TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

THỰC TẬP TỐT NGHIỆP

Thiết kế camera không dây

NGUYỄN TIẾN ĐẠT
dat.nt173727@sis.hust.edu.vn

TRẦN QUANG HUY
huy.tq173952@sis.hust.edu.vn

Ngành Kỹ thuật điều khiển và tự động hóa
Chuyên ngành Kỹ thuật đo và tin học công nghiệp

Giảng viên hướng dẫn: TS. Nguyễn Đại Dương

Chữ ký của GVHD

Bộ môn: Kỹ thuật đo và tin học công nghiệp
Viện: Điện

HÀ NỘI, 1/2022

ĐỀ TÀI THỰC TẬP

Làm camera không dây. Dùng cảm biến camera kết nói với vi xử lý, chụp ảnh rồi gửi qua wifi về máy tính. Vấn đề là giải quyết việc truyền dữ liệu với tốc độ cao real-time.

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cảm ơn

Đầu tiên em xin gửi lời cảm ơn vô cùng sâu sắc và chân thành tới TS. Nguyễn Đại Dương, người đã hướng dẫn em rất tận tình trong suốt quá trình học tập cũng như thực hiện đề tài này.

Do thời gian và kiên thức có hạn nên không thể tránh khỏi những thiếu sót nhất định. Em rất mong nhận được ý kiến đóng góp quý báu của thầy.

Em xin chân thành cảm ơn!

Tóm tắt nội dung đề tài

Báo cáo đề ra mục tiêu tổng quát là:

Nghiên cứu, thiết kế và phát triển một thiết bị camera không dây.

Để đạt được mục tiêu này, các vấn đề chính sau đây cần được giải quyết:

- Thiết kế lắp ráp hệ thống bắt ảnh gồm Camera OV7670 với Kit vi điều khiển ESP32.
- Phát triển phần mềm nhúng cho phép cấu hình hệ thống và bắt các khung ảnh của luồng video.
- Truyền dữ liệu ảnh về máy tính, hiển thị hình ảnh theo thời gian thực.

Sinh viên thực hiện
Ký và ghi rõ họ tên

MỤC LỤC

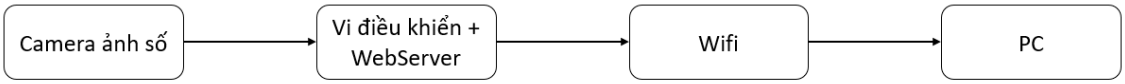
DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

CHƯƠNG 1. TỔNG QUAN VỀ HỆ THỐNG THU THẬP VÀ XỬ LÝ ẢNH DÙNG VI ĐIỀU KHIỂN ESP32

1.1 Giới thiệu chung

Một hệ thống thu thập và xử lý ảnh dùng vi điều khiển có sơ đồ khối điển hình như sau:



Hình 1.1 Sơ đồ khối hệ thống thu thập xử lý ảnh

Hệ gồm có các thành phần sau:

- Camera ảnh số: là cảm biến biến đổi các pixel điểm ảnh (màu hoặc xám) ra thành các số liệu. Tốc độ biến đổi phải đủ nhanh để đáp ứng yêu cầu xử lý theo thời gian thực.
- Vi điều khiển: có nhiệm vụ chính tạo xung nhịp cơ bản để đồng bộ các tín hiệu (cột VSYNC và dòng HYSNC) cho camera, thu thập các dữ liệu hình ảnh đưa vào bộ nhớ đệm ra màn hình (hoặc truyền qua các kênh thông tin, VD: cổng USART của PC, ra kênh Wifi-TCP/IP, v.v…).

Trong báo cáo này, vi điều khiển được xử dụng là loại ESP32 và camera OV7670.

1.2 Giới thiệu về ESP32

ESP32 là một vi điều khiển giá rẻ, năng lượng thấp có hỗ trợ WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

1.2.1 Cấu hình của ESP32

CPU

- CPU: Xtensa Dual-Core LX6 microprocessor.
- Chạy hệ 32 bit.
- Tốc độ xử lý 160MHZ up to 240 MHz.
- Tốc độ xung nhịp đọc flash chip 40mhz --> 80mhz (tùy chỉnh khi lập trình).
- RAM: 520 KByte SRAM.
- 520 KB SRAM liền chip (trong đó 8 KB RAM RTC tốc độ cao – 8 KB RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

Hỗ trợ 2 giao tiếp không dây

- Wi-Fi: 802.11 b/g/n/e/i.
- Bluetooth: v4.2 BR/EDR and BLE.

Hỗ trợ tất cả các loại giao tiếp:

- 8-bit DAC (digital to analog) 2 cổng.
- Analog (ADC) 12-bit 16 cổng.
- I²C – 2 cổng.
- UART – 3 cổng.
- SPI – 3 cổng (1 cổng cho chip FLASH).
- I²S – 2 cổng.
- SD card /SDIO/MMC host.
- Slave (SDIO/SPI).
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support.
- CAN bus 2.0.
- IR (TX/RX).
- Băm xung PWM (tất cả các chân)
- Ultralow power analog pre-amplifier.

Cảm biến tích hợp trên chip esp32:

- 1 cảm biến Hall (cảm biến từ trường).
- 1 cảm biến đo nhiệt độ.
- Cảm biến chạm (điện dung) với 10 đầu vào khác nhau.

Bảo mật:

- IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI.
- Secure boot.
- Flash encryption.
- 1024-bit OTP, up to 768-bit for customers.
- Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG).

Nguồn điện hoạt động:

- Nhiệt độ hoạt động -40 + 85C.
- Điện áp hoạt động: 2.2-3.6V.
- Số cổng GPIO: 34.

1.2.2 Điểm mạnh của ESP32 so với các dòng khác

Có thể nói ESP32 là sự nâng cấp hoàn hảo của ESP8266, với ESP8266 phù hợp với các dự án nhỏ và tiết kiệm chi phí. ESP32 lại phù hợp với các dự án phức tạp hơn, tốc độ xử lý cao hơn và tích hợp nhiều ngoại vi mạnh mẽ hơn.

