Respiratory motion synchroniser in magnetic resonance imaging

1. Introduction

When imaging small animals (or indeed larger species!) respiratory motion is a very undesirable artefact. There are many ways of dealing with this but an obvious and effective method is to stop imaging (or delete images) when large rates of change in the breathing pattern are taking place. Anaesthetized animals have a highly non-linear breathing pattern and are effectively still during a large portion of the breathing period, typically lasting foe several hundred milliseconds. The breathing pattern can be sensed in several ways, but in general some form of pressure transducer is used. The signal form this transducer can be passed thorough a threshold circuit and a logic pulse can be derived, corresponding to periods when the animal is still and when inhalation and exhalation is taking place. The device described here can be used to isolate the 'still' period, as shown in Figure 1. The idea is to provide a short period after the breathing pulse to inhibit imaging and to set a timer which allows imaging for a set period. Should a breathing pulse arrive during this second timed period, imaging is inhibited and a minimal amount of breathing artefact in the resulting image is then unavoidable.

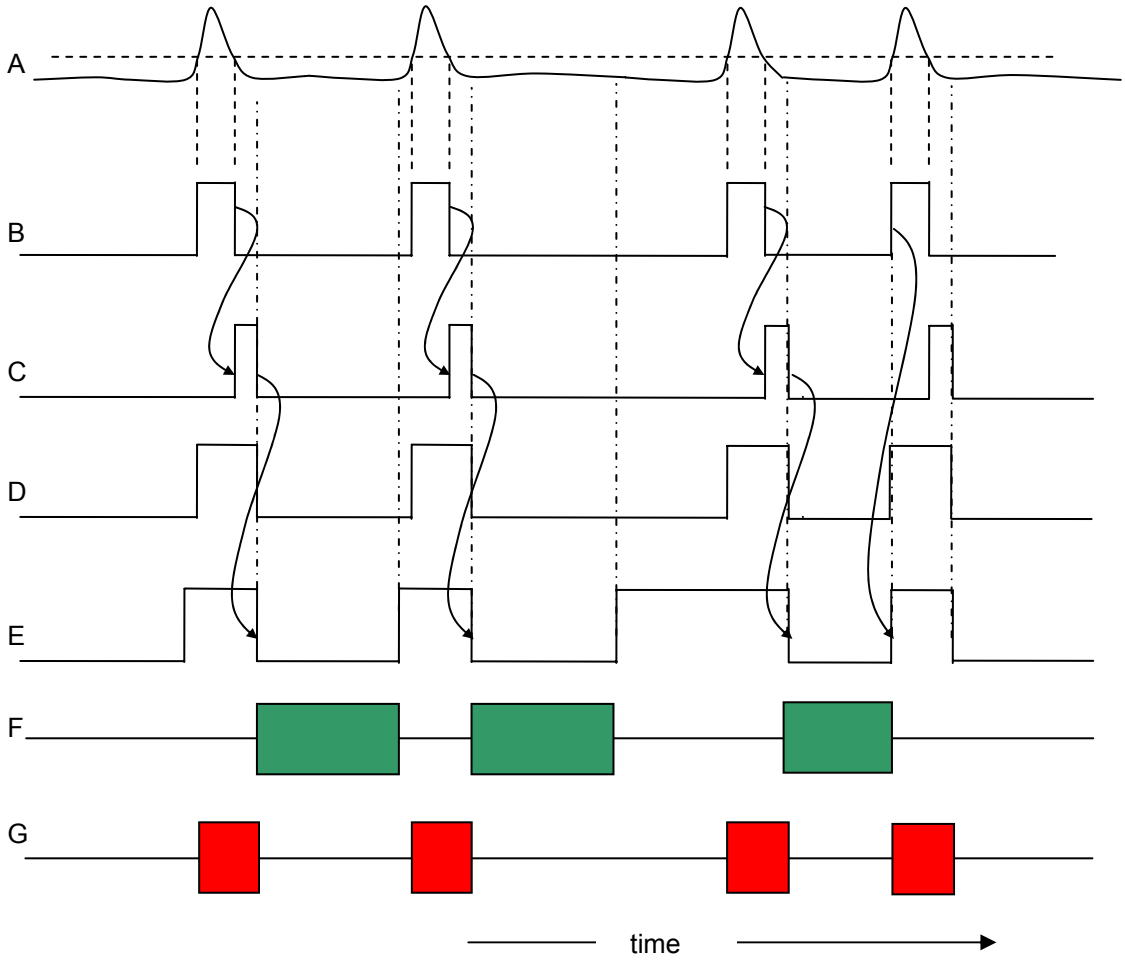

Figure 1. Waveforms associated with the respiratory motion synchroniser. Waveform A is shown for completeness, representing the respiratory motion as sensed by a pressure transducer. Waveform B represents a thresholded logic output derived from A and forms the input trigger signal to the device described here. Waveform C represents the output of a timer triggered by the falling edge of waveform C; the falling edge of that signal triggers a second internal timer, the output of which is shown in waveform E. Waveform D is an internal; signal formed by 'OR'ing B and C and used to illuminate a 'red' indicator, as shown by trace G. The output shown at E is used to gate the imaging device and to turn on a 'green' LED as shown in trace F.

