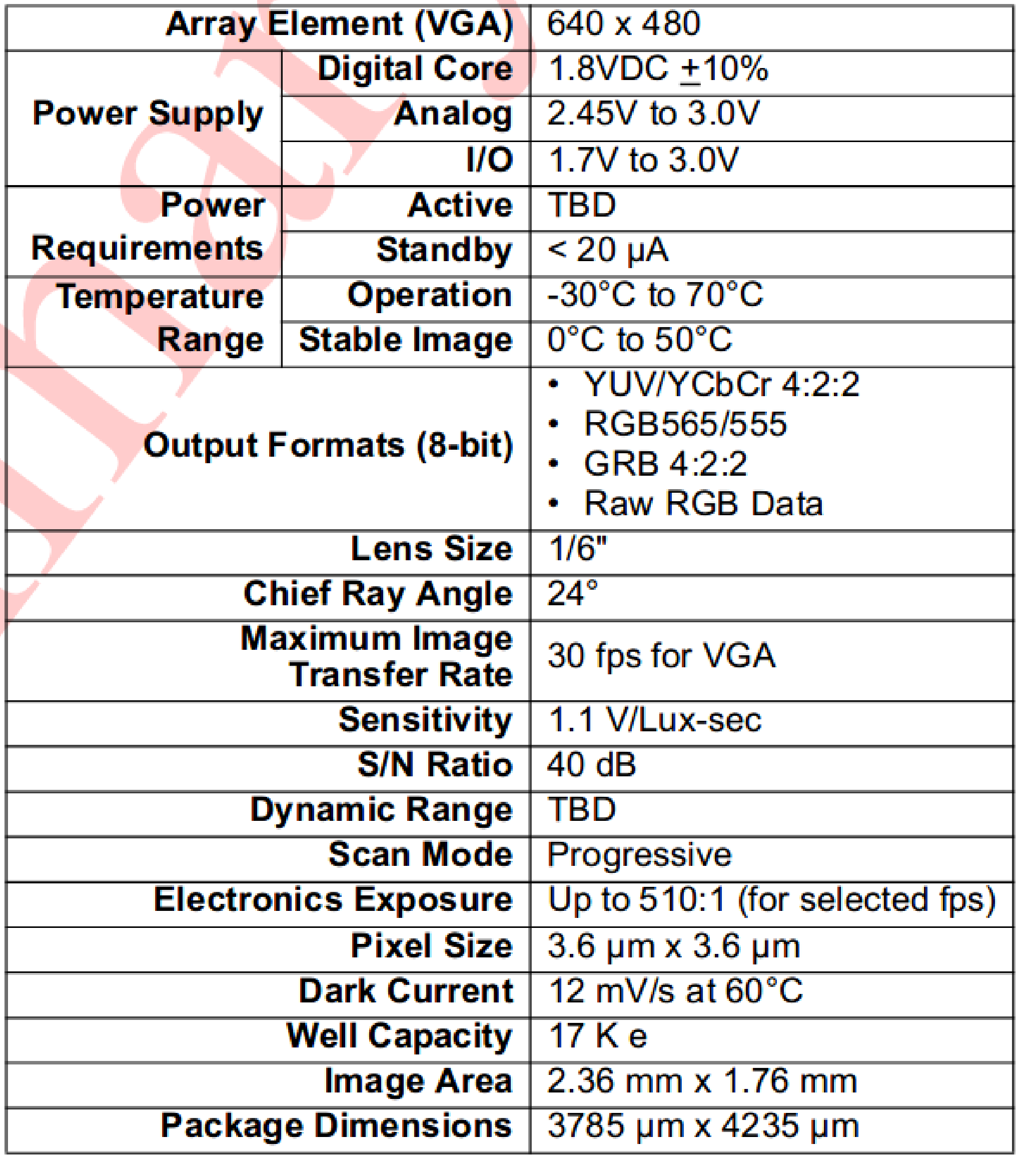
Práctica 6

Sensor de video y FPGA

Sensor de video

- OV7670

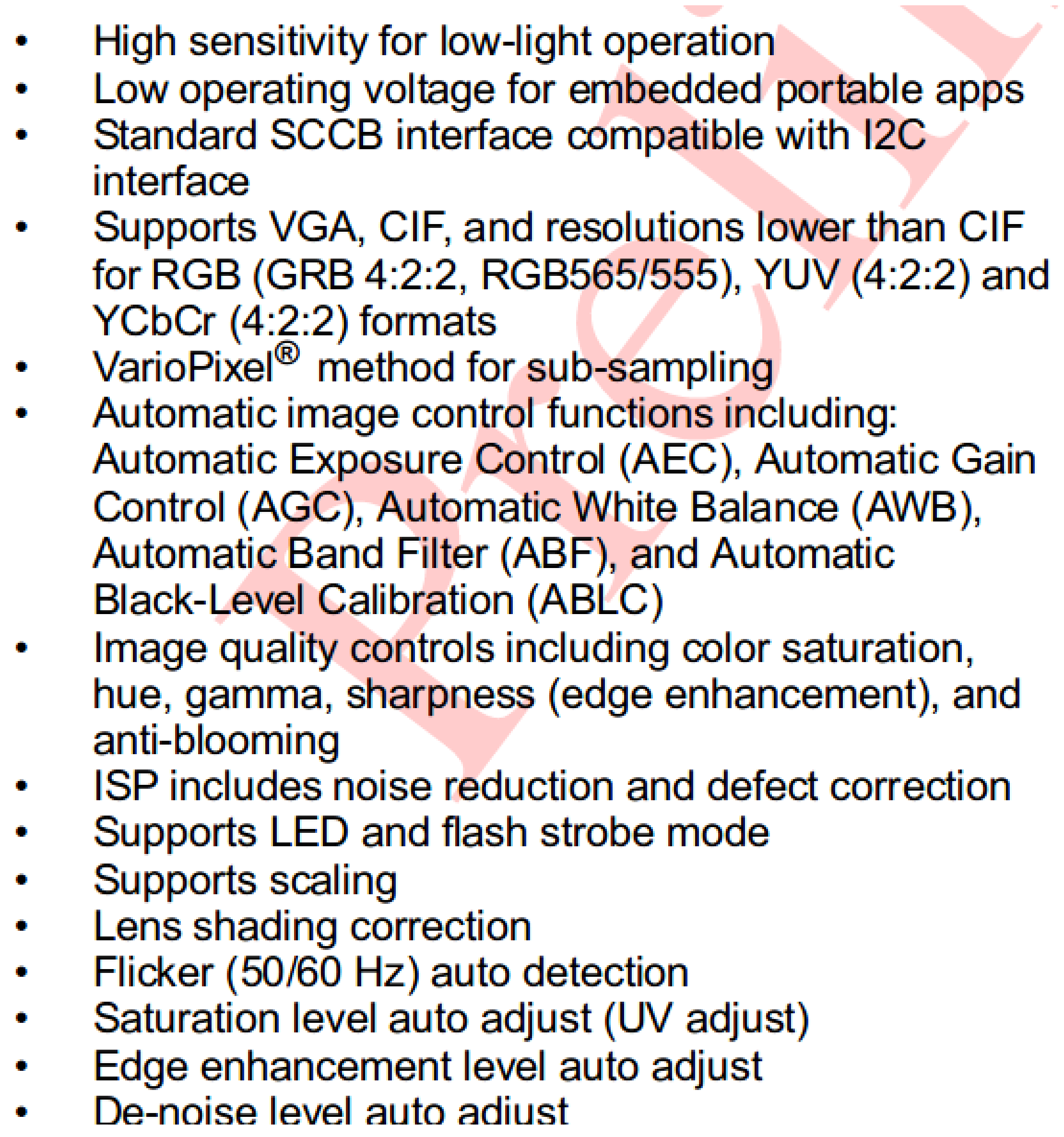


Array Element (VGA)		640 x 480
Power Supply	Digital Core	1.8VDC ±10%
	Analog	2.45V to 3.0V
	I/O	1.7V to 3.0V
Power Requirements	Active	TBD
	Standby	< 20 µA
Temperature Range	Operation	-30°C to 70°C
	Stable Image	0°C to 50°C
Output Formats (8-bit)		• YUV/YCbCr 4:2:2 • RGB565/555 • GRB 4:2:2 • Raw RGB Data
Lens Size		1/6"
Chief Ray Angle		24°
Maximum Image Transfer Rate		30 fps for VGA
Sensitivity		1.1 V/Lux-sec
S/N Ratio		40 dB
Dynamic Range		TBD
Scan Mode		Progressive
Electronics Exposure		Up to 510:1 (for selected fps)
Pixel Size		3.6 µm x 3.6 µm
Dark Current		12 mV/s at 60°C
Well Capacity		17 K e
Image Area		2.36 mm x 1.76 mm
Package Dimensions		3785 µm x 4235 µm