ESP8266 là 17 chân GPIO, ADC độ phân giải 10 bit, 8 kênh PWM mềm trong khi đó ESP 32 có tới 30/36 chân GPIO, 18 kênh ADC độ phân giải 12-bit, 16 kênh PWM mềm, Touch Sensor, Hall Effect Sensor, Ethernet MAC Interface, Cảm biến nhiệt độ được tích hợp sẵn.

Về bộ nhớ ESP32 có thêm 4MB External Flash và 520KB SRAM (static random access memory) trong đó 8 KB RAM RTC tốc độ cao – 8 KB RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

ESP32 hỗ trợ Bluetooth 4.2 và BLE (Bluetooth Low Energy). Việc hỗ trợ cả bluetooth khiến ESP32 có thể tương tác với các thiết bị như là bàn phím, chuột, điện thoại khi mà không có wifi.

UltraLow Power giải quyết vấn đề năng lượng cho ESP bởi vì sử dụng Wi-Fi sẽ rất ngốn điện đặc biệt khi chúng ta sử dụng pin phải tính toán rất kĩ.

1.3 Giới thiệu Camera OV7670

Báo cáo sử dụng Camera OV7670 no FIFO (First In, First Out) là loại camera giá rẻ nhưng có tính năng đủ cao để sử dụng trong thí nghiệm.

1.3.1 Giới thiệu chung

OV7670 là một cảm biến ảnh kết hợp với xử lý tín hiệu số (DSP), cho độ phân giải VGA 640x480, tốc độ khung hình lên tới 30 fps. Hình ảnh thu nhận được có thể được tiền xử lý bởi khối DSP trước khi được truyền đi. Việc cấu hình chế độ hoạt động của OV7670 được cấu hình qua Bus điều khiển camera tuần tự SCCB.

Hình 1.2 Camera OV7670 no FIFO

Chức năng các chân tín hiệu Camera OV7670 no FIFO trong Hình 1.2 được mô tả trong Bảng 1.1 dưới đây:

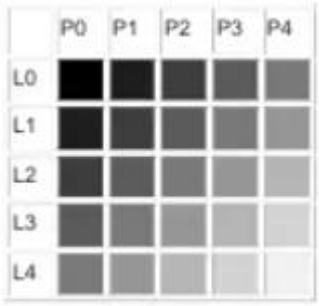
Bảng 1.1 Chức năng các chân tín hiệu của camera OV7670

Chân	Loại	Mô tả
VDD	Nguồn	Nguồn 3.3V
GND	Nguồn	Ground
SDIOC	Đầu vào	Tín hiệu xung clock SCCB
SDIOD	Đầu vào	Tín hiệu SCCB Data
VSYNC	Đầu ra	Xung đồng bộ mành
HREF	Đầu ra	Xung đồng bộ dòng
PCLK	Đầu ra	Tần số Pixel clock
XCLK	Đầu ra	Tần số cấp cho Camera hoạt động
D0-D7	Đầu ra	Tín hiệu video ra song song 8 bit
RESET	Đầu vào	Tín hiệu Reset (tích cực mức thấp)
PWDN	Đầu vào	Tắt nguồn Power down (tích cực mức cao)

1.3.2 Tín hiệu hình ảnh của Camera OV7670

Trước khi đi vào mô tả tín hiệu của Camera OV7670, cần tìm hiểu khái niệm video và hình ảnh được biểu diễn ở định dạng kỹ thuật số.

Một đoạn video là một chuỗi các khung hình, một khung hình frame là một hình ảnh tĩnh chụp tại một thời điểm nhất định. Một khung hình được chia làm các dòng(line), mỗi dòng được chia thành các điểm ảnh(pixel). Một điểm ảnh là một phần nhỏ của hình ảnh kỹ thuật số, và nó trông giống như một dấu chấm màu.



Hình 1.3 Điểm ảnh 5x5

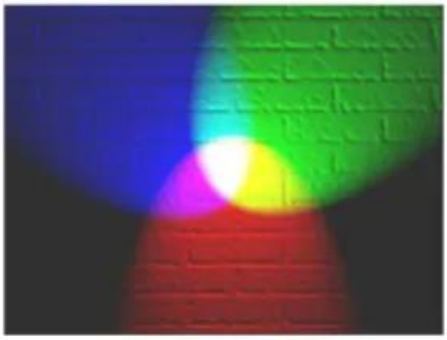
Ví dụ, hình có 5 dòng, mỗi dòng có 5 pixel. Điều này có nghĩa là hình ảnh có độ phân giải điểm ảnh 5x5. Đây là ảnh đơn sắc (đa mức xám), ngoài ra cũng có những ảnh màu. Các màu của hình ảnh có thể được mã hóa trong các định dạng khác nhau như RGB, YUV.

1.3.2.1. Định dạng ảnh đa mức xám (Monochrome)

Trong ảnh đa mức xám, mỗi điểm ảnh được lưu trữ bởi 8 bit số liệu, đại diện cho mức độ xám trong dải từ 0 đến 255. Giá trị 0 là màu đen, 255 là màu trắng và các giá trị trung gian là màu xám.