The first period in Figure 1 shows the relative timings when timer E is adjusted correctly; the second period shows the situation when the next breathing cycle is late; the third period shows the situation when the next breathing cycle is early. In this last situation, some image degradation is unavoidable.

We developed this device for use with the Institute's magnetic resonance imaging scanners, though it could be used with other forms of imaging instruments. The design was inspired through discussions with S Smart at the Radiobiology Research Institute.

2. Circuit design

The circuit of the synchronizer is shown in Figure 2. The system could have been constructed using a pair of monostables a a few gates. However there are good reasons why a PIC microcontroller has advantages. Firstly, the 'OK to image' delay time, trace F in figure 1 could be several hundred milliseconds long and needs to be adjusted carefully and precisely during the imaging experiment: a 10 turn potentiometer is thus suggested. Since such potentiometers cannot be readily obtained in values >100 kΩ, large capacitance values would be required. Furthermore, we may wish to use complementary inputs and complementary outputs in different situations and this would increase the chip count required had the circuit been built from HC or HCT logic devices.

We thus settled on the design shown in Figure 2. This makes use of the analogue inputs of the PIC to digitize the settings of the delay time potentiometers. The PIC drives a pair of LEDs and allows changes to the code should some other logic pattern be required in the future. A BNC trigger input is provided and outputs to the imaging device and to a monitoring oscilloscope are provided on two further BNC connectors. A small +5V wall plug power supply is used to provide DC power.

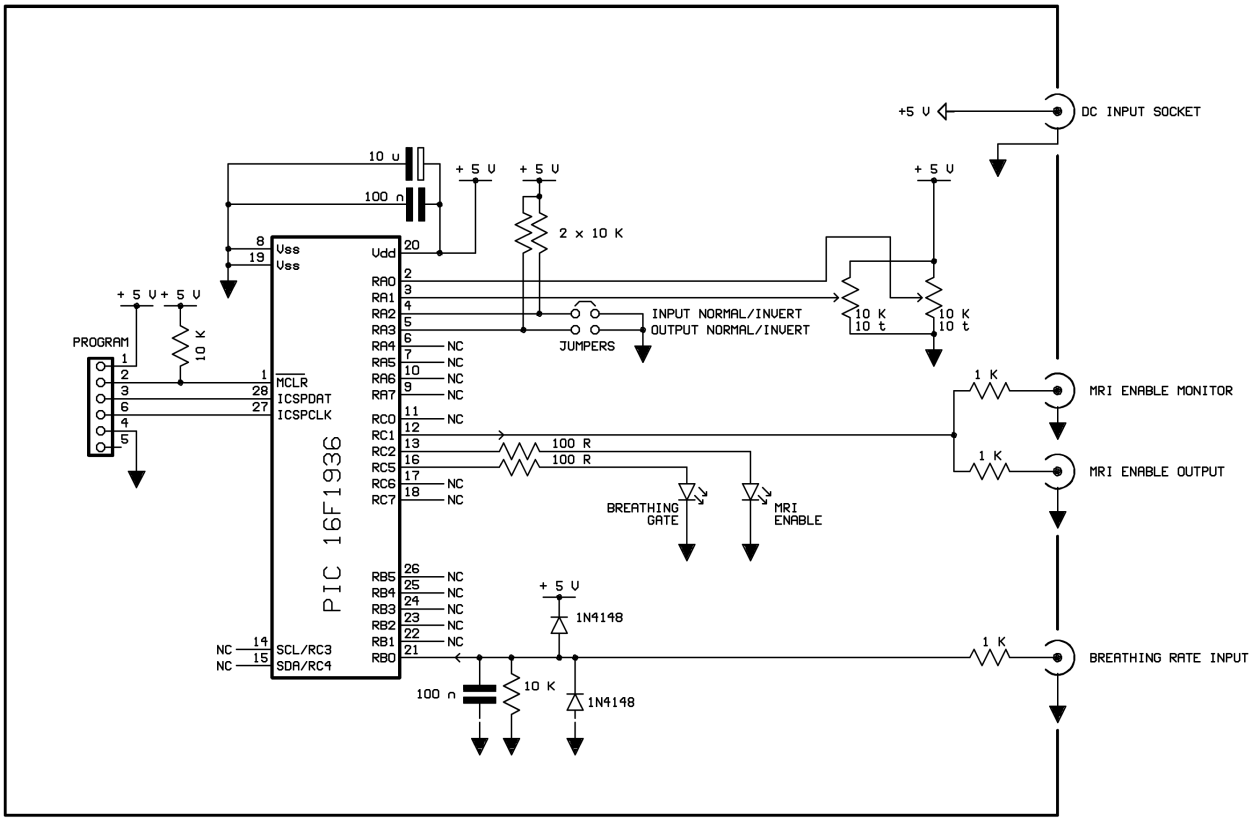

Figure 2. Circuit diagram of the breathing rate synchronizer.

