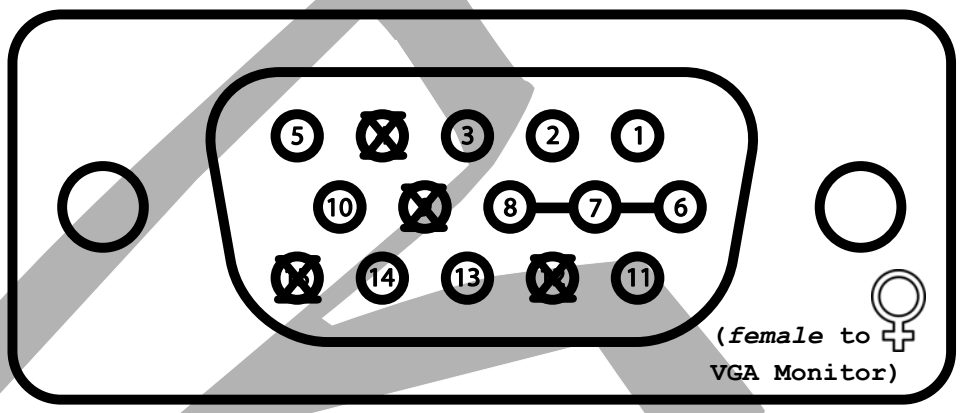
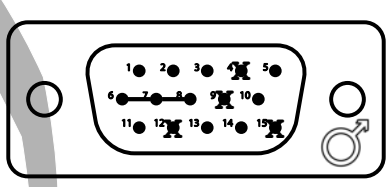
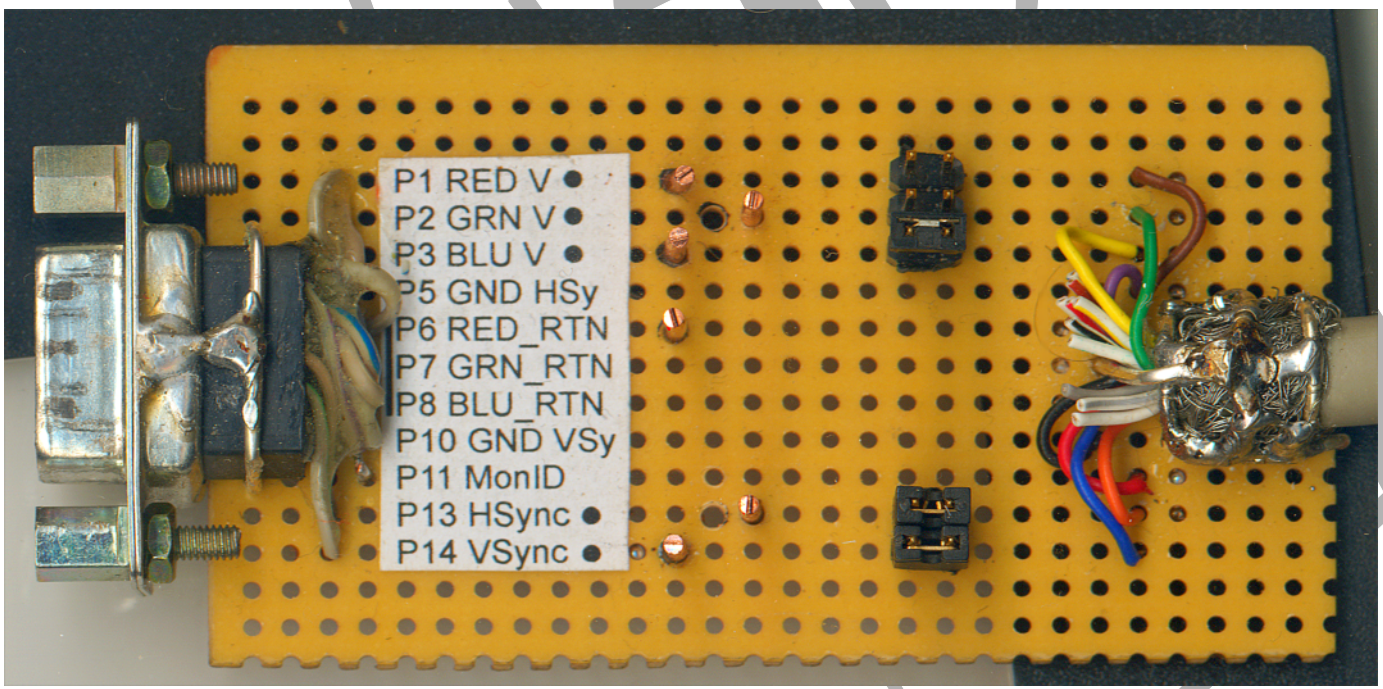



```
P1 RED V●●
P2 GRN V●●
P3 BLU V●●
P5 GND HSy
P6 RED_RTN
P7 GRN_RTN
P8 BLU_RTN
P10 GND VSy
P11 MonID
P13 HSync●●
P14 VSync●●
```