Tomado Datasheet

Sensor de video

- OV7670

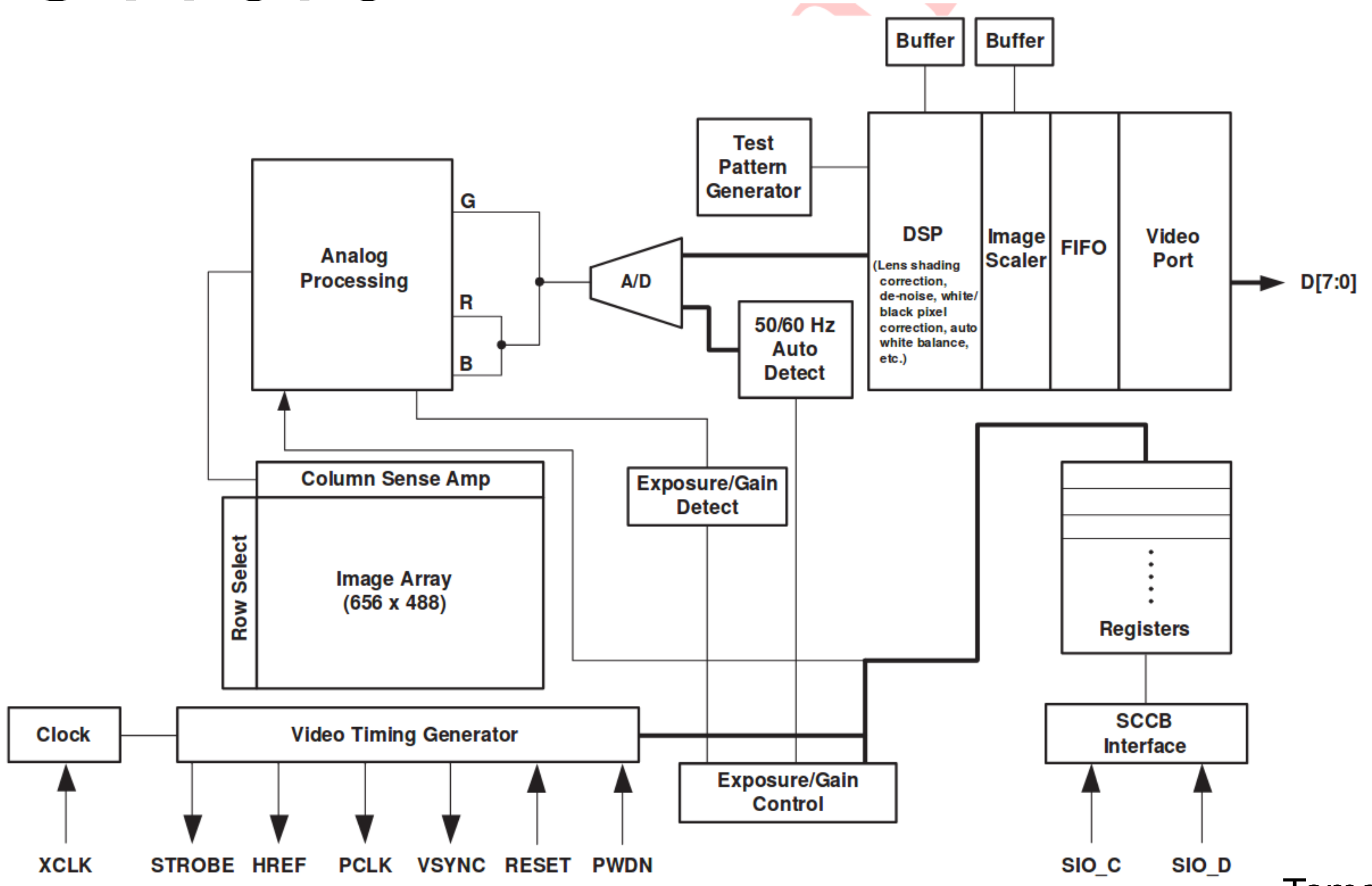


- High sensitivity for low-light operation
- Low operating voltage for embedded portable apps
- Standard SCCB interface compatible with I2C interface
- Supports VGA, CIF, and resolutions lower than CIF for RGB (GRB 4:2:2, RGB565/555), YUV (4:2:2) and YCbCr (4:2:2) formats
- VarioPixel® method for sub-sampling
- Automatic image control functions including: Automatic Exposure Control (AEC), Automatic Gain Control (AGC), Automatic White Balance (AWB), Automatic Band Filter (ABF), and Automatic Black-Level Calibration (ABLC)
- Image quality controls including color saturation, hue, gamma, sharpness (edge enhancement), and anti-blooming
- ISP includes noise reduction and defect correction
- Supports LED and flash strobe mode
- Supports scaling
- Lens shading correction
- Flicker (50/60 Hz) auto detection
- Saturation level auto adjust (UV adjust)
- Edge enhancement level auto adjust
- De-noise level auto adjust