3. Construction details

The device is constructed on a small piece of prototype board and is housed in a small plastic box, as shown in Figures 3 and 4.

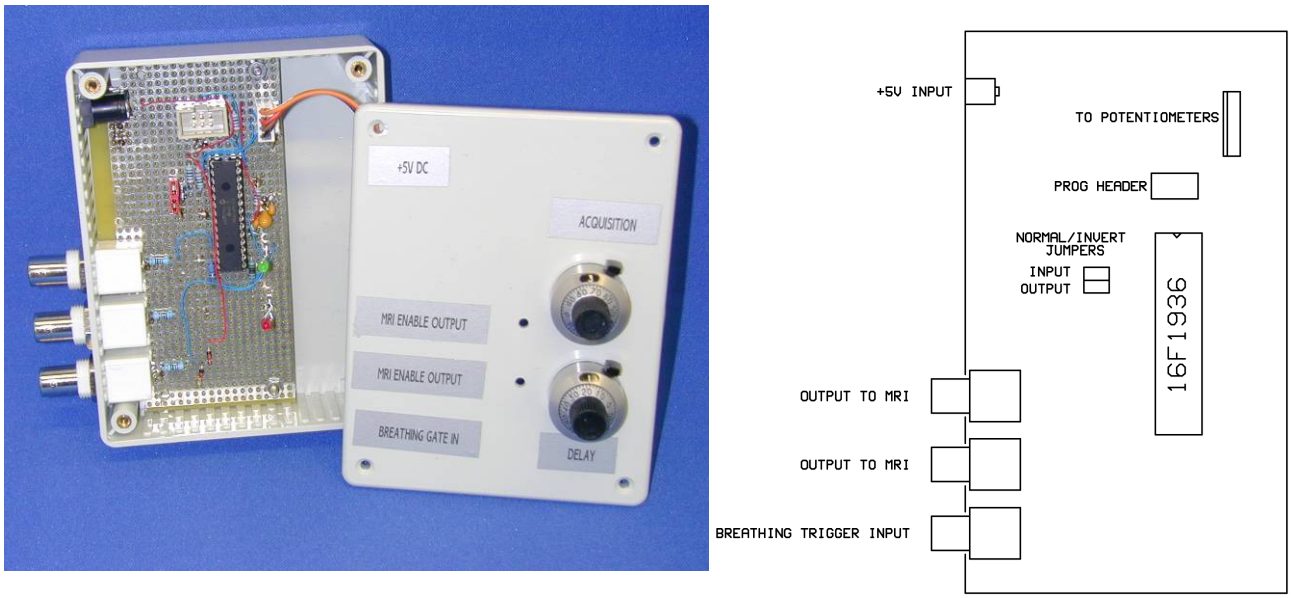

Figure 3. Construction of the breathing rate synchronizer. An internal view of the unit is shown on the left and the component layout is shown on the right of the figure.

Figure 4. The completed breathing rate synchronizer.