Pin 01 RED Red video
Pin 02 GREEN Green video <- PC senses CONNECT [to GND (75R)]
Pin 03 BLUE Blue video
X - Pin 04 N/C Not connected
Pin 05 GND Ground (HSync) -> Monitor senses CONNECT [to GND]
Pin 06 GND RED_RTN Red return
Pin 07 GND GREEN_RTN Green return
Pin 08 GND BLUE_RTN Blue return
X - Pin 09 SENSE +5 V DC from gfx adapter
Pin 10 GND Ground (VSync, DDC)
Pin 11 N/C Monitor ID
X - Pin 12 SDA I²C data
Pin 13 HSync Horizontal sync
Pin 14 VSync Vertical sync
X - Pin 15 SCL I²C clock





VGA - Standard: HSync : 31,47 kHz / VSync : 60 Hz

http://www.tinyvga.com/vga-timing/640x480@60Hz
http://en.wikipedia.org/wiki/Video_Graphics_Array

VGA Exp. "Horror Vacui"
Paper for xxxxx Workshop
Jo frgmnt Grys / tob.de.vu / 2009
Transmitting Object Behaviors

```
/*
 * VGA Sync Generator - Arduino 2014
 *
 * Based on :
 *  - code by Dwan (dwanafite@yahoo.fr)
 *  - RG Matrix Example v.2 8/1/08, by BroHogan, from the Arduino Playground
 *  - Simplest universal VGA/PAL terminal, by Ibragimov Maxim Rafikovich, http://www.vga-avr.narod.ru/main.html
 *  - the very useful Timer/Counter/Prescaler Calculator, http://www.et06.dk/atmega_timers/
 *  - Arduino.cc Port Manipulation Tutorial
 *
 * This program outputs pretty accurate VGA syncronization signals. It's using a timer interrupt on timer2, so
 * hopefully you can make other cool things in loop(). */

// 640 * 480 @ 60Hz - FvSync = 60 Hz / FhSync = 31.4 kHz
// HSync : pin 9 Arduino, pin 13 VGA
// VSync : pin 8 Arduino, pin 14 VGA
//
// See this wonderful page : http://www.anatekcorp.com/driving.htm
//

/*

 VSync us (Frames)
 64    - Sync   = 1024 cycles
 1020  - Blank  = 16320 cycles
 15240 - Lines  = 243840 cycles
 350   - Blank  = 5600 cycles

       266784 = cycles total (60 times per second = 60hz)

 HSync us (Lines)
 3.77  - Sync   = 60.32 cycles
 1.89  - Blank  = 30.24 cycles
 25.17 - RGB    = 402.72 cycles
 0.94  - Blank  = 15.04 cycles

        508 Cycles per line

 Sync Start = 2 lines
 sync blank = 32 lines
 480 lines
 Sync Blank = 11 lines

 Total of 525 lines

 */

#define vga_field_line_count 525
volatile unsigned int linecount;
#define VSyncLow PORTB &= ~(1 << 0);
#define VSyncHigh PORTB |= (1 << 0);

void setup()
{
  pinMode(8, OUTPUT);                                  // VGA pins 6 & 10 grounded
  pinMode(9, OUTPUT);                                  // VSync = VGA Pin 14
                                                       // HSync = VGA Pin 13

  TCCR0A = TCCR0B = 0;                                 // Disable Arduino Timer0
  TIMSK0 = 0;                                          // Disable UART Timer

  TCCR1A =  _BV(COM1A1) | _BV(COM1A0) | _BV(WGM11);
  TCCR1B = _BV(WGM12) | _BV(WGM13) | _BV(CS10);  // No Prescaler
  ICR1 = 508;                                          // 508 Cycles per line
  TIMSK1 = _BV(TOIE1);                                 // Enable timer overflow interrupt
  OCR1A = 60;                                          // HSync Pulse

  sei(); // enable global interrupts
}

void loop(){}

ISR(TIMER1_OVF_vect)
{
  if (linecount == 0) VSyncLow;                        // VSync Pulse
  if (linecount == 2) VSyncHigh;
  if (++linecount == vga_field_line_count) linecount = 0;
}
```