Tomado Datasheet

Sensor de video

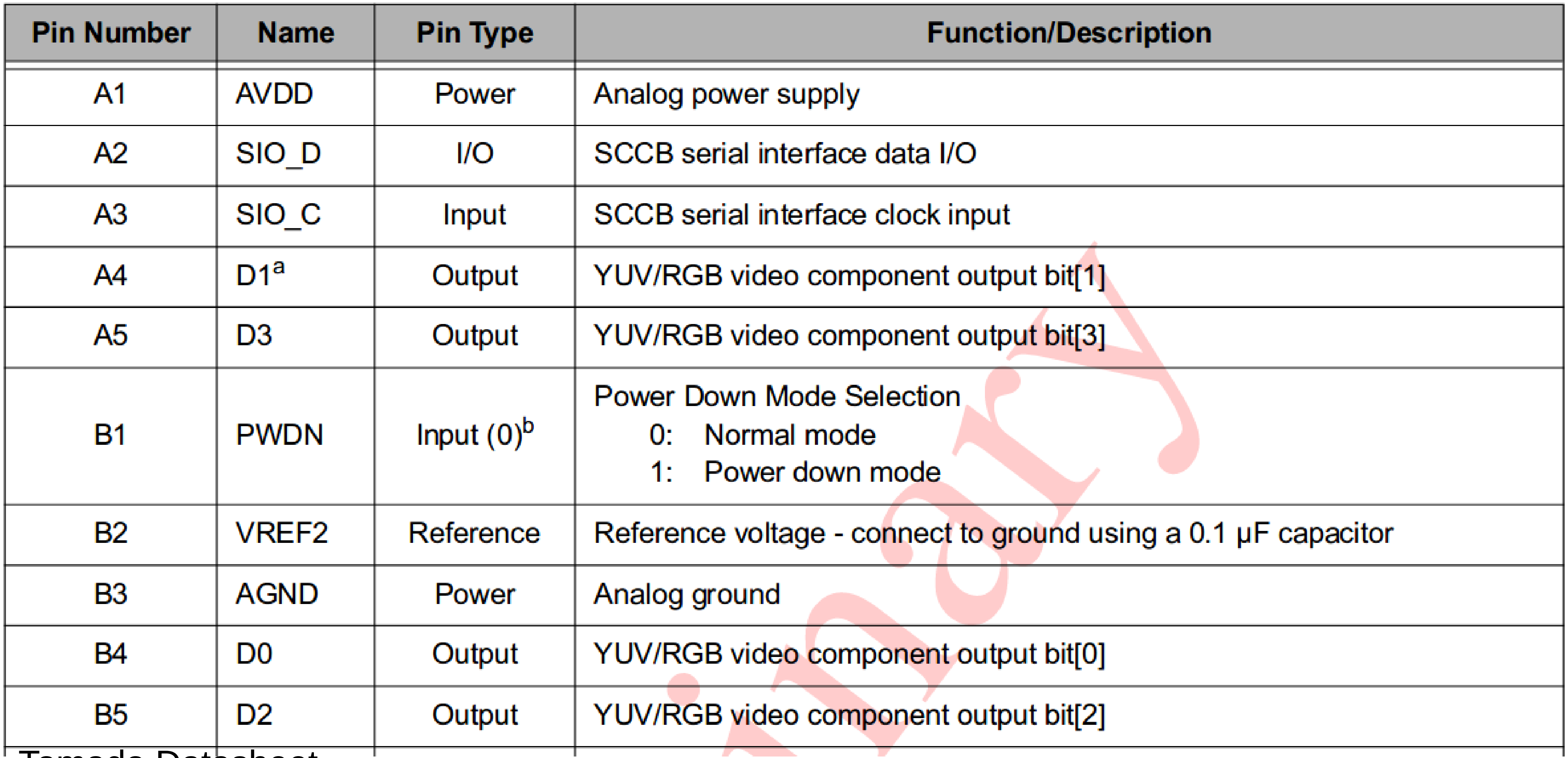
- OV7670



Tomado Datasheet

Sensor de video

- OV7670

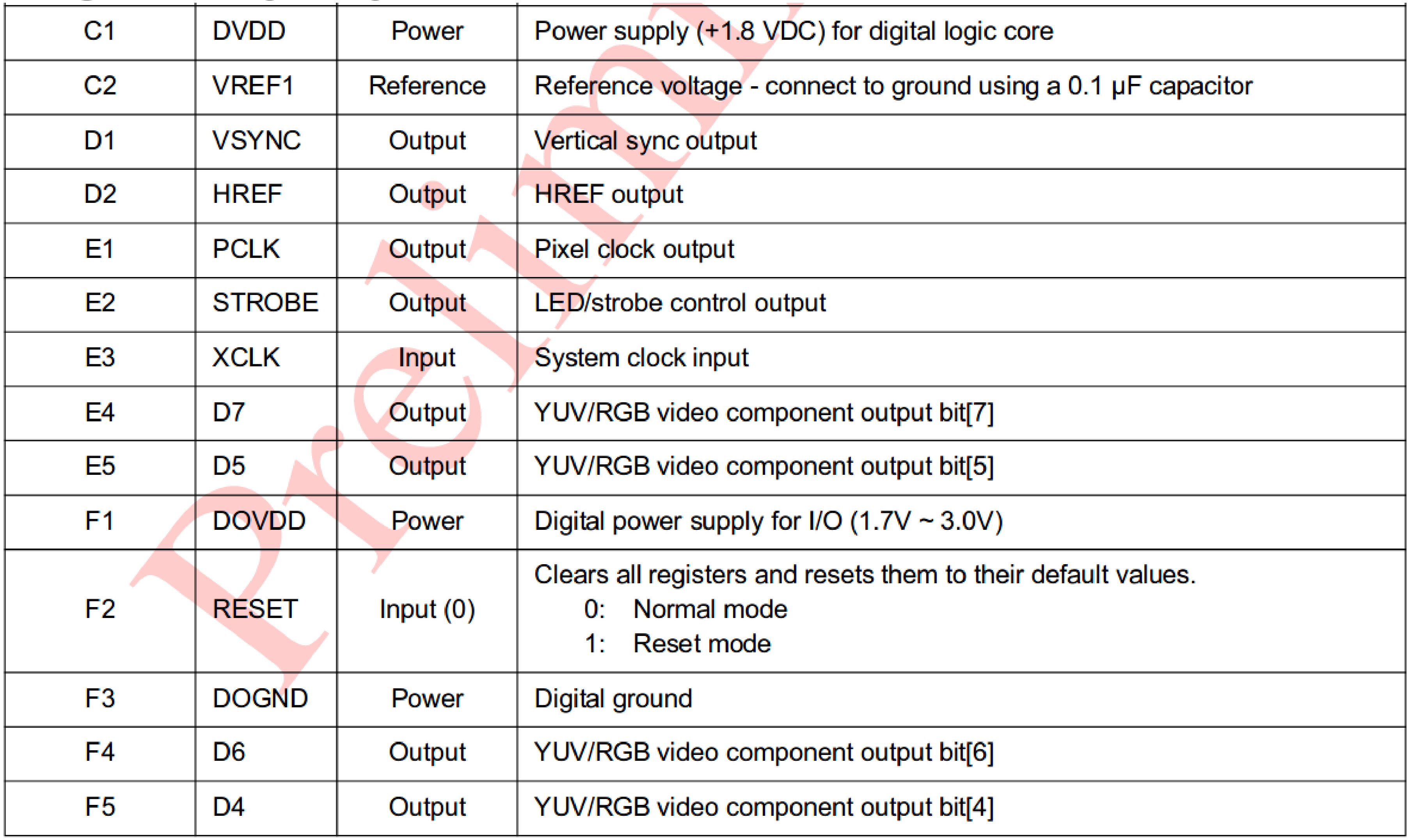


Pin Number	Name	Pin Type	Function/Description
A1	AVDD	Power	Analog power supply
A2	SIO_D	I/O	SCCB serial interface data I/O
A3	SIO_C	Input	SCCB serial interface clock input
A4	D1[a]	Output	YUV/RGB video component output bit[1]
A5	D3	Output	YUV/RGB video component output bit[3]
B1	PWDN	Input (0)[b]	Power Down Mode Selection 0: Normal mode 1: Power down mode
B2	VREF2	Reference	Reference voltage - connect to ground using a 0.1 µF capacitor
B3	AGND	Power	Analog ground
B4	D0	Output	YUV/RGB video component output bit[0]
B5	D2	Output	YUV/RGB video component output bit[2]

Tomado Datasheet

Sensor de video

- OV7670

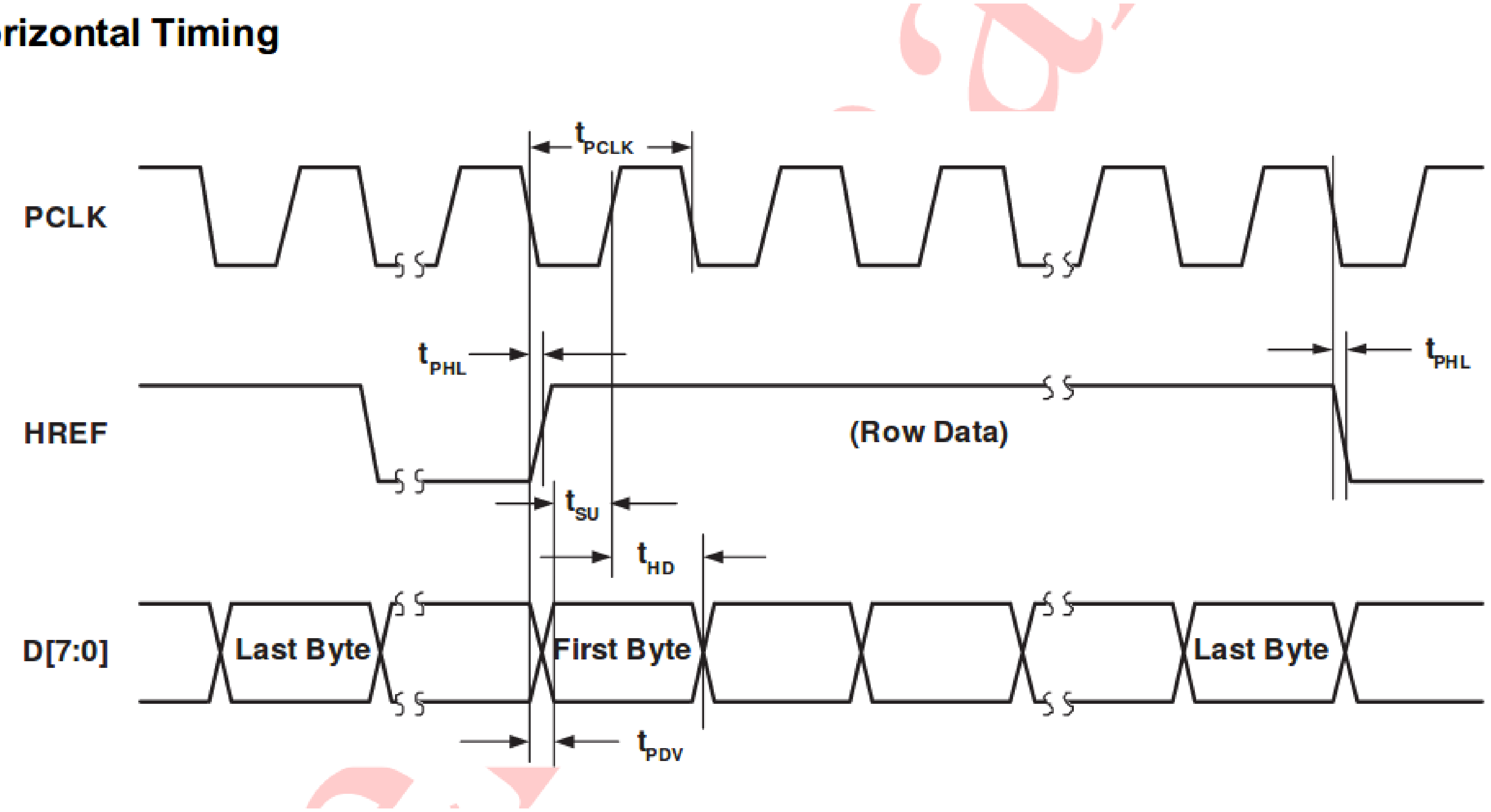


C1	DVDD	Power	Power supply (+1.8 VDC) for digital logic core
C2	VREF1	Reference	Reference voltage - connect to ground using a 0.1 μF capacitor
D1	VSYNC	Output	Vertical sync output
D2	HREF	Output	HREF output
E1	PCLK	Output	Pixel clock output
E2	STROBE	Output	LED/strobe control output
E3	XCLK	Input	System clock input
E4	D7	Output	YUV/RGB video component output bit[7]
E5	D5	Output	YUV/RGB video component output bit[5]
F1	DOVDD	Power	Digital power supply for I/O (1.7V ~ 3.0V)
F2	RESET	Input (0)	Clears all registers and resets them to their default values. 0: Normal mode 1: Reset mode
F3	DOGND	Power	Digital ground
F4	D6	Output	YUV/RGB video component output bit[6]
F5	D4	Output	YUV/RGB video component output bit[4]

