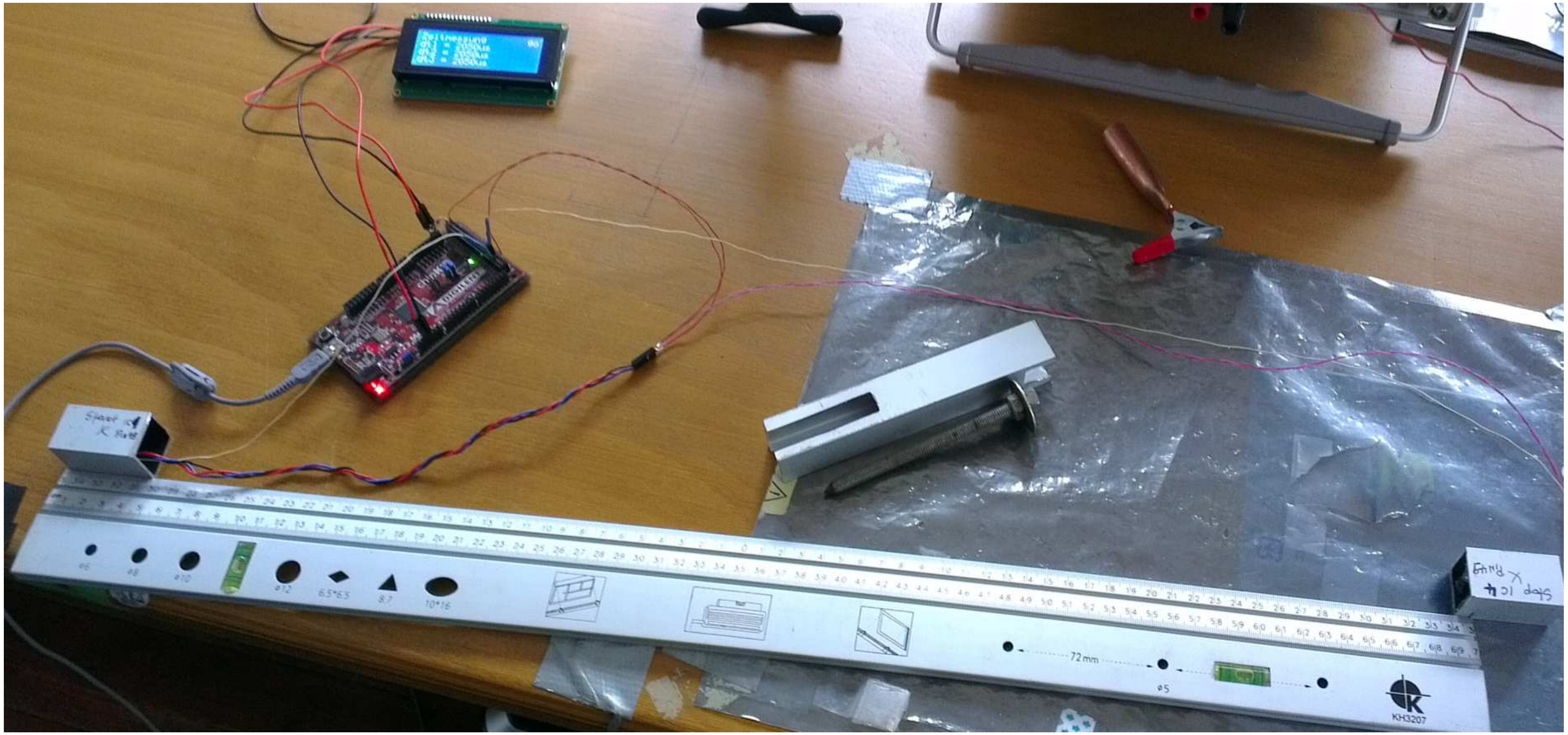
$$ds := 0.7 \cdot m \qquad dt := 2048 \cdot 10^{-6} \cdot s \qquad v := \frac{ds}{dt} \qquad v = 341.797\,\frac{m}{s}$$

```
void __ISR(_INPUT_CAPTURE_4_VECTOR,IPL2AUTO/*ipl2*/) my_capture4(void)
 {
  if(n == 1){

   dt = mIC4ReadCapture();//is: dt = ReadTimer23(); ← TMR

   n++;
  } else mIC4ReadCapture();
  mIC4ClearIntFlag();
 }
```



```
Zeitmessung     go
dt1 = 2050us
dt2 = 2050us
dt3 = 2050us
```