NEW

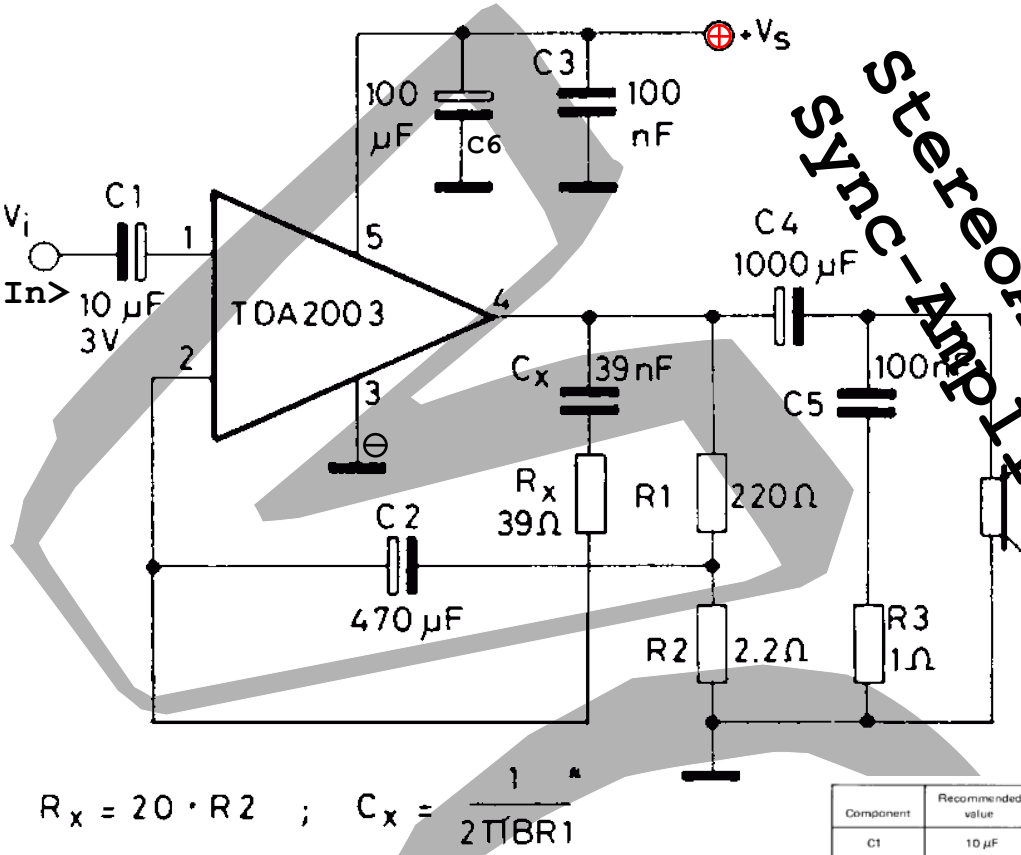
ARDUINO-Code for Sync-Generation

NEW

```
/* VGA Sync Generator - Arduino 0012
 * 11/11/08 - dwan : dwanafite@yahoo.fr
 *
 * Based on :
 *  - RG Matrix Example v.2 8/1/08, by BroHogan, from the Arduino Playground
 *  - Simplest universal VGA/PAL terminal, by Ibragimov Maxim Rafikovich, http://www.vga-avr.narod.ru/main.html
 *  - the very useful Timer/Counter/Prescaler Calculator, http://www.et06.dk/atmega_timers/
 *  - Arduino.cc Port Manipulation Tutorial
 *
 * This program outputs pretty accurate VGA syncronization signals. It's using a timer interrupt on timer2, so hopefully you
 * can make other cool things in loop(). */

// 640 * 480 @ 60Hz - FvSync = 60.3 Hz / FhSync = 31.3 kHz
// HSync : pin 7 Arduino, pin 13 VGA
// VSync : pin 6 Arduino, pin 14 VGA
// Arduino's pin 4 is HIGH when video can be sent, LOW otherwise. I use it to power a transistor.
// See this wonderful page : http://www.anatekcorp.com/driving.htm
// 1 NOP = 62,5 ns wasted

#define NOP asm("NOP")

#define vga_field_line_count 525         // number of VGA lines (default 525)
                                         // 625 lines produces 50Hz with default 0x3F ################################
#define ISR_FREQ 0x3f                    // Sets the speed of the ISR - LOWER IS FASTER - 62 (default 0x3F)

volatile unsigned int linecount;

void setup()
{
  Serial.begin(9600);           // set up Serial library at 9600 bps - serial has 2 b active (?) ##############################
  DDRD |= B11010000;            // it sets pins 7, 6 and 4 as output without changing the value of pins 0 & 1, which are RX & TX
  //       76543210 <- pin translation                < ---- now pin 4 ##############################

  PORTD |= B11000000;   // sets pins 7 (hSync) and 6 (vSync) HIGH
  //       76543210;

  setISRtimer();                        // setup the timer
  startISR();                           // start the timer to toggle shutdown
}

void loop() ////////////////////////////////////////////////////////////////
{

}

////////////////////////////// ISR Timer Functions ////////////////////////////
ISR(TIMER2_COMPA_vect)
{
        // Stop video - pin 4 LOW