Sensor de video

- OV7670

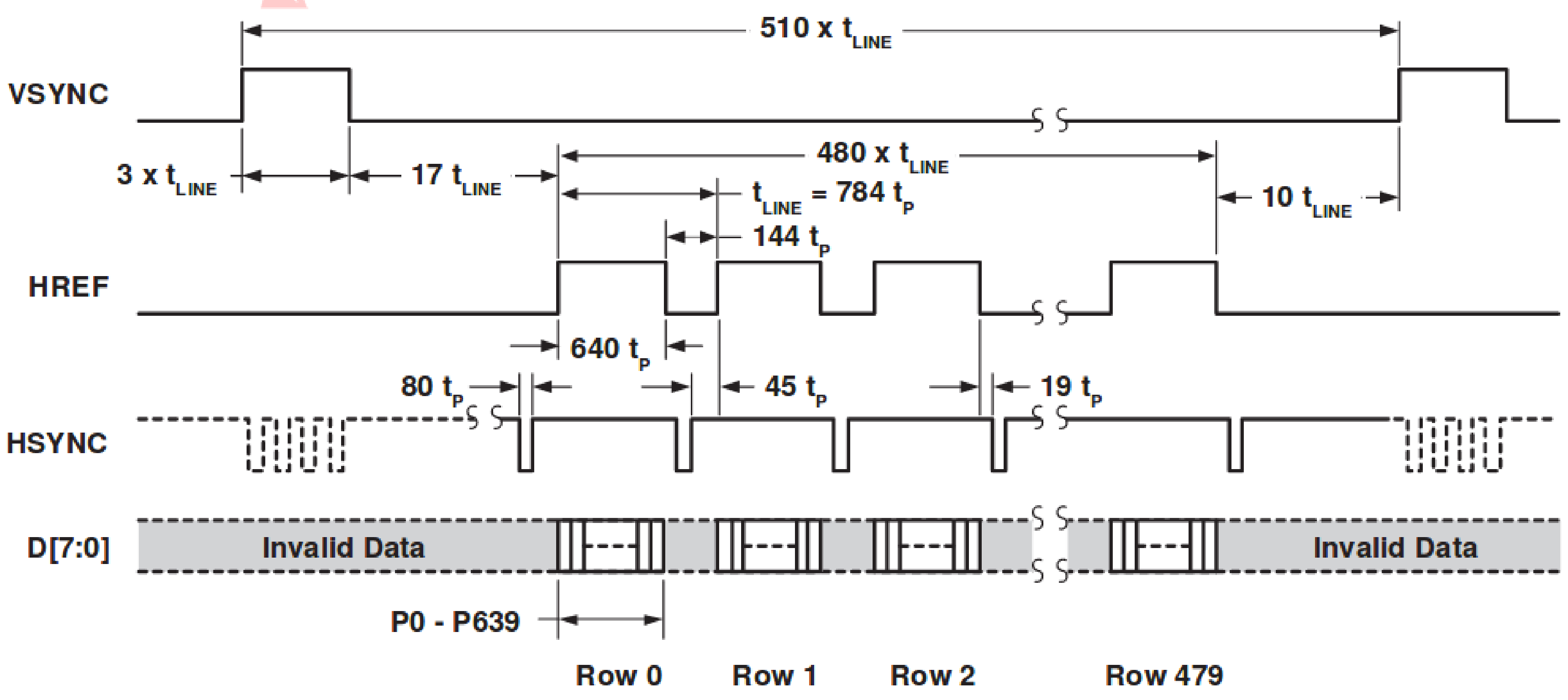
Figure 5 Horizontal Timing



Sensor de video

- OV7670

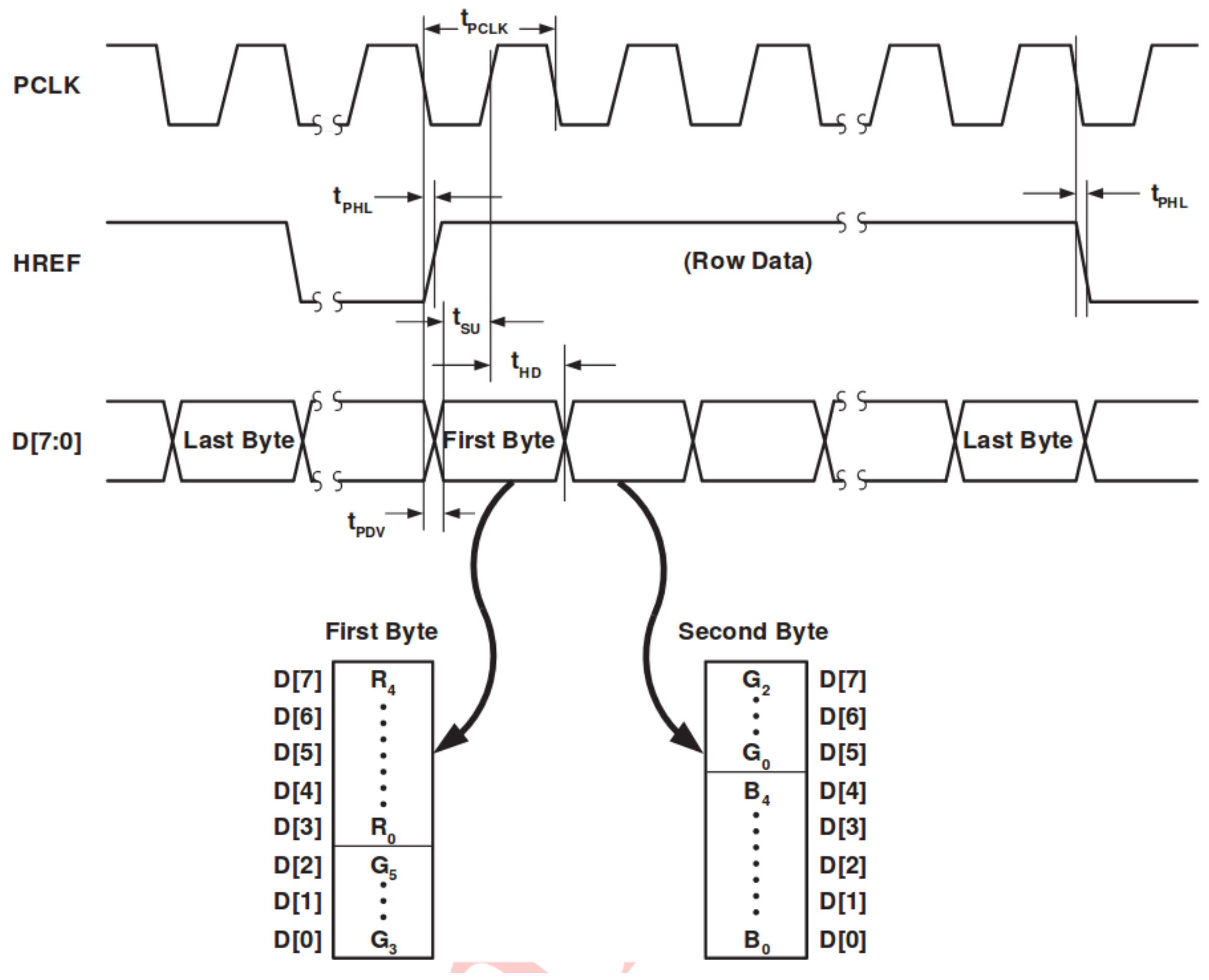
Figure 6 VGA Frame Timing



Sensor de video

- OV7670

Figure 11 RGB 565 Output Timing Diagram



Sensor de video

- OV7670

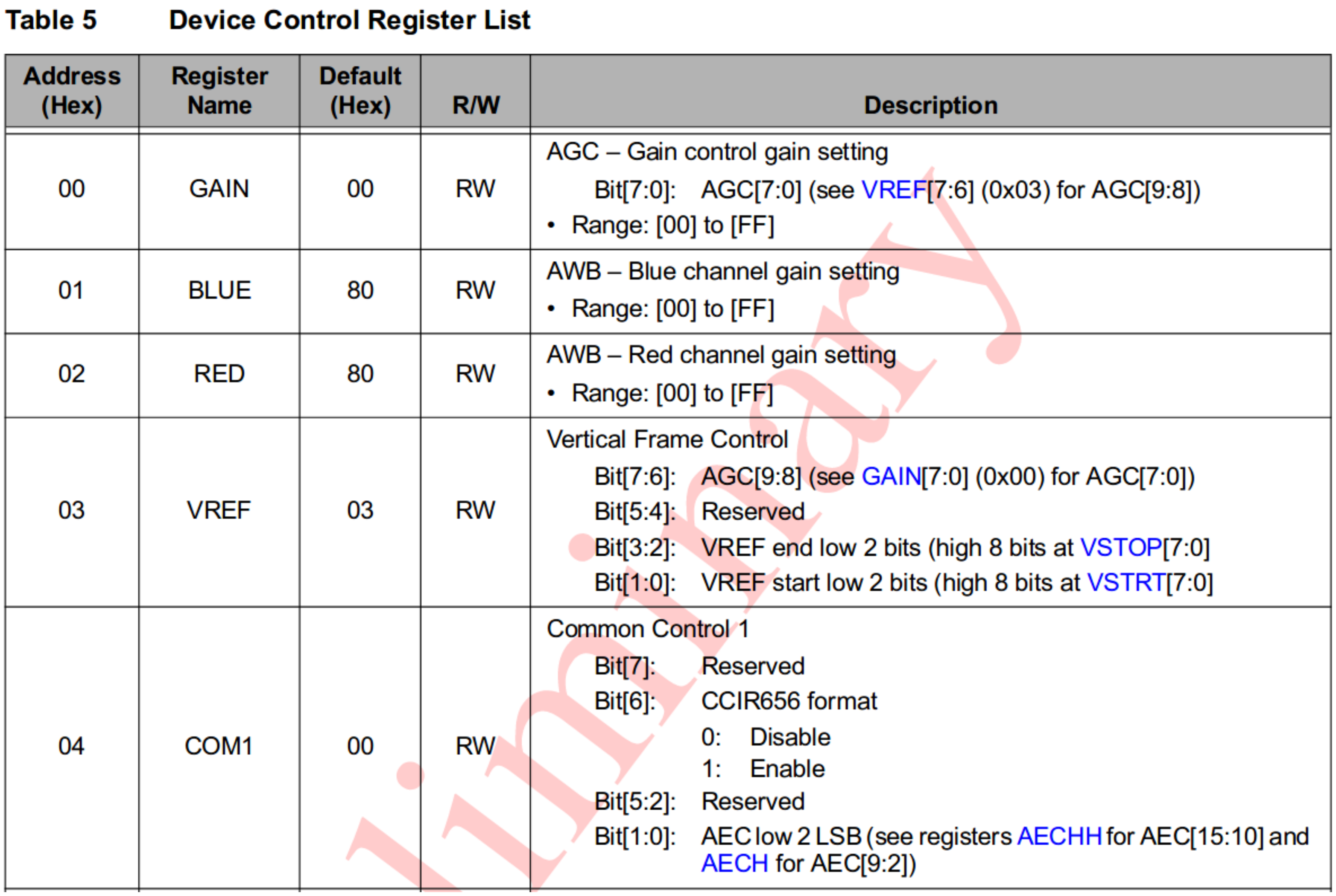
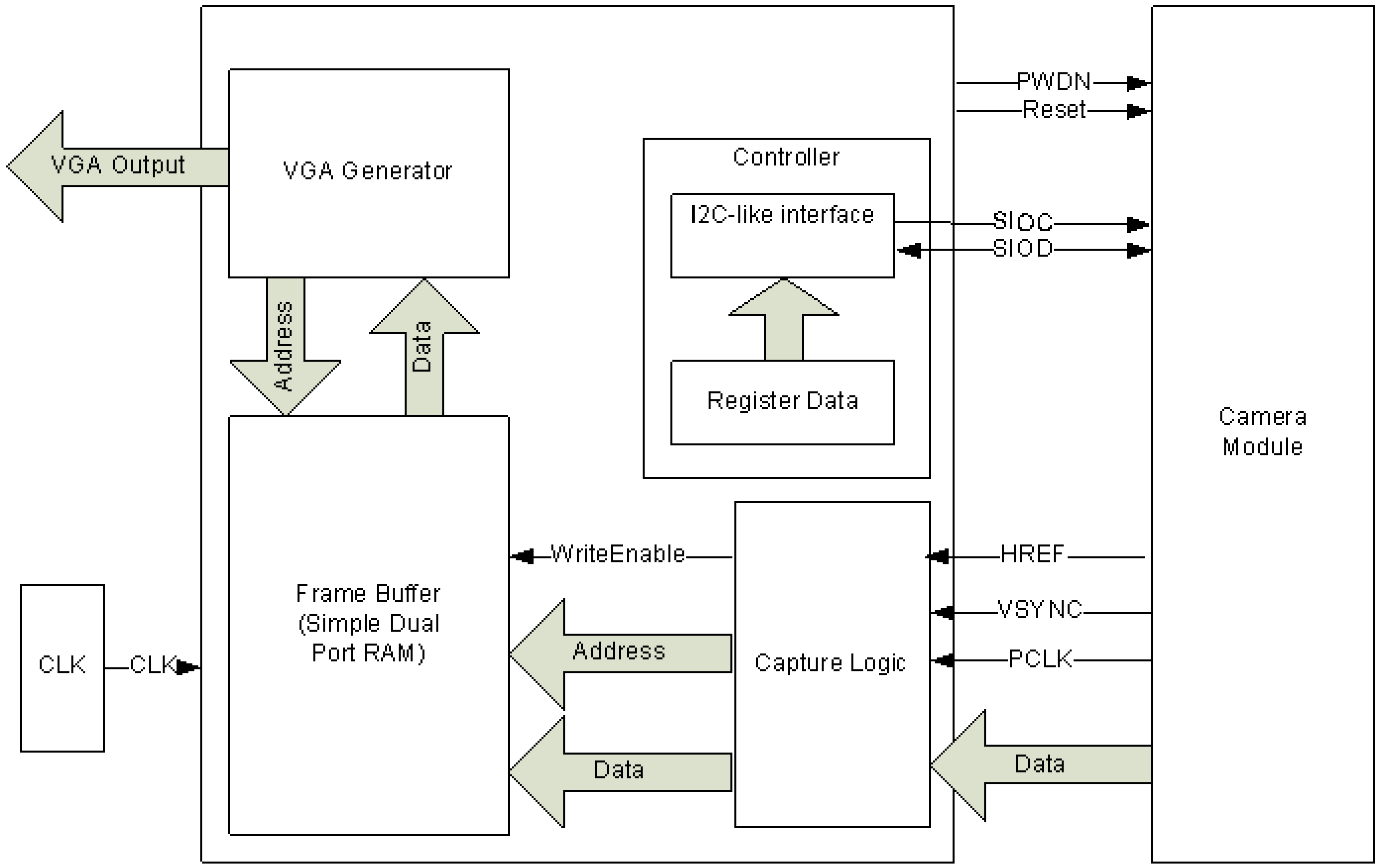


Table 5 Device Control Register List

Address (Hex)	Register Name	Default (Hex)	R/W	Description
00	GAIN	00	RW	AGC – Gain control gain setting Bit[7:0]: AGC[7:0] (see VREF[7:6] (0x03) for AGC[9:8]) • Range: [00] to [FF]
01	BLUE	80	RW	AWB – Blue channel gain setting • Range: [00] to [FF]
02	RED	80	RW	AWB – Red channel gain setting • Range: [00] to [FF]