1.3.2.2. Định dạng ảnh RGB

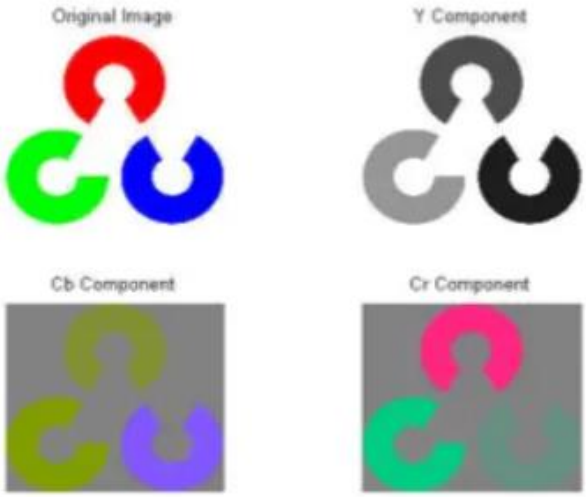
Trong thực tế, màu bất kỳ có thể được tạo bởi sự kết hợp của ba nguồn ánh sáng đỏ (Red), xanh lá cây (Green) và màu xanh dương (Blue) với những cường độ khác nhau. Cách tiếp cận này được gọi là mô hình màu RGB. Sử dụng mô hình này, mỗi điểm ảnh phải lưu trữ ba giá trị cường độ của ánh sáng đỏ, xanh lá cây và xanh dương. Các định dạng phổ biến nhất là RGB888, ở định dạng này mỗi điểm ảnh được lưu trữ trong 24 bit, mỗi kênh màu đỏ, xanh lá cây và màu xanh dương được lưu trữ trong 8 bit. Cường độ của mỗi ánh sáng thành phần có thể trong dải từ 0 đến 255, trong đó 0 là không có ánh sáng, và 255 là cường độ sáng tối đa.



Hình 1.4 Mô hình màu RGB

Các định dạng RGB được sử dụng bởi OV7670 là RGB565, RGB555 và RGB444. Sự khác biệt với định dạng RGB888 là số bit được gán cho mỗi kênh. Ví dụ, trong định dạng RGB565, kênh màu đỏ được lưu trữ với 5 bit, kênh màu xanh lá cây là 6 bit và kênh màu xanh dương là 5 bit. Các định dạng RGB565, RGB555 và RGB444 tốn ít bộ nhớ khi lưu trữ nhưng làm giảm số lượng màu sắc so với định dạng RGB888.

1.3.2.3. Định dạng ảnh YCbCr



Hình 1.5 Phân giải của một hình ảnh thành các thành phần Y, Cb, Cr

YCbCr là một định dạng trong đó một màu RGB có thể được mã hóa. Y hoặc độ sáng thành phần là lượng ánh sáng trắng của một màu sắc, và Cb và Cr là những thành phần chroma, tương ứng mã hóa các cấp độ màu xanh và màu đỏ tương đối so với các thành phần độ sáng.

Hình 1.5 thể hiện hình ảnh phân giải ảnh màu gốc YcbCr thành cách thành phần Y, Cb, Cr. Có thể thấy kênh Y mã hóa các mức độ màu xám của hình ảnh. Vì vậy, cách dễ nhất để có được một hình ảnh đa mức xám từ OV7670 là trích xuất kênh Y của định dạng YCbCr.

Giống như định dạng RGB, định dạng YCbCr cũng lưu trữ mỗi kênh là 8 bit (0-255), có thể chuyển đổi từ YCbCr sang RGB sử dụng các biểu thức sau đây:

$$R = Y + 1{,}402\ (Cr - 128)$$

$$G = Y - 0{,}34414\ (Cb - 128) - 0{,}71414\ (Cr - 128)$$

$$B = Y + 1{,}772\ (Cb - 128)$$

Camera OV7670 sử dụng định dạng YCbCr422, định dạng này được lưu trữ thành các từ theo Bảng 1.2

Bảng 1.2 Dữ liệu hình ảnh được lưu trữ thành từ (4 byte)

	Byte 0	Byte 1	Byte 2	Byte 3
Word 0	Cb0	Y0	Cr0	Y1
Word 1	Cb2	Y2	Cr2	Y3
Word 2	Cb4	Y4	Cr4	Y5

Hoặc tương đương, các dữ liệu đến theo thứ tự trong Bảng 1.3

Bảng 1.3 Thứ tự dữ liệu đến dạng YCbCr422

Thứ tự	Byte
1st	Cb0
2nd	Y0
3rd	Cr0
4th	Y1
5th	Cb2
6th	Y2
7th	Cr2
8th	Y3
…	…

Bảng 1.4 Các điểm ảnh YCbCr422

Pixel 0	Y0 Cb0 Cr0
Pixel 1	Y1 Cb0 Cr0
Pixel 2	Y2 Cb2 Cr2
Pixel 3	Y3 Cb2 Cr2
Pixel 4	Y4 Cb4 Cr4
Pixel 5	Y5 Cb4 Cr4

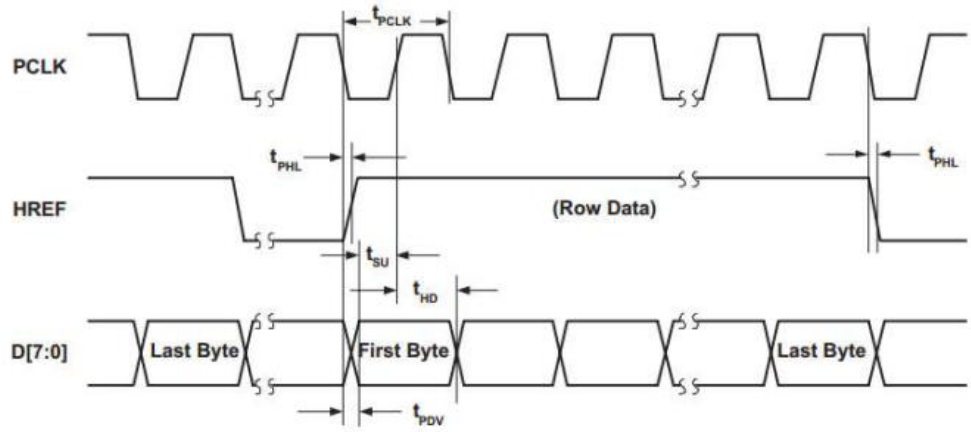
Các điểm ảnh thực tế như Bảng 1.4, mỗi điểm ảnh gồm 3 byte (ví dụ pixel 0 gồm 3 byte Y0, Cb0 và Cr0) như trong các định dạng RGB. Nhưng trong định dạng YCbCr422, các kênh Cb và Cr được chia sẻ giữa hai điểm ảnh liên tiếp (ví dụ pixel 0 và 1 dùng chung phần Cb0 và Cr0). Do đó hai điểm ảnh được "nén" thành 4 byte hoặc 32 bit, điều này có nghĩa là trung bình mỗi điểm ảnh được lưu trữ như là 2 byte hoặc 16 bit. Từ ví dụ trên, 3 từ (12 byte) lưu trữ 6 pixel. Ưu điểm của định dạng YCbCr là kênh Y là hình ảnh đa mức xám, trong khi ở định dạng RGB sẽ cần lấy trung bình của 3 kênh để có được hình ảnh đa mức xám.