        PORTD &= ~(1 << 4);                     // < ---- now pin 4 instead of 5 ##############################

        // Front porch
        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;   // 562,5 ns


        // Count number of lines
        if (++linecount == vga_field_line_count)
        {
        linecount = 0;
        }

        // can it be vsync tiem nao ?
        if ((linecount == 10 )||(linecount == 11 ))
        {
                // hsync LOW
                // vsync LOW
                PORTD &= ~(1 << 7);
                PORTD &= ~(1 << 6);
        }
        else    // ,hsync only
        {
                // hsync LOW
                // vsync HIGH
                PORTD &= ~(1 << 7);
                PORTD |= (1 << 6);
        }

        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;   // 2500 ns
        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;
        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;
        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;

        // nonetheless,
        // hsync HIGH
        PORTD |= (1 << 7);

        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;   // 1937,5 ns
        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;
        NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP; NOP;
        NOP;

        // Start video if we are in a visible area ////////////////////////////////////////////////
        if (linecount > 45)
        {
        // pin 4 HIGH
        PORTD |= (1 << 4);                      // < ---- now pin 4 instead of 5 ##############################
        }
}

void setISRtimer(){  // setup ISR timer controling toggleing
  TCCR2A = 0x02;                        // WGM22=0 + WGM21=1 + WGM20=0 = Mode2 (CTC)
  TCCR2B = (1 << CS01);                 // /8 prescaler (2MHz)
  TCNT2 = 0;                            // clear counter
  OCR2A = ISR_FREQ;                     // set TOP (divisor) - see #define
}

void startISR(){  // Starts the ISR
  TCNT2 = 0;                            // clear counter (needed here also)
  TIMSK2|=(1<<OCIE2A);                  // set interrupts=enabled (calls ISR(TIMER2_COMPA_vect)
}

void stopISR(){    // Stops the ISR
  TIMSK2&=~(1<<OCIE2A);                  // disable interrupts
}
```