03	VREF	03	RW	Vertical Frame Control Bit[7:6]: AGC[9:8] (see GAIN[7:0] (0x00) for AGC[7:0]) Bit[5:4]: Reserved Bit[3:2]: VREF end low 2 bits (high 8 bits at VSTOP[7:0] Bit[1:0]: VREF start low 2 bits (high 8 bits at VSTRT[7:0]
04	COM1	00	RW	Common Control 1 Bit[7]: Reserved Bit[6]: CCIR656 format 0: Disable 1: Enable Bit[5:2]: Reserved Bit[1:0]: AEC low 2 LSB (see registers AECHH for AEC[15:10] and AECH for AEC[9:2])

Sensor de video

- Control de OV7670

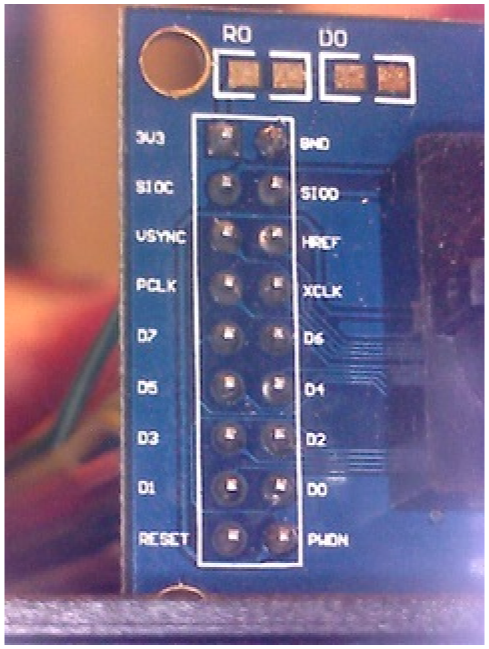


Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- Pines del sensor

The camera's interface



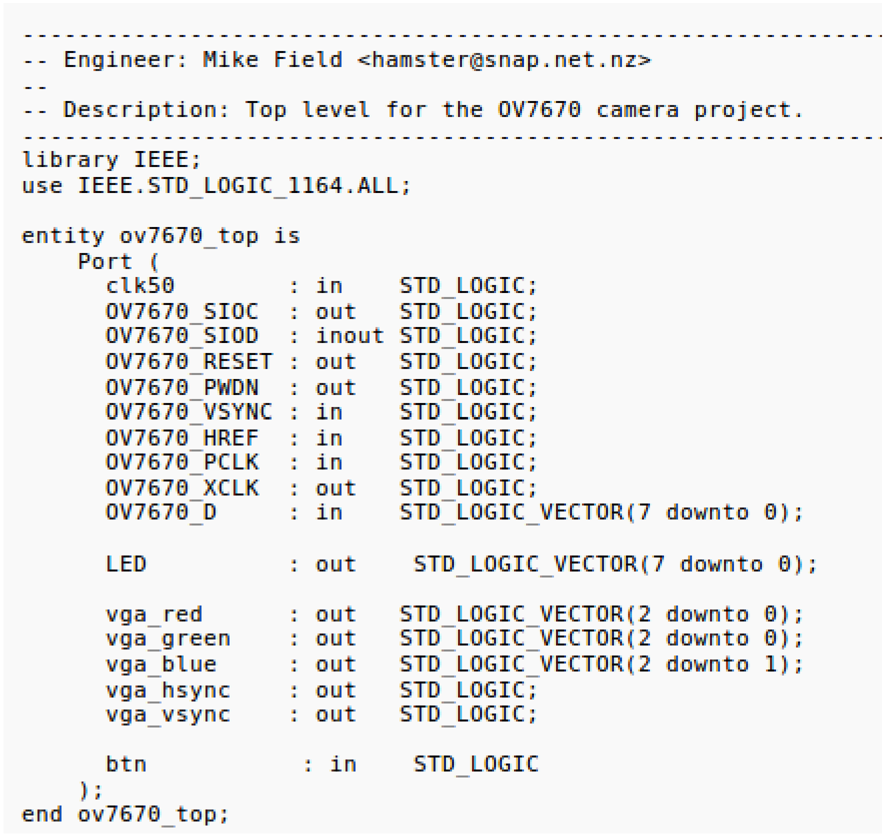
Full specs are in the datasheet and in the implmentation guide