1.3.2.4. Tín hiệu video của OV7670

Trước hết, để cho Camera OV7670 hoạt động, cần cấp một tín hiệu xung clock đến chân XCLK. Theo datasheet, xung clock này phải có tần số trong dải từ 10 ÷ 48 MHz.

Nếu vi điều khiển có đầu ra timer hoặc dao động, có thể sử dụng để cấp xung clock cho OV7670. Nếu vi điều khiển không có khả năng tạo ra xung clock thích hợp, có thể sử dụng một nguồn dao động ngoài cấp cho OV7670.

Sau khi có tín hiệu xung clock đến chân XCLK, Camera OV7670 sẽ điều khiển xung đồng bộ VSYN, HREF, PCLK và gửi dữ liệu D0÷D7 dạng song song 8 bit. Giản đồ thời gian các tín hiệu xung clock đồng bộ dòng được mô tả trong Hình 1.6

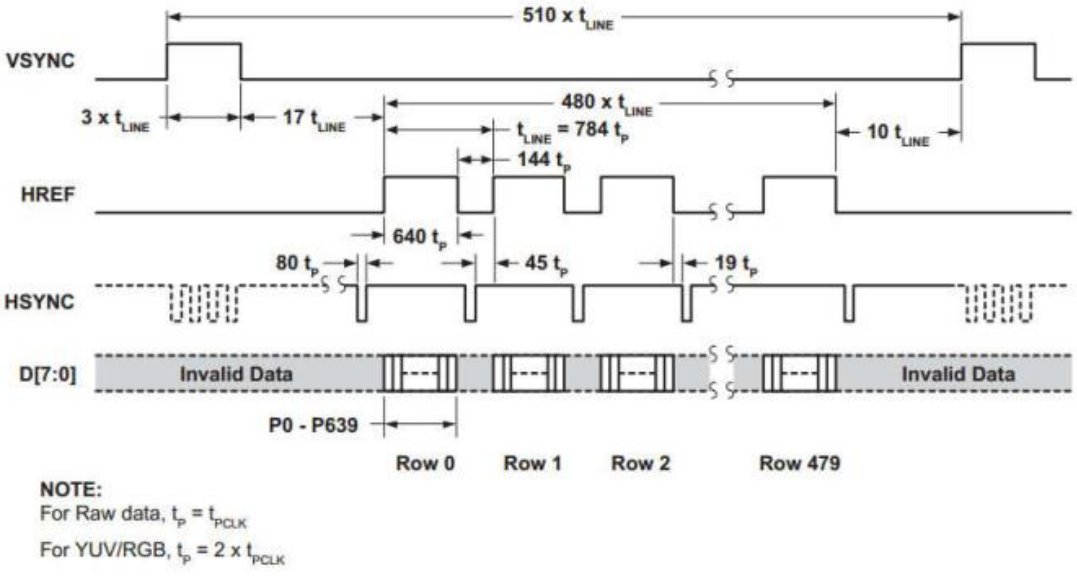


Hình 1.6 Đồng bộ dòng

Điều đầu tiên cần chú ý, dữ liệu D0÷D7 phải được lấy mẫu tại sườn lên của tín hiệu PCLK. Thứ hai, D0÷D7 phải được lấy mẫu chỉ khi HREF ở mức cao. Ngoài ra, sườn lên của tín hiệu HREF báo hiệu sự bắt đầu của một dòng, sườn xuống của tín hiệu HREF báo hiệu kết thúc của một dòng.

Tất cả các byte D0÷D7 được lấy mẫu khi HREF ở mức cao, tương ứng với các điểm ảnh trong một dòng. Lưu ý rằng một byte không phải là một điểm ảnh, nó phụ thuộc vào định dạng lựa chọn. Theo cài đặt mặc định của Camera OV7670, định dạng hình ảnh là YCbCr422, điều này có nghĩa là trung bình hai byte tương ứng với một điểm ảnh (16 bit/pixel).

Theo Hình 1.7, sườn xuống của tín hiệu VSYNC bắt đầu một frame, sườn lên của tín hiệu VSYNC kết thúc một frame. Khi tín hiệu HSYNC ở mức cao, khối thu nhận 640 pixel, tương ứng một dòng, 480 dòng tương ứng một frame được thu nhận khi tín hiệu VSYNC ở mức thấp.