SainSmart I2C Serial 2004 20x4 LCD Module



The workbench …

The whole program …

```
#if defined(__PIC32MX__)
#include <p32xxxx.h> /* this gives all the CPU/hardware definitions */
#include <plib.h> /* this gives the i/o definitions */
#include <Wire.h>
#endif

#define LCD_DISPLAYON 0x04
#define LCD_DISPLAYOFF 0x00
#define LCD_CURSORON 0x02
#define LCD_CURSOROFF 0x00
#define LCD_BLINKON 0x01
#define LCD_BLINKOFF 0x00
#define LCD_DISPLAYCONTROL 0x08
#define LCD_CLEARDISPLAY 0x01
#define LCD_ENTRYMODESET 0x04
#define LCD_ENTRYLEFT 0x02
#define LCD_ENTRYSHIFTDECREMENT 0x00
#define LCD_RETURNHOME 0x02
#define LCD_SETDDRAMADDR 0x80
#define LCD_SETCGRAMADDR 0x40
#define LCD_BACKLIGHT 0x08
#define LCD_NOBACKLIGHT 0x00

uint8_t n = 0;
unsigned long dt1=0,dt2=0,dt3=0,dt=0;
double dt_ys;
String str;

void setup()
{
  mIC1ClearIntFlag();
  mIC4ClearIntFlag();
  OpenTimer23(T2_ON | T2_32BIT_MODE_ON | T2_PS_1_1,4000000000);
  ConfigIntCapture1(IC_INT_ON | IC_INT_PRIOR_2);
  OpenCapture1(IC_INT_1CAPTURE | IC_CAP_32BIT | IC_TIMER2_SRC | IC_EVERY_EDGE | IC_ON ); //IC_EVERY_EDGE is IC1CONbits.ICM == 0b001
  ConfigIntCapture4(IC_INT_ON | IC_INT_PRIOR_2);
  OpenCapture4(IC_INT_4CAPTURE | IC_CAP_32BIT | IC_TIMER2_SRC | IC_EVERY_EDGE | IC_ON ); //IC_EVERY_EDGE is IC1CONbits.ICM == 0b001
  INTEnableSystemMultiVectoredInt();
  Wire.begin(); // join i2c bus (address optional for master)
  init_LCD();
  send_string(1,1,"Zeitmessung");
}

void loop(){
  if(n>1){
    dt_ys = dt/80.;//dt = prescaler*TMR2/Freq (Unit: s)
    dt3=dt2;
    dt2=dt1;
    dt1=(unsigned int)dt_ys;
    str =  String(dt1);
    send_string(1,2,"                    ");
    send_string(1,2,"dt1 = " + str + "us");
    str =  String(dt2);
    send_string(1,3,"                    ");
    send_string(1,3,"dt2 = " + str + "us");
    str =  String(dt3);
    send_string(1,4,"                    ");
    send_string(1,4,"dt3 = " + str + "us");
    send_string(17,1,"stop");
    delay(3000);
    n = 0;
  } else send_string(17,1,"  go");
}
```

```
#ifdef __cplusplus
extern "C" {
#endif

  void __ISR(_INPUT_CAPTURE_1_VECTOR,IPL2AUTO/*ipl2*/) my_capture1(void)
  {
    if(n == 0) {
      n++;
      TMR2 = 0;//WriteTimer23(0); is TMR2 = 0;
      dt = 0;
    }
    mIC1ReadCapture(); //1: it's a must; it's essential to get timerdata
    mIC1ClearIntFlag();
  }

  void __ISR(_INPUT_CAPTURE_4_VECTOR,IPL2AUTO/*ipl2*/) my_capture4(void)
  {
    if(n == 1){
      dt = mIC4ReadCapture();//is: dt = ReadTimer23();
      n++;
    } else mIC4ReadCapture();
    mIC4ClearIntFlag();
  }

#ifdef __cplusplus
}
#endif

void Tr_send(uint8_t data){
     Wire.beginTransmission(0x27);
       Wire.send(data | 0x08);//0x08 is backlight on
     Wire.endTransmission();
}
```

```
void send_enable(uint8_t data){
     Tr_send(data | 0b00000100);    // En high
     delayMicroseconds(1);          // enable pulse must be > 450ns
     Tr_send(data & 0b11111011);   // En low
     delayMicroseconds(50);         // commands need > 37us to settle
}

void send_PCF(uint8_t value) {
     Tr_send(value);
     send_enable(value);
}

void send_4_4(uint8_t value, uint8_t mode) {
     uint8_t highnib=value&0xf0;
     uint8_t lownib=(value<<4)&0xf0;
     send_PCF((highnib)|mode);
     send_PCF((lownib)|mode);
}

void send_command(uint8_t value) {
     send_4_4(value, 0);
}

void send_data(uint8_t value) {
     send_4_4(value, 1);
}

//void send_string(char *str) {
void send_string(uint8_t col, uint8_t row, String str) {
  set_Cursor(col, row);
  for(uint8_t i = 0; i < str.length(); ++i){
      send_data(str[i]);
  }
}
```

```
void set_Cursor(uint8_t col, uint8_t row){
  int row_offsets[] = { 0x00, 0x40, 0x14, 0x54 };
  send_command(LCD_SETDDRAMADDR | (col - 1 + row_offsets[row - 1]));
}

// Allows us to fill the first 8 CGRAM locations with custom characters
// Print them with send_data(n); 0<=n<=7
void create_char(uint8_t location, uint8_t charmap[]) {
     location &= 0x7; // we only have 8 locations 0-7
     send_command(LCD_SETCGRAMADDR | (location << 3));
     for (int i=0; i<8; i++) {
              send_data(charmap[i]);
     }
}

// Turn the (optional) backlight off/on
void Backlight_off(void) {
       Wire.beginTransmission(0x27);
     Wire.send(LCD_NOBACKLIGHT);
     Wire.endTransmission();
}

void Backlight_on(void) {
     Wire.beginTransmission(0x27);
     Wire.send(LCD_BACKLIGHT);
     Wire.endTransmission();
}

void init_LCD(void){
  delay(50);
  Tr_send(0);// reset and turn backlight off
  delay(1000);
  send_PCF(0x03 << 4);
  delayMicroseconds(4500); // wait min 4.1ms
  send_PCF(0x03 << 4);
  delayMicroseconds(4500); // wait min 4.1ms
  send_PCF(0x03 << 4);
  delayMicroseconds(150);
  send_PCF(0x02 << 4);
  send_command(0x20 | 0x08);
  send_command(LCD_DISPLAYCONTROL | LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF);
  send_command(LCD_CLEARDISPLAY);// clear display, set cursor position to zero
  delayMicroseconds(2000);
  send_command(LCD_ENTRYMODESET | LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT);
  send_command(LCD_RETURNHOME);  // set cursor position to zero
  delayMicroseconds(2000);  // this command takes a long time!
}
```

… have fun!

edgarmarx@t-online.de