Signal	Usage	Active
3V3	3.3V power	
Gnd	Ground	
SIOC	Serial command bus clock (up to 400KHz)	
SIOD	Serial command bus data	
VSYNC	Vertical Sync	Active High, configurable
HREF	CE output for pixel sampling	Active High, configurable
PCLK	Pixel Clock	
XCLK	System clock (10-48MHz, 24MHz Typ)	
D0-D7	Pixel data	
RESET	Device Reset	Active Low
PWDN	Device Power Down	Active High

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- ov7670_top.vhd

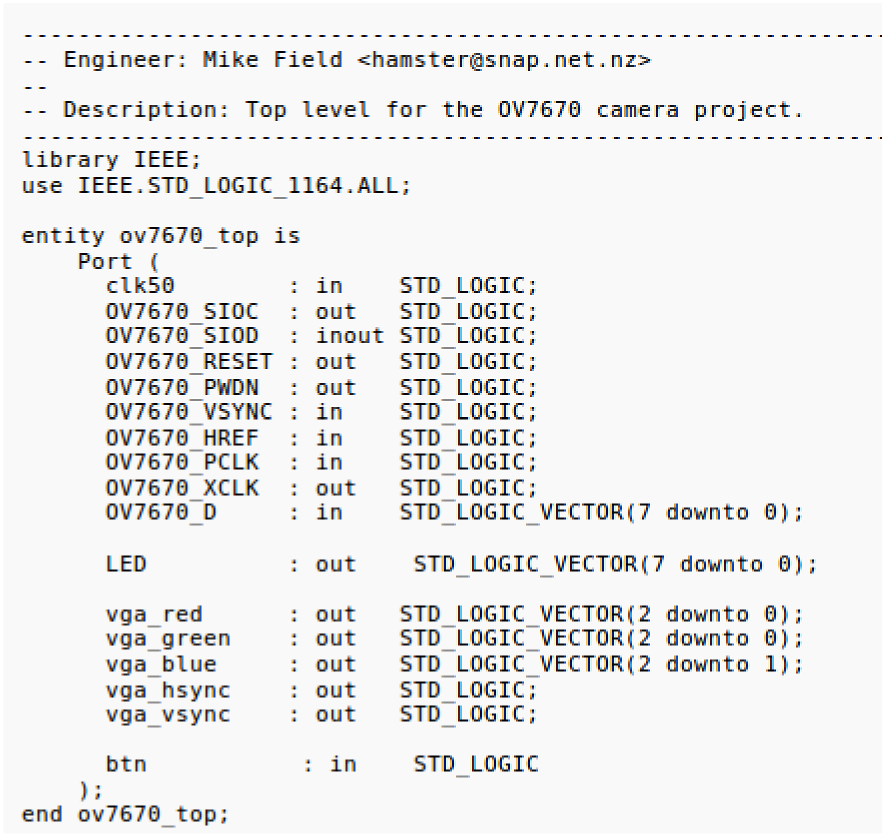


```
-------------------------------------------------------------------
-- Engineer: Mike Field <hamster@snap.net.nz>
--
-- Description: Top level for the OV7670 camera project.
-------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ov7670_top is
    Port (
      clk50        : in    STD_LOGIC;
      OV7670_SIOC  : out   STD_LOGIC;
      OV7670_SIOD  : inout STD_LOGIC;
      OV7670_RESET : out   STD_LOGIC;
      OV7670_PWDN  : out   STD_LOGIC;
      OV7670_VSYNC : in    STD_LOGIC;
      OV7670_HREF  : in    STD_LOGIC;
      OV7670_PCLK  : in    STD_LOGIC;
      OV7670_XCLK  : out   STD_LOGIC;
      OV7670_D     : in    STD_LOGIC_VECTOR(7 downto 0);

      LED          : out    STD_LOGIC_VECTOR(7 downto 0);

      vga_red      : out   STD_LOGIC_VECTOR(2 downto 0);
      vga_green    : out   STD_LOGIC_VECTOR(2 downto 0);
      vga_blue     : out   STD_LOGIC_VECTOR(2 downto 1);
      vga_hsync    : out   STD_LOGIC;
      vga_vsync    : out   STD_LOGIC;

      btn           : in    STD_LOGIC
    );
end ov7670_top;
```

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- ov7670_top.vhd

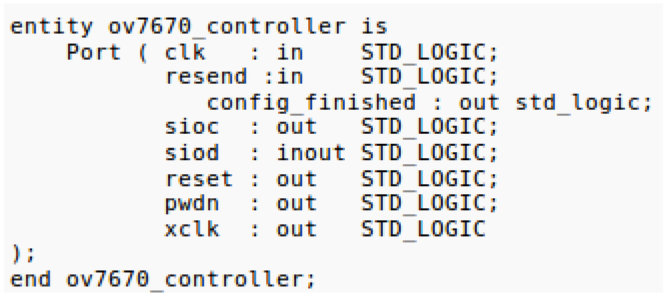


```
--------------------------------------------------------------
-- Engineer: Mike Field <hamster@snap.net.nz>
--
-- Description: Top level for the OV7670 camera project.
--------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ov7670_top is
    Port (
      clk50         : in    STD_LOGIC;
      OV7670_SIOC   : out   STD_LOGIC;
      OV7670_SIOD   : inout STD_LOGIC;
      OV7670_RESET  : out   STD_LOGIC;
      OV7670_PWDN   : out   STD_LOGIC;
      OV7670_VSYNC  : in    STD_LOGIC;
      OV7670_HREF   : in    STD_LOGIC;
      OV7670_PCLK   : in    STD_LOGIC;
      OV7670_XCLK   : out   STD_LOGIC;
      OV7670_D      : in    STD_LOGIC_VECTOR(7 downto 0);

      LED           : out    STD_LOGIC_VECTOR(7 downto 0);

      vga_red       : out   STD_LOGIC_VECTOR(2 downto 0);
      vga_green     : out   STD_LOGIC_VECTOR(2 downto 0);
      vga_blue      : out   STD_LOGIC_VECTOR(2 downto 1);
      vga_hsync     : out   STD_LOGIC;
      vga_vsync     : out   STD_LOGIC;

      btn            : in    STD_LOGIC
    );
end ov7670_top;
```

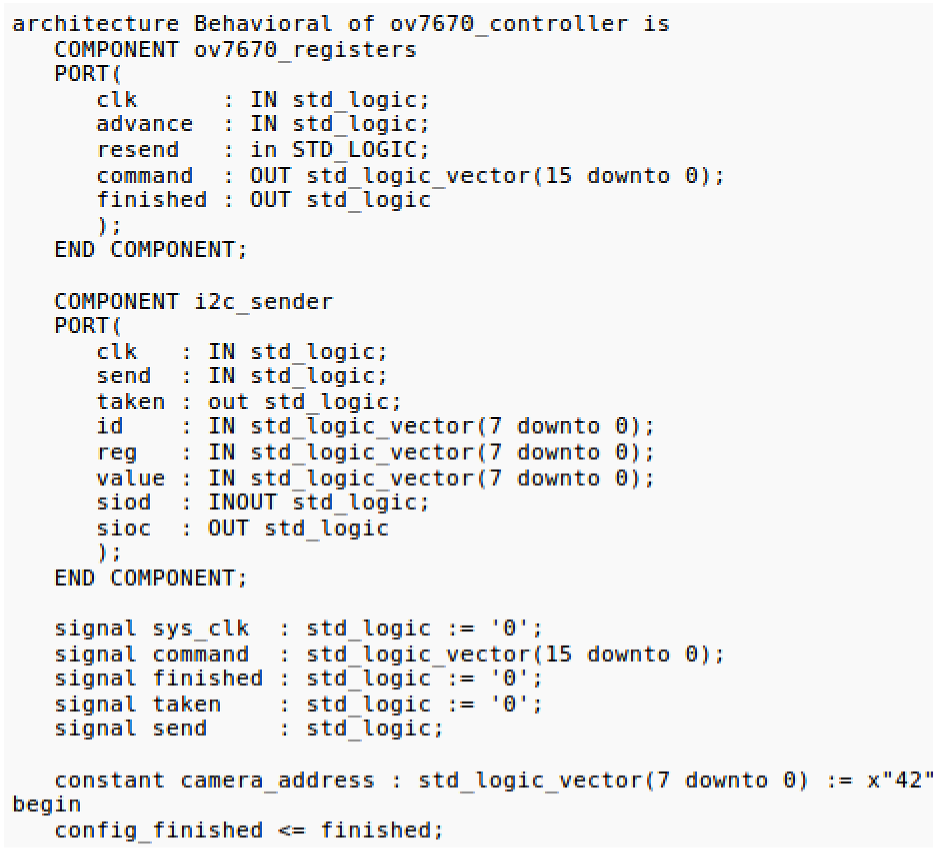
Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- ov7670_controller.vhd



```
entity ov7670_controller is
    Port ( clk   : in    STD_LOGIC;
           resend :in    STD_LOGIC;
              config_finished : out std_logic;
           sioc  : out   STD_LOGIC;
           siod  : inout STD_LOGIC;
           reset : out   STD_LOGIC;
           pwdn  : out   STD_LOGIC;
           xclk  : out   STD_LOGIC
);
end ov7670_controller;
```