4. PIC firmware

The firmware was written using the CCS C-code compiler (http://www.ccsinfo.com/) which makes generating the code considerably easier than using an assembler code language such as Microchip MPLAB. The sample code below may be found useful should future modifications be required. It is extremely simple and largely self-explanatory. The code 'reads' the positions of the two 10 turn potentiometers by digitising with the PIC's 10 bit converters. Note that the code has been designed to run with 32 MHz clock frequency PICs. The code is blown into the PIC and no external control is catered for; however external programming through the PIC's serial or I2C ports can be readily added if required.

```
////////////////////////////////////////////////
//    PCM,PCH    4.101 compiler                //
//    MRI breathing rate                       //
//    25/07/2011                               //
//                                             //
////////////////////////////////////////////////

#include "MRI breathing rate.h"

#byte PIC_IOCBF=0x396                                   //Register of IOC flags

int1 trigger_level,output_level;
int16 delay_time=0,pulse_length=0,timer0_count=0,timer1_count=0;

 #int_TIMER0
void  TIMER0_isr(void)
{
      timer0_count++;
       set_timer0(15);
   if(timer0_count>=delay_time){
      output_bit( PIN_C0, (output_level^0x01));
      output_bit( PIN_C1, (output_level));              //Set imaging pulse
      set_timer1(0);                                    //Clear counter
      enable_interrupts(INT_TIMER1);                    //Enable timer1
      disable_interrupts(INT_TIMER0);                   //Disable timer0
      timer0_count=0;                                   //Reset counter
      timer1_count=0;
      output_bit( PIN_C2, 1);                           //LED indicator MRI enable
      output_bit( PIN_C5, 0);                           //LED indicator breathing gate off
   }
}
#int_TIMER1
void  TIMER1_isr(void)
{
      timer1_count++;
      set_timer1(50000);
   if(timer1_count>=pulse_length){
      output_bit( PIN_C1, (output_level^0x01));         //Reset imaging pulse
      disable_interrupts(INT_TIMER1);                   //Disable timer1
      timer1_count=0;                                   //Reset counter
      output_bit( PIN_C2, 0);                           //LED indicator MRI disable
   }
}
#int_RB             //Port B interrupt
void RB_isr(void)
{
int1 val;

val=input_state(PIN_B0);

if(val==trigger_level){                                 //Set output 1
   output_bit( PIN_C0, (output_level));
   output_bit( PIN_C1, (output_level^0x01));            //Reset imaging pulse
   output_bit( PIN_C5, 1);                              //LED indicator breathing gate on
   output_bit( PIN_C2, 0);                              //LED indicator MRI disnable

}
else{            //Start delay timer
      set_timer0(0);                                    //Clear counter
      enable_interrupts(INT_TIMER0);                    //Enable interrupt
}
   #asm
   BCF    0x396.0                                       //Clear RB0 flag
   BCF    0x0B.0                                        //Clear IOC flag
   #endasm
}
////////////////////////////////////////////////////////////////////////////
void read_pulse_widths(void)
{

set_adc_channel(0);                                     //Select channel 0 (RA0)
delay_us(20);                                           //Allow switching time
delay_time = read_adc();
set_adc_channel(1);                                     //Select channel 1 (RA1)
delay_us(20);                                           //Allow switching time
pulse_length = read_adc();

}
//*************************************************************************
//Init() - routine
//*************************************************************************
void Init(void)
{
```

```
 trigger_level=input_state(PIN_A2);                    //Set input edge trigger high or low
 output_level=input_state(PIN_A3);                     //Set output pulses high or low
 output_bit( PIN_C0, (output_level^0x01));             //Invert
 output_bit( PIN_C1, (output_level^0x01));             //Invert
 output_bit( PIN_C2, 0);                               //LED indicator MRI enable
 output_bit( PIN_C5, 0);                               //LED indicator breathing gate on

 SETUP_ADC_PORTS(sAN0 | sAN1);                         //Set RA0 and A1 as analogue inputs
 setup_adc (ADC_CLOCK_DIV_16);

 setup_timer_0(RTCC_INTERNAL|RTCC_DIV_16);             //Setup timer0
 setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);               //Setup timer1
 disable_interrupts(INT_TIMER0);
 disable_interrupts(INT_TIMER1);
 set_timer0(0);
 set_timer1(0);
}

//-----------------------------------------------------------------------------------------

void main()
{

 Init();                                               //Initialize 16F1936 Microcontroller
 enable_interrupts(GLOBAL);


#asm
BSF    0x0B.3                                         //enable interrupt on change port B and
BSF    0x395.0                                        //set bits 0,1,2,3 in register IOCBN for negative edge
BSF    0x394.0                                        //set bits 0,1 in register IOCBP
#endasm

while(1){
          read_pulse_widths();
 }
}
```

MRI breathing rate.h header file:

```
#include <16F1936.h>
#FUSES MCLR                                           //Master Clear pin enabled
#FUSES DEBUG                                          //Debug mode for use with ICD
#device ICD=TRUE                                      //Comment out these two lines when not debugging
#device adc=10                                        //10 bit A/D conversion
#device *=16                                          //Increase memory
#FUSES NOWDT                                          //No Watch Dog Timer
#FUSES PLL                                            //If enabled PLL multiplies clock by 4 **THIS FOR 32MHz**
#FUSES INTRC_IO                                       //Internal RC Osc, no CLKOUT           **THIS FOR 32MHz**
#FUSES NOLVP                                          //No low voltage programing, B3(PIC16) or B5(PIC18) used for I/O
#use delay(clock=32000000)
```

This note was prepared in August 2011 by B. Vojnovic and R.G. Newman, who also constructed the unit and programmed it. The basic need for this device was suggested by Dr S. Smart, at the Gray Institute small animal imaging facility.

We acknowledge the financial support of Cancer Research UK, the MRC and EPSRC.