ARDUINO-Code for Sync-Generation

StereoAmp 12V Sync-Amplification



R1 - 220R
R2 - 2R2
R3 - 1R0
Rx - 39R

C1 - 4u7, 3V
C2 - 470u, 3V
C3 - 1000u, 10V
C4,5 - 100n
C6 - 100u, 16V
Cx - 39n

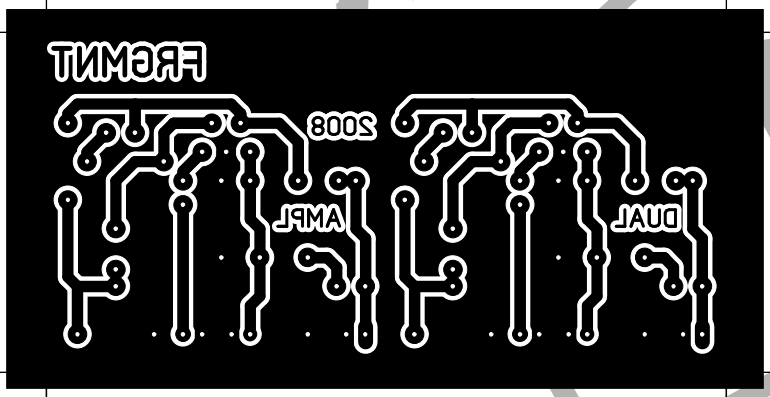
IC1 - TDA2003/2002

$$R_x = 20 \cdot R2 \quad ; \quad C_x = \frac{1}{2\pi BR1}$$

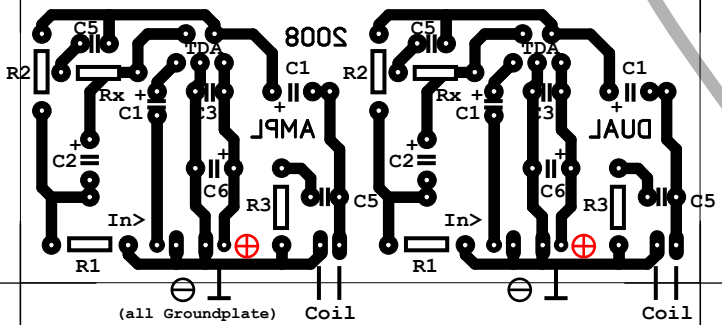
Component	Recommended value	Purpose	Larger than recommended value	Smaller than recommended value
C1	10 µF	Input DC decoupling		Noise at switch-on, switch-off
C2	470 µF	Ripple rejection		Degradation of PSRR
C3	0.1 µF	Supply bypassing		Danger of oscillation
C4	1000 µF	Output coupling to load		Higher low frequency cutoff
C5	0.1 µF	Frequency stability		Danger of oscillation at high frequencies with inductive loads
CF_B	$\simeq \frac{1}{2\pi BR1}$	Upper frequency cutoff	Lower bandwidth	Larger bandwidth
R1	$(A_V - 1) \cdot R2$	Closed loop gain determination		Increase of drain current
R2	2.2 Ω	Closed loop gain and PSRR determination	Degradation of PSRR	
R3	1 Ω	Frequency stability	Danger of oscillation at high frequencies with inductive loads	
R_{FB}	≅ 20 R2	Upper frequency cutoff	Poor high frequency attenuation	Danger of oscillation

TABLE 1

90*90 mm PCB effective, 100*100 mm built



Etch Pattern 1:1

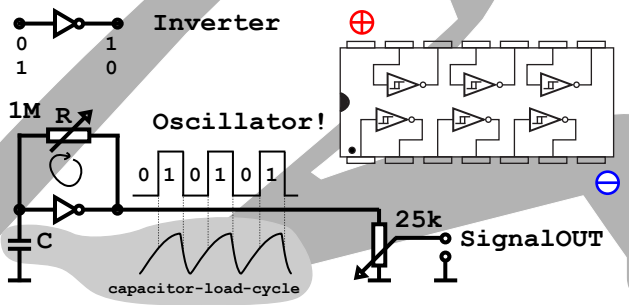


P1 RED V●●
P2 GRN V●●
P3 BLU V●●
P5 GND HSy
P6 RED_RTN
P7 GRN_RTN
P8 BLU_RTN
P10 GND VSy
P11 MonID
P13 HSync●●
P14 VSync●●

VGA Exp. "Horror Vacui"
Paper for xxxxx Workshop
Jo frgmnt Grys / tob.de.vu / 2009
Transmitting Object Behaviors

The Experiments!

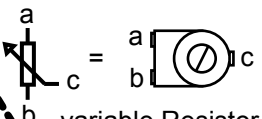
H/V-Sync from Computer - R,G,B Injection (Signalgenerator Software or Hardware)
http://www.abacom-online.de/uk/html/audiowave.html
http://linux.maruhn.com/sec/siggen.html



Hex CMOS Inverters' **Power Supply**:
74AC14 2..6V, 7V max
74HC14 2..6V, 7V max
40106 3..18V

Frequency f = 1 / 0.8 * R * C in Hz

Expl. R=1M, C=100nF
f=1 / (0.8 * 1000000 * 0.0000001)
= 12.5 Hz
= 12.5 pulses per sec, when pot on max



try VSync -> RGB & Arduino Pin 4 -> RGB..

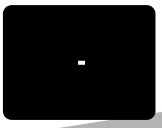
H/V-Sync from Arduino - R,G,B Injection (Signalgenerator Software or Hardware)
see AVR code!

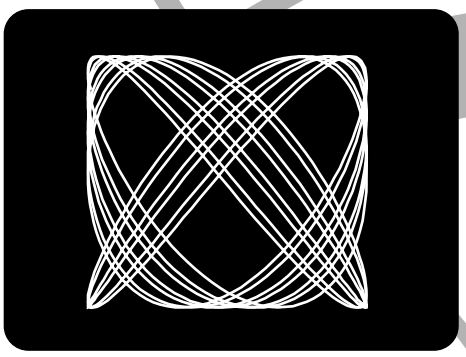
Inside THE BEAST: Dont kill yourself - just dont touch HIGH VOLTAGE i.e. the Anode!

V-Coil connected - H-Coil driven by AMP; Signal Injection
this coil needs higher driving power then the H-Coil

H-Coil connected - V-Coil driven by AMP; Signal Injection

V-Coil & H-Coil driven by AMP; Signal Injection





ANNOTATIONS:
 # Horizontal coil (H-Coil) on old monitors is used for internal high-voltage generation - so it cannot be simply unplugged but needs to be substituted with external coil.
 # Computer senses monitor on Pin 02 (connected to GND via 75ohms)
 # Monitor senses computer on Pin 05 (connected to GND]