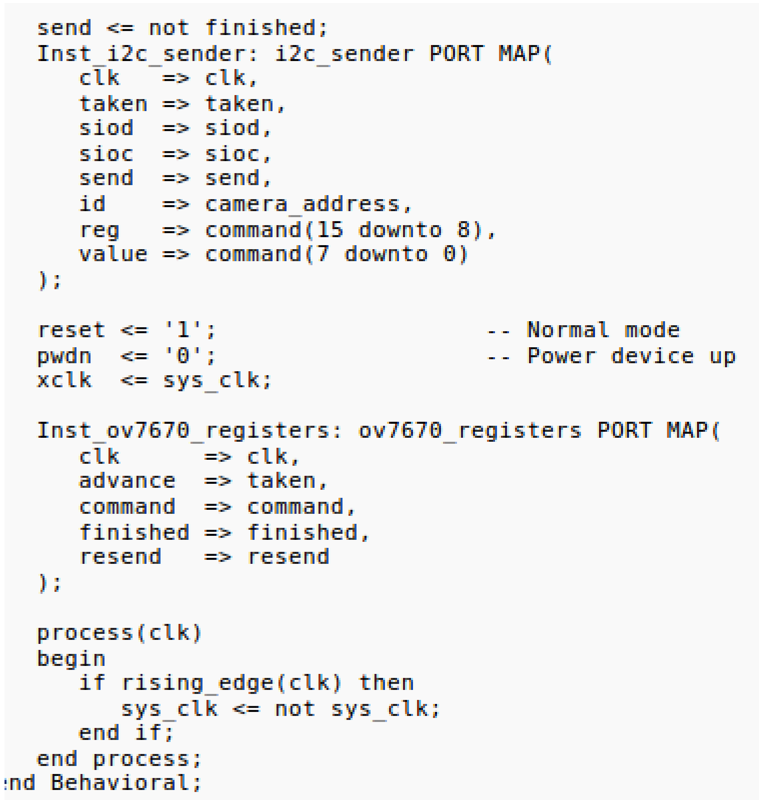


```
architecture Behavioral of ov7670_controller is
   COMPONENT ov7670_registers
   PORT(
      clk      : IN std_logic;
      advance  : IN std_logic;
      resend   : in STD_LOGIC;
      command  : OUT std_logic_vector(15 downto 0);
      finished : OUT std_logic
      );
   END COMPONENT;

   COMPONENT i2c_sender
   PORT(
      clk   : IN std_logic;
      send  : IN std_logic;
      taken : out std_logic;
      id    : IN std_logic_vector(7 downto 0);
      reg   : IN std_logic_vector(7 downto 0);
      value : IN std_logic_vector(7 downto 0);
      siod  : INOUT std_logic;
      sioc  : OUT std_logic
      );
   END COMPONENT;

   signal sys_clk  : std_logic := '0';
   signal command  : std_logic_vector(15 downto 0);
   signal finished : std_logic := '0';
   signal taken    : std_logic := '0';
   signal send     : std_logic;

   constant camera_address : std_logic_vector(7 downto 0) := x"42"
begin
   config_finished <= finished;
```

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- ov7670_controller.vhd

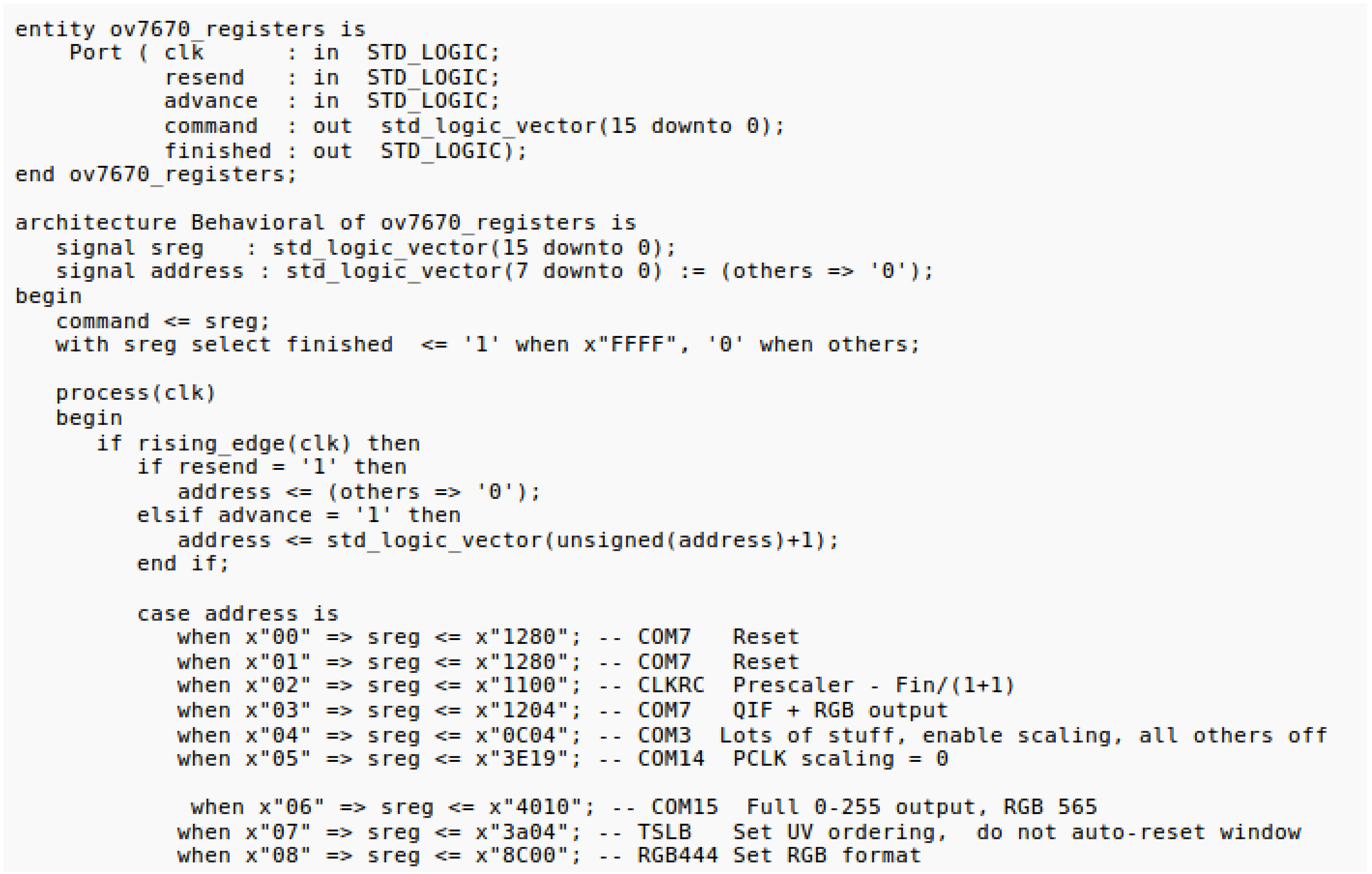


```
   send <= not finished;
   Inst_i2c_sender: i2c_sender PORT MAP(
      clk   => clk,
      taken => taken,
      siod  => siod,
      sioc  => sioc,
      send  => send,
      id    => camera_address,
      reg   => command(15 downto 8),
      value => command(7 downto 0)
   );

   reset <= '1';                        -- Normal mode
   pwdn  <= '0';                        -- Power device up
   xclk  <= sys_clk;

   Inst_ov7670_registers: ov7670_registers PORT MAP(
      clk      => clk,
      advance  => taken,
      command  => command,
      finished => finished,
      resend   => resend
   );

   process(clk)
   begin
      if rising_edge(clk) then
         sys_clk <= not sys_clk;
      end if;
   end process;
nd Behavioral;
```