Hình 1.7 Tín hiệu một frame ảnh VGA (640 x 480)

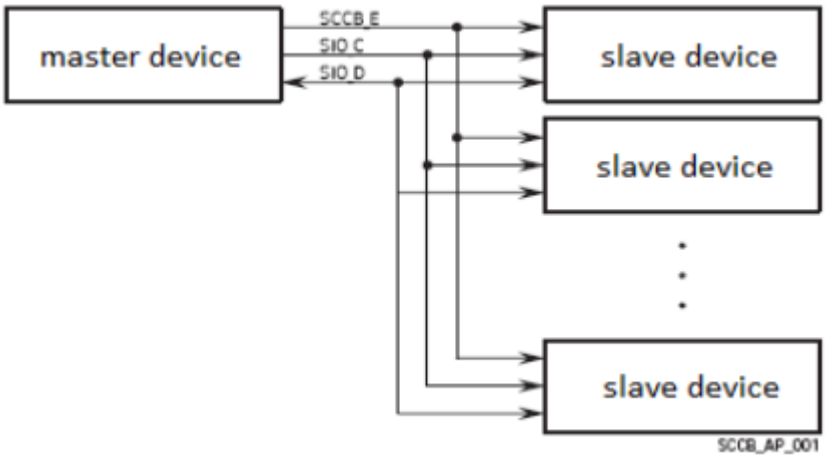
Trên đây đã mô tả toàn bộ quá trình lấy một khung hình (frame). Theo mặc định, tín hiệu PCLK sẽ có cùng một tần số với XCLK, tuy nhiên bằng cách sử dụng

SCCB để cấu hình tỉ lệ chia tần số prescalers và vòng khóa pha PLL, chúng ta có thể tạo ra tín hiệu PCLK có tần số khác nhau.

Thông thường, một tín hiệu PCLK 24 MHz sẽ cho tốc độ khung hình 30 fps, tín hiệu PCLK 12 MHz cho tốc độ khung hình là 15 fps. Điều này độc lập với định dạng hình ảnh (VGA, CIF, QCIF, …).

1.3.3 Bus điều khiển camera tuần tự SCCB

Bus điều khiển camera tuần tự SCCB do hãng OmniVision Technologies phát triển để điều khiển các chức năng của dòng cảm biến camera OmniVision, là một chuẩn bus tuần tự 3 dây, có sự tương thích với giao thức I2C (InterIntegrated Circuit).



Hình 1.8 Sơ đồ khối chức năng SCCB tổng quát sử dụng 3 dây

Trong các ứng dụng master chỉ làm việc với một slave, SCCB có thể hoạt động với hai dây tín hiệu, điều này giúp giảm số chân linh kiện(bỏ chân SCCB_E).

Hình 1.9 Sơ đồ khối chức năng SCCB tổng quát sử dụng 2 dây

1.3.3.1. Chức năng của các chân tín hiệu trong SCCB

Tín hiệu SCCB_E:

Là tín hiệu 1 chiều, kích hoạt ở mức thấp, điều khiển bởi master. Báo hiệu bắt đầu hoặc ngừng truyền dữ liệu.

Mức thay đổi tín hiệu từ cao xuống thấp báo hiệu bắt đầu truyền dữ liệu. Mức thay đổi tín hiệu từ thấp lên cao báo hiệu ngừng truyền dữ liệu.

SCCB_E phải giữ mức thấp trong suốt quá trình truyền dữ liệu. SCCB_E ở mức logic 1 khi hệ thống ở trạng thái idle.

Tín hiệu SIO_C:

Là tín hiệu 1 chiều, tích cực ở mức cao, điều khiển bởi master. Báo hiệu mỗi khi truyền một bit. Truyền dữ liệu bắt đầu khi SIO_C ở mức logic 0.

Mức logic 1 của SIO_C trong quá trình truyền dữ liệu báo hiệu 1bit đã truyền xong. Vì vậy, tín hiệu SIO_D chỉ có thể truyền khi SIO_C ở mức logic 0.

Master đặt SIO_C ở mức logic 1 khi bus ở trạng thái idle.

Chu kỳ truyền 1 bit thông thường khoảng 10 µs => tần số 100KHz.

Tín hiệu SIO_D:

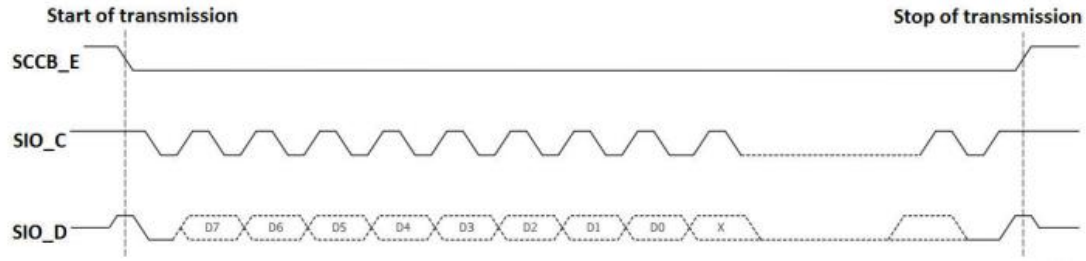
Là tín hiệu dữ liệu 2 chiều, có thể điều khiển được bởi master hoặc slave. Một bit truyền được nhận biết bằng một mức logic 1 của SIO_C.

1.3.3.2. Báo hiệu truyền dữ liệu trong SCCB

Thủ tục truyền dữ liệu trong giao thức SCCB gồm 3 bước:

- Báo hiệu bắt đầu truyền dữ liệu.
- Truyền dữ liệu.
- Báo hiệu kết thúc truyền dữ liệu.

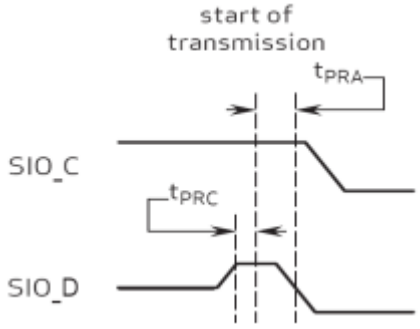
Giản đồ thời gian của quá trình truyền dữ liệu của SCCB 3 dây tổng quát thể hiện trong Hình 1.10



Hình 1.10 Quá trình truyền dữ liệu của SCCB 3 dây

Trong báo cáo sử dụng truyền dữ liệu qua SCCB 2 dây, sau đây sẽ tập trung mô tả giao thức truyền SCCB 2 dây.

