

# Propeller Ultrasonic VGA Demo

Contributed by RichardJones  
 Thursday, 21 February 2008  
 Last Updated Friday, 22 February 2008

' Propeller Ultrasonic Send and Receive with display on a VGA Monitor

' by Richard Jones rjtp<atsym>ihug.co.nz

' V1.0 18th February 2008

' This program uses the Propeller Proto Board

' The ultrasonic sender is energised briefly at the beginning of the scope trace.

' The ultrasonic receive signal is digitized and the samples are displayed on a VGA monitor, just like an oscilloscope with trigger from the emitted pulse.

' This program is based on the Microphone to VGA demo from Parallax.

```

      3 +3v3
      -@1n
P0    K 0.1u
      5k6 # «
P1    1/2 3/4 K Receiver Sender \ P7
      -@1n O| MA40A3R MA40A3S []
      MuRata MuRata / P6
      0v ; ;
      A-D Convertor Sender
  
```

' Note 1. R & C's must be v close to propellor pins.

' Note 2. Ultrasonic sender and receiver from Surplustronics in Auckland

' Note 3. A->D conversion details may be found here:

<http://www.parallax.com/Portals/0/Downloads/apnt/prop/AN001-PropellerCountersv1.1.zip>

CON

```
_clkmode = xtal1 + pll16x
```

```
_xinfreq = 5_000_000
```

```
tiles = vga#xtiles * vga#ytiles
```

```
tiles32 = tiles * 32
```

```
apin_sonic = 6 ' ultrasonic output pin a
```

```
bpin_sonic = 7 ' ultrasonic output pin b (in antiphase to a)
```

OBJ

```
vga : "vga_512x384_bitmap"
```

VAR

```
long sync, pixels[tiles32]
```

```
word colors[tiles], ypos[512]
```

PUB start | i

```
'start vga
```

```
vga.start(16, @colors, @pixels, @sync)
```

```
'init colors to cyan on black
```

```
repeat i from 0 to tiles - 1
```

```
colors[i] := $3C00
```

```
'fill top line so that it gets erased by COG
longfill(@pixels, $FFFFFFF, vga#xtiles)
```

```
'implant pointers and launch assembly program into COG
asm_pixels := @pixels
asm_ypos := @ypos
cognew(@asm_entry, 0)
repeat
```

CON

```
' At 80MHz the ADC sample resolutions and rates are as follows:
```

```
' sample sample
' bits rate
'-----
' 9 156 KHz
' 10 78 KHz
' 11 39 KHz
' 12 19.5 KHz
' 13 9.77 KHz
' 14 4.88 KHz
```

```
bits = 9 'try different values from table here
attenuation = 0 'try 0-4
```

```
averaging = 13 '2-power-n samples to compute average with
```

DAT

```
'
'
' Assembly program
'
```

```
org
```

```
asm_entry mov dira,asm_dira_us 'make pin p1 (ADC) output
```

```
movs ctra,#0 'POS W/FEEDBACK mode for CTRA
movd ctra,#1
movi ctra,##01001_000
mov frqa,#1
```

```
mov xpos,#0
```

```
mov asm_cnt,cnt 'prepare for WAITCNT loop
add asm_cnt,asm_cycles
```

```
:loop waitcnt asm_cnt,asm_cycles 'wait for next CNT value (timing is determinant after WAITCNT)
```

```
mov asm_sample,phsa 'capture PHSA and get difference
sub asm_sample,asm_old
add asm_old,asm_sample
```

```
add average,asm_sample 'compute average periodically so that
djnz average_cnt,#:avgsame 'we can 0-justify samples
mov average_cnt,average_load
shr average,#averaging
mov asm_justify,average
mov average,#0 'reset average for next averaging
```

```
:avgsame
```

```

max    peak_min,asm_sample      'track min and max peaks for triggering
min    peak_max,asm_sample
djnz   peak_cnt,#:pkssame
mov    peak_cnt,peak_load
mov    x,peak_max              'compute min+12.5% and max-12.5%
sub    x,peak_min
shr    x,#3
mov    trig_min,peak_min
add    trig_min,x
mov    trig_max,peak_max
sub    trig_max,x
mov    peak_min,bignum        'reset peak detectors
mov    peak_max,#0
:pkssame
    cmp    xpos, #0 wz
if_nz  jmp    #:not_start

    ' Initiate Output of a few cycles at 40kHz on apin and inverted on bpin
    mov    dira,asm_dira_us    'set ultrasonic pins to output
    mov    phsb, #0
    mov    frqb, asm_frqb
    mov    ctrb, asm_ctrb
:not_start  cmp    xpos, #20 wz
if_nz  jmp    #:not_end

    ' set ultrasonic pins to open circuit
    mov    dira,asm_dira_oc
    mov    ctrb, #0 ' disable 40kHz output pins

:not_end  mov    mode,#1        'flag pulse sent
    sub    asm_sample,asm_justify 'justify sample to bitmap center y
    sar    asm_sample,#attenuation 'this # controls attenuation (0=none)
    add    asm_sample,#384 / 2
    mins   asm_sample,#0
    maxs   asm_sample,#384 - 1

    mov    x,xpos              'xor old pixel off
    shl    x,#1
    add    x,asm_ypos
    rdword y,x                'get old pixel-y
    wrword asm_sample,x        'save new pixel-y
    mov    x,xpos
    call   #plot

    mov    x,xpos              'xor new pixel on
    mov    y,asm_sample
    call   #plot

    add    xpos,#1            'increment x position and mask
    and    xpos,#$1FF wz
if_nz    jmp    #:loop
    mov    mode,#0            'if rollover, reset mode for trigger
    mov    xpos,#0

    jmp    #:loop            'wait for next sample period
,
,
' Plot
,
plot    mov    asm_mask,#1      'compute pixel mask
    shl    asm_mask,x
    shl    y,#6                'compute pixel address
    add    y,asm_pixels
    shr    x,#5
    shl    x,#2

```

```

    add    y,x
    rdlong asm_data,y          'xor pixel
    xor    asm_data,asm_mask
    wrlong asm_data,y

plot_ret  ret
,
,
' Data
,
asm_cycles long    |< bits - 1          'sample time
'asm_cycles long    380                  'sample time
asm_dira_us long    $000002c2           'output mask p0=adc ip,p1=adc op, p6,p7 ultrasonic output
asm_dira_oc long    $00000200
asm_pixels long    0                    'pixel base (set at runtime)
asm_ypos  long    0                    'y positions (set at runtime)
average_cnt long    1
peak_cnt  long    1
peak_load long    512
mode      long    0
bignum    long    $FFFFFFFF
average_load long    |< averaging

asm_frqb  long    $8000_0000 / 1000 'Set FRQ so PHS[xx] toggles @40kHz
asm_ctrb  long    ( %00101_000 << 23 ) + ( bpin_sonic << 9 ) + apin_sonic 'Establish mode and APIN / BPIN
asm_time  long    0
asm_delay long    10

asm_justify res    1
trig_min   res    1
trig_max   res    1
average    res    1
asm_cnt    res    1
asm_old    res    1
asm_sample res    1
asm_mask   res    1
asm_data   res    1
xpos       res    1
x          res    1
y          res    1
peak_min   res    1
peak_max   res    1

```