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- ov7670_registers.vhd

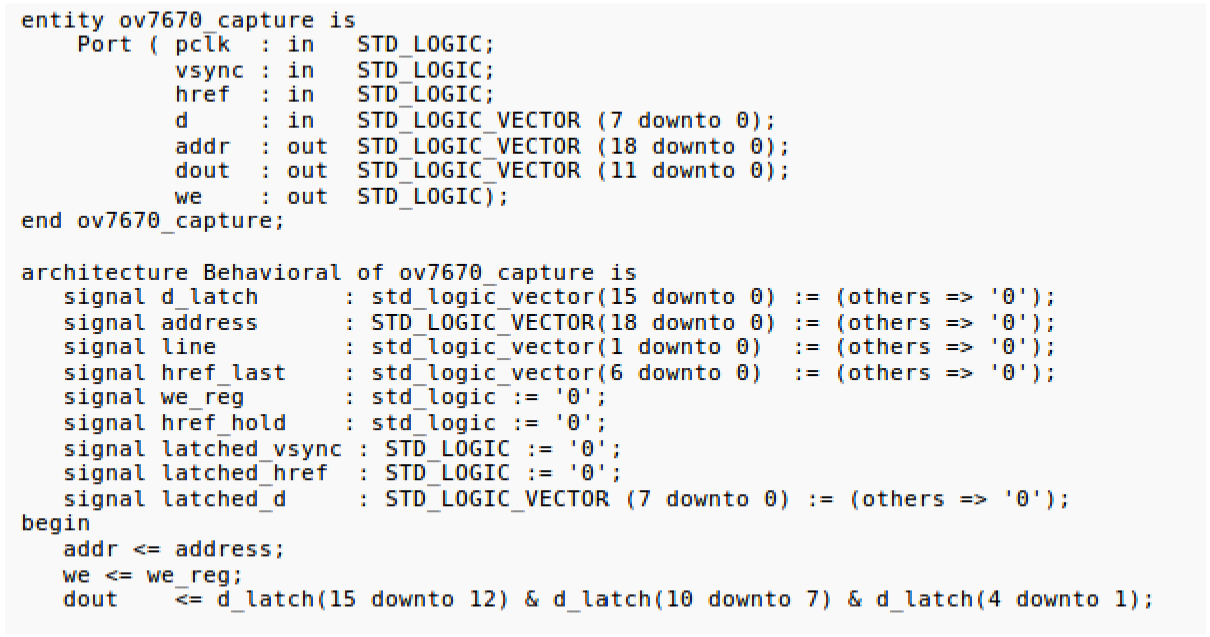


```
entity ov7670_registers is
    Port ( clk      : in  STD_LOGIC;
           resend   : in  STD_LOGIC;
           advance  : in  STD_LOGIC;
           command  : out  std_logic_vector(15 downto 0);
           finished : out  STD_LOGIC);
end ov7670_registers;

architecture Behavioral of ov7670_registers is
   signal sreg    : std_logic_vector(15 downto 0);
   signal address : std_logic_vector(7 downto 0) := (others => '0');
begin
   command <= sreg;
   with sreg select finished  <= '1' when x"FFFF", '0' when others;

   process(clk)
   begin
      if rising_edge(clk) then
         if resend = '1' then
            address <= (others => '0');
         elsif advance = '1' then
            address <= std_logic_vector(unsigned(address)+1);
         end if;

         case address is
            when x"00" => sreg <= x"1280"; -- COM7   Reset
            when x"01" => sreg <= x"1280"; -- COM7   Reset
            when x"02" => sreg <= x"1100"; -- CLKRC  Prescaler - Fin/(1+1)
            when x"03" => sreg <= x"1204"; -- COM7   QIF + RGB output
            when x"04" => sreg <= x"0C04"; -- COM3  Lots of stuff, enable scaling, all others off
            when x"05" => sreg <= x"3E19"; -- COM14  PCLK scaling = 0

             when x"06" => sreg <= x"4010"; -- COM15  Full 0-255 output, RGB 565
            when x"07" => sreg <= x"3a04"; -- TSLB   Set UV ordering,  do not auto-reset window
            when x"08" => sreg <= x"8C00"; -- RGB444 Set RGB format
```

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•Control de OV7670

- ov7670_capture.vhd

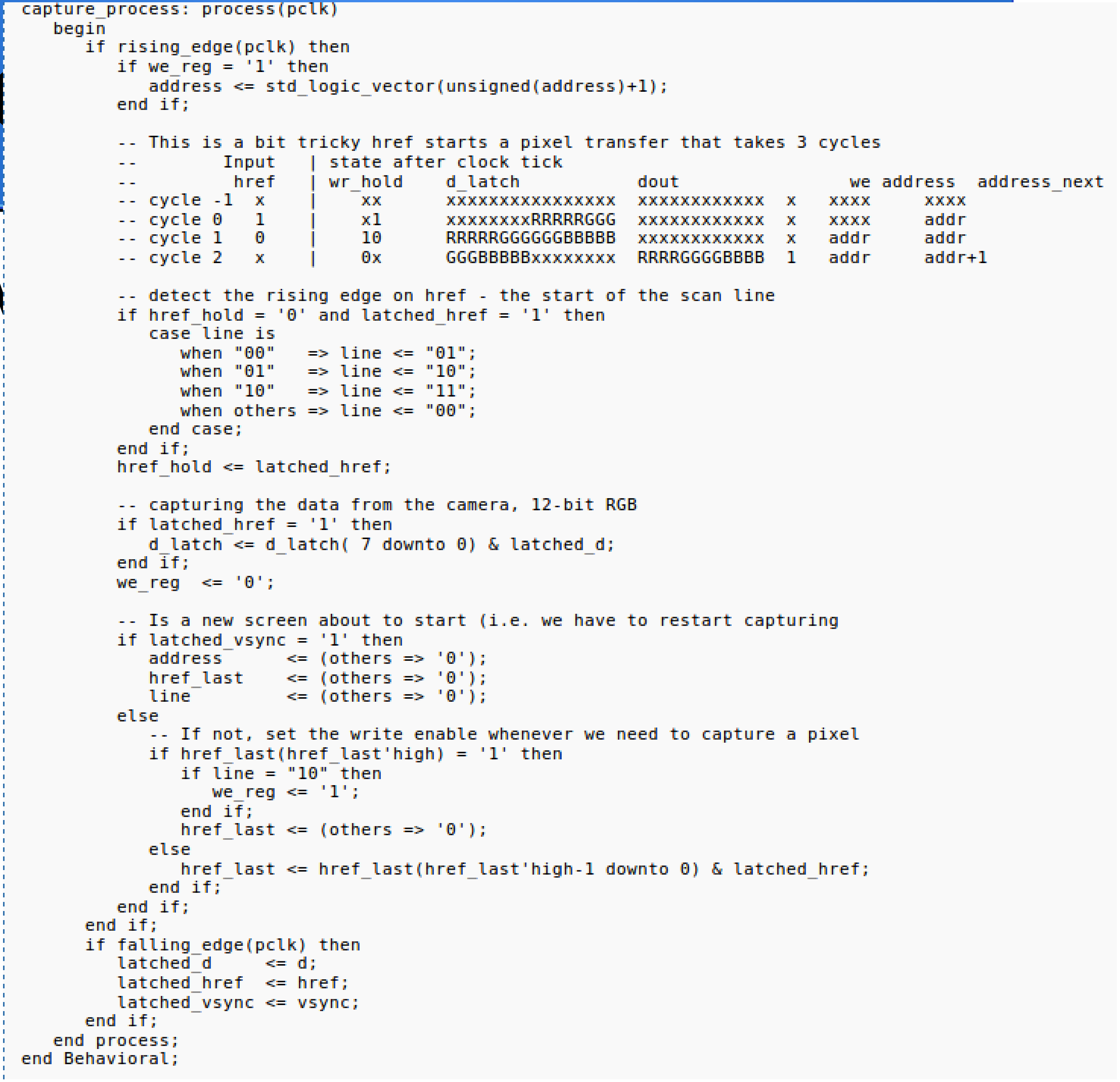


```
entity ov7670_capture is
    Port ( pclk  : in   STD_LOGIC;
           vsync : in   STD_LOGIC;
           href  : in   STD_LOGIC;
           d     : in   STD_LOGIC_VECTOR (7 downto 0);
           addr  : out  STD_LOGIC_VECTOR (18 downto 0);
           dout  : out  STD_LOGIC_VECTOR (11 downto 0);
           we    : out  STD_LOGIC);
end ov7670_capture;

architecture Behavioral of ov7670_capture is
   signal d_latch      : std_logic_vector(15 downto 0) := (others => '0');
   signal address      : STD_LOGIC_VECTOR(18 downto 0) := (others => '0');
   signal line         : std_logic_vector(1 downto 0)  := (others => '0');
   signal href_last    : std_logic_vector(6 downto 0)  := (others => '0');
   signal we_reg       : std_logic := '0';
   signal href_hold    : std_logic := '0';
   signal latched_vsync : STD_LOGIC := '0';
   signal latched_href  : STD_LOGIC := '0';
   signal latched_d     : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
begin
   addr <= address;
   we <= we_reg;
   dout    <= d_latch(15 downto 12) & d_latch(10 downto 7) & d_latch(4 downto 1);
```

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera

•C

• OV



```
capture_process: process(pclk)
   begin
      if rising_edge(pclk) then
         if we_reg = '1' then
            address <= std_logic_vector(unsigned(address)+1);
         end if;

         -- This is a bit tricky href starts a pixel transfer that takes 3 cycles
         --        Input   | state after clock tick
         --         href   | wr_hold    d_latch           dout                we address  address_next
         -- cycle -1  x    |    xx      xxxxxxxxxxxxxxxx  xxxxxxxxxxxx  x    xxxx     xxxx
         -- cycle 0   1    |    x1      xxxxxxxxRRRRRGGG  xxxxxxxxxxxx  x    xxxx     addr
         -- cycle 1   0    |    10      RRRRRGGGGGGBBBBB  xxxxxxxxxxxx  x    addr     addr
         -- cycle 2   x    |    0x      GGGBBBBBxxxxxxxx  RRRRGGGGBBBB  1    addr     addr+1

         -- detect the rising edge on href - the start of the scan line
         if href_hold = '0' and latched_href = '1' then
            case line is
               when "00"   => line <= "01";
               when "01"   => line <= "10";
               when "10"   => line <= "11";
               when others => line <= "00";
            end case;
         end if;
         href_hold <= latched_href;

         -- capturing the data from the camera, 12-bit RGB
         if latched_href = '1' then
            d_latch <= d_latch( 7 downto 0) & latched_d;
         end if;
         we_reg  <= '0';

         -- Is a new screen about to start (i.e. we have to restart capturing
         if latched_vsync = '1' then
            address      <= (others => '0');
            href_last    <= (others => '0');
            line         <= (others => '0');
         else
            -- If not, set the write enable whenever we need to capture a pixel
            if href_last(href_last'high) = '1' then
               if line = "10" then
                  we_reg <= '1';
               end if;
               href_last <= (others => '0');
            else
               href_last <= href_last(href_last'high-1 downto 0) & latched_href;
            end if;
         end if;
      end if;
      if falling_edge(pclk) then
         latched_d     <= d;
         latched_href  <= href;
         latched_vsync <= vsync;
      end if;
   end process;
end Behavioral;
```

Tomado de http://hamsterworks.co.nz/mediawiki/index.php/OV7670_camera