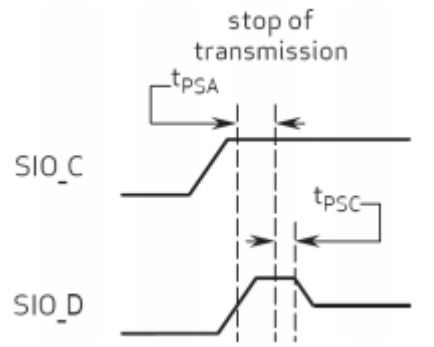
Báo hiệu bắt đầu truyền tín hiệu (Start): Mức tín hiệu SIO_D, SIO_C ở mức logic 1 để nhận biết trạng thái đường truyền.



Hình 1.11 Tín hiệu báo hiệu Start

Start: Sau khi mức tín hiệu SIO_D chuyển trạng thái từ 1 xuống 0, tín hiệu SIO_C chuyển trạng thái từ 1 xuống 0 để báo hiệu bắt đầu truyền.

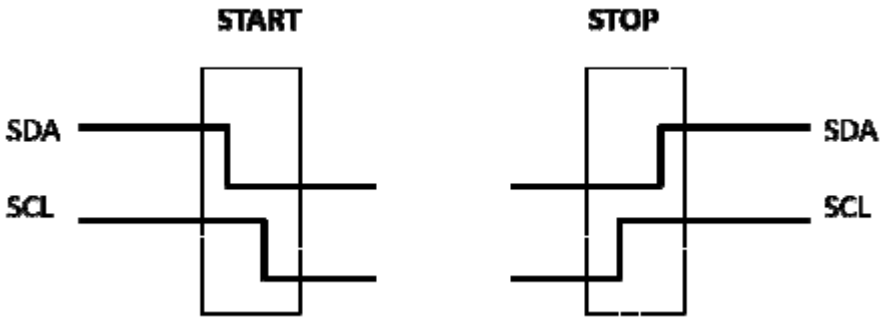
Báo hiệu ngừng truyền dữ liệu (Stop): Sau khi mức tín hiệu SIO_C chuyển trạng thái từ 0 lên 1, tín hiệu SIO_D chuyển trạng thái từ 0 lên 1 để báo hiệu ngừng truyền dữ liệu.



Hình 1.12 Tín hiệu báo hiệu Stop

Stop: Tín hiệu SIO_D từ mức logic 0 lên 1 khi tín hiệu SIO_C ở mức logic 1

So sánh với tín hiệu báo hiệu bắt đầu và kết thúc truyền dữ liệu của I2C



Hình 1.13 Tín hiệu báo hiệu Start/Stop của I2C

Start: Tín hiệu SDA từ mức logic 1 xuống 0 khi tín hiệu SCL ở mức logic 1.

Stop: Tín hiệu SDA từ mức logic 0 lên 1 khi tín hiệu SCL ở mức logic 1.

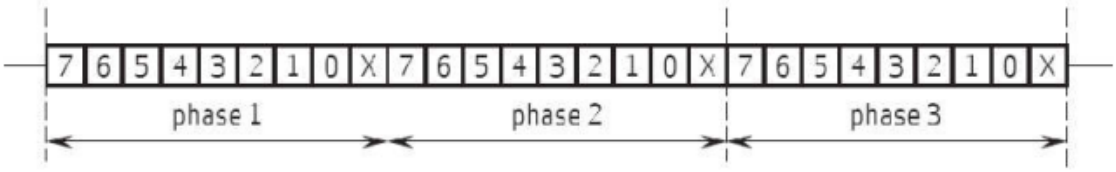
So sánh cho thấy tín hiệu báo hiệu truyền của SCCB giống I2C.

Tín hiệu SIO_C tương đương với tín hiệu SCL.

Tín hiệu SIO_D tương đương với tín hiệu SDA.

1.3.3.3. Chu kỳ truyền dữ liệu trong SCCB

Đơn vị truyền dữ liệu cơ bản trong SCCB được gọi là pha, một pha truyền dữ liệu gồm có 9 bit: 8 bit dữ liệu truyền tuần tự, bit thứ 9 là Don’t care bit hoặc NA bit phụ thuộc vào việc truyền dữ liệu là đọc hay ghi dữ liệu. Mục đích của bit thứ 9 là để nhận biết truyền dữ liệu thành công. Bit có trọng số cao MSB luôn được truyền đầu tiên.



Hình 1.14 Pha truyền dữ liệu trong SCCB

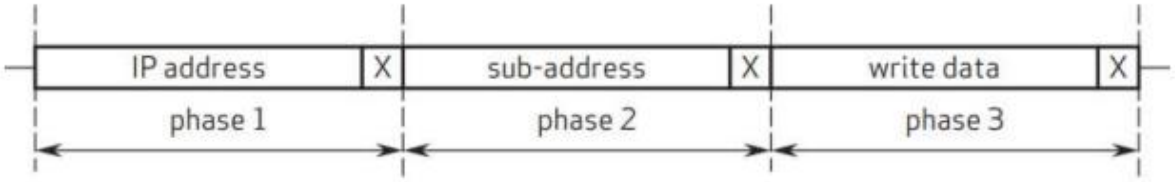
Dưới đây sẽ mô tả 3 loại truyền dữ liệu:

- Chu kỳ ghi dữ liệu 3 pha.
- Chu kỳ ghi dữ liệu 2 pha.

- Chu kỳ đọc dữ liệu 2 pha.

1.3.3.4. Chu kỳ ghi dữ liệu 3 pha:

Là một chu kỳ ghi dữ liệu đầy đủ, master thực hiện ghi một byte dữ liệu lên thanh ghi slave.



Hình 1.15 Chu kỳ ghi dữ liệu 3 pha trong SCCB

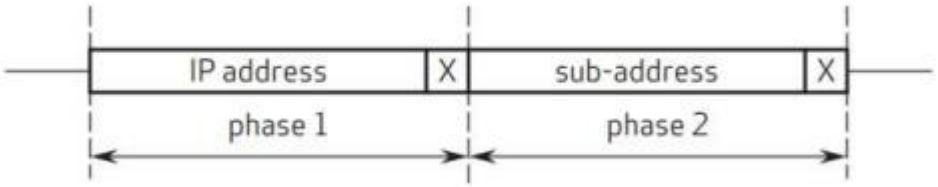
Pha 1: IP address => truyền địa chỉ slave.

Pha 2: sub address => truyền địa chỉ thanh ghi của slave.

Pha 3: write data => ghi dữ liệu 8 bit vào thanh ghi của slave.

Bit thứ 9 trong cả 3 pha là Don’t care bit.

1.3.3.5. Chu kỳ ghi dữ liệu 2 pha:



Hình 1.16 Chu kỳ ghi dữ liệu 2 pha trong SCCB

Chu kỳ ghi dữ liệu 2 pha phải theo sau bởi một chu kỳ đọc dữ liệu 2 pha. Mục đích của chu kỳ ghi dữ liệu 2 pha là xác định thanh ghi của slave để master đọc trong chu kỳ đọc 2 pha tiếp theo.

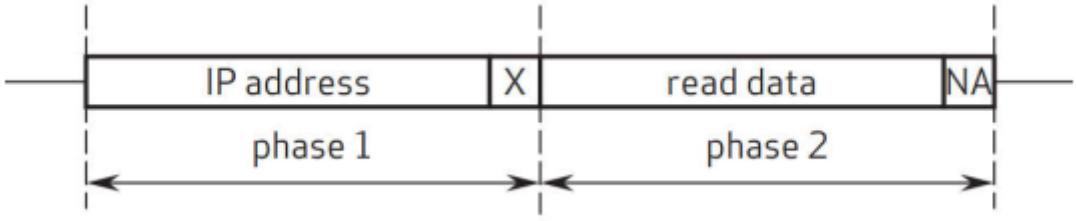
Pha 1: IP address => truyền địa chỉ slave.

Pha 2: sub address => truyền địa chỉ thanh ghi của slave.

Bit thứ 9 trong cả 2 pha là Don’t care bit.

1.3.3.6. Chu kỳ đọc dữ liệu 2 pha:

Bản thân chu kỳ đọc 2 pha không thể nhận biết được địa chỉ thanh ghi của slave. Do đó, trước khi thực hiện chu kỳ đọc dữ liệu 2 pha, phải có một chu kỳ dữ liệu 3 pha hoặc 2 pha để slave nhận biết địa chỉ thanh ghi cần đọc dữ liệu.



Hình 1.17 Chu kỳ đọc dữ liệu 2 pha trong SCCB

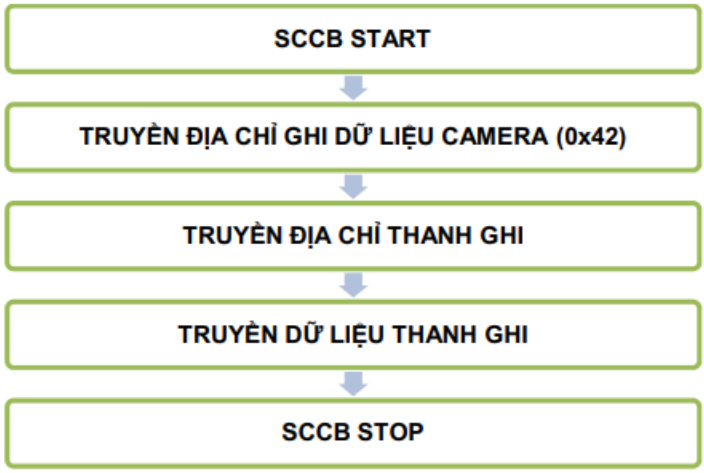
Pha 1: IP address => truyền địa chỉ slave. Bit thứ 9 trong pha 1 là Don’t care bit.

Pha 2: read data => đọc dữ liệu từ slave. Bit thứ 9 trong pha 2 của chu kỳ đọc NA. Master phải kéo tín hiệu NA lên mức 1.

So sánh bit thứ 9 trong giao thức I2C và SCCB: đều dùng để báo hiệu truyền dữ liệu thành công. Trong giao thức I2C, bit thứ 9 là bit ACK luôn ở mức logic 0, trong giao thức SCCB khi master truyền dữ liệu, bit thứ 9 là Don't care bit có mức logic 0, khi master đọc dữ liệu, bit thứ 9 là NA bit có mức logic 1.

1.3.3.7. Ghi dữ liệu vào thanh ghi OV7670

Để ghi dữ liệu vào thanh ghi của Camera OV7670 theo giao thức SCCB, sử dụng chu kỳ truyền dữ liệu 3 pha.



Hình 1.18 Ghi dữ liệu vào thanh ghi OV7670

1.3.3.8. Đọc dữ liệu từ thanh ghi OV7670

Để đọc dữ liệu thanh ghi của Camera OV7670 theo giao thức SCCB, sử dụng hai chu kỳ truyền dữ liệu:

Chu kỳ ghi dữ liệu 2 pha.

Chu kỳ đọc dữ liệu 2 pha.

Hình 1.19 Đọc dữ liệu thanh ghi OV7670

1.3.4 Cấu hình hoạt động của Camera OV7670

Camera OV7670 có nhiều thanh ghi cho phép cấu hình chế độ hoạt động như lựa chọn định dạng hình ảnh, độ phân giải, tốc độ khung hình, tiền xử lý ảnh, cân bằng trắng… Dưới đây sẽ mô tả một số thanh ghi quan trọng trong việc cấu hình chế độ hoạt động của Camera OV7670.

1.3.4.1. Cài đặt tần số dao động nội cho Camera OV7670 hoạt động

Để Camera OV7670 hoạt động, cần cấp xung clock từ bên ngoài đến chân XCLK, đầu tiên tín hiệu clock này được qua vòng khóa pha PLL để nhân tần số, sau đó qua bộ chia để tạo ra xung clock để Camera OV7670 làm việc.

Bộ nhân PLL được điều khiển bởi thanh ghi DBLV có địa chỉ 0x6B.

Bộ chia được điều khiển bởi thanh ghi CLKRC có địa chỉ 0x11.

Bảng 1.5 Các thanh ghi cài đặt tần số dao động nội Camera

Thanh ghi	Địa chỉ	Giá trị mặc định	Mô tả
CLKRC	0x11	0x80	Bit[6] = 0: cho phép đặt tỷ lệ chia tần số Bit[6] = 1: sử dụng clock ngoài Bit[5:0]: hệ chia tần số, trong dải từ [00000] đến [11111], tính theo công thức 2*(CLKCR[5:0]+1)
DBLV	0x6B	0x0A	Bit[7:6]: Điều khiển PLL để nhân tần số 00: Bypass PLL, giữ nguyên tần số 01: Input clock x 4 10: Input clock x 6 11: Input clock x 8

Tần số làm việc của Camera được tính như sau:

$F_{INT\ CLK} = f_{CLK} \times PLL\_Multiplier / 2* (CLKRC[5:0]+1)$

Ví dụ: tần số 16 MHz bên ngoài cấp cho Camera ở chân XCLK, để nhận được tần số dao động nội 24 MHz cấp cho Camera hoạt động, cần cấu hình như sau:

Đặt Bit[6] thanh ghi CLKRC về 0 để cho phép chia tỷ lệ.

Đặt Bit[5:0] thanh ghi CLKRC bằng 1 (00001) để thực hiện chia 4.

Đặt Bit[7:6] thanh ghi DBLV về 10 để PLL nhân 6.

```
// Tan so ESP32 cap cho camera 16MHz
SCCB_write_reg(0x6B,0x80); // nhan tan so voi 6
SCCB_write_reg(0x11, 0x01);// chia tan so cho 2(1+1)
// Tan so lam viec cua camera = 16MHz*6/2(1+1) = 24MHz
```

$F_{INT\ CLK} = f_{CLK} \times PLL\_Multiplier / 2* (CLKRC[5:0]+1)$

$= 16\ MHz \times 6 / 4 = 24\ MHz$

Thông thường, tần số pixel clock ra tại chân PCLK Camera OV7670 có cùng tần số với tần số $F_{INT\ CLK}$. Bằng cách cài đặt một số thanh ghi, chúng ta có thể thay đổi tần số PCLK qua đó thay đổi tốc độ khung hình/giây.

1.3.4.2. Định dạng hình ảnh

Camera OV7670 hỗ trợ 4 định dạng hình ảnh: YCbCr, RGB565/RGB555, Bayer raw RGB và Processed raw RGB. Định dạng hình ảnh ra có thể được cấu hình bằng cách thiết lập thanh ghi các bit của hai thanh ghi COM7 (địa chỉ 0x12) và thanh ghi COM15 (địa chỉ 0x40) theo Bảng 1.6

Bảng 1.6 Thiết lập định dạng hình ảnh cho Camera OV7670

Định dạng hình ảnh	**Pixel Data Output**	**Cài đặt thanh ghi**			
		COM7[2]	COM7[0]	COM15[5]	COM15[4]
Raw Bayer RGB	8 bit R or 8 bit G or 8 bit B	0	1	X	0
Processed BayerRGB	8 bit R or 8 bit G or 8 bit B	1	1	X	0
YUV/YCbCr 4:2:2	8 bit Y, 8 bit U or 8 bit Y, 8 bit V	0	0	X	0
RGB 4:2:2	8 bit G, 8 bit R or 8 bit G, 8 bit B	1	0	X	0
RGB565	5 bit R, 6 bit G, 5 bit B	1	0	0	1
RGB555	5 bit R, 5 bit G, 5 bit B	1	0	1	1

Căn cứ tài liệu Camera OV7670, có thể đặt giá trị một số thanh ghi để có định dạng hình ảnh RGB565 và YUV422 như sau:

Định dạng hình ảnh RGB565:

SCCB_write_reg(0x12, 0x04); // COM7, RGB format

SCCB_write_reg(0x40, 0xD0); // COM15, RGB565

SCCB_write_reg(0x04, 0x0); // COM1, disable CCIR656

SCCB_write_reg(0x8C, 0x0); // disable RGB444

Định dạng hình ảnh YUV422:

SCCB_write_reg(0x12, 0x0); // COM7, YUV mode

SCCB_write_reg(0x40, 0x0); // COM15

SCCB_write_reg(0x04, 0x0); // COM1, disable CCIR656

SCCB_write_reg(0x8C, 0x0); // disable RGB444

Với định dạng YUV, mặc định thứ tự tín hiệu ra là YUYV, nếu cần thay đổi thứ tự tín hiệu ra, thay đổi giá trị các bit thanh ghi COM13[1] địa chỉ 0x3D và thanh ghi TSLB[3] địa chỉ 0x3A theo Bảng 1.7

Bảng 1.7 Thứ tự tín hiệu YUV

Bit TSLB[3]	Bit COM13[1] (0x3D)	Thứ tự
0	0	Y U Y V
0	1	Y V Y U
1	0	U Y V Y
1	1	V Y U Y

1.3.4.3. Cài đặt độ phân giải hình ảnh

Camera OV7670 có độ phân giải mặc định là VGA (640x480), trong thanh ghi COM7 (0x12) có một số độ phân giải đã định nghĩa trước như QVGA, CIF và QCIF. Để lựa chọn các độ phân giải này, cài đặt các bit thanh ghi COM7 như Bảng 1.8

Bảng 1.8 Thiết lập độ phân giải QVGA, CIF, QCIF

Độ phân giải hình ảnh	Cài đặt thanh ghi		
	COM7[5]	COM7[4]	COM7[3]
QVGA (320x240)	0	1	0
CIF (352x288)	1	0	0
QCIF (176x144)	0	0	1

Căn cứ vào Bảng 1.7 Bảng 1.8, có thể thiết lập các giá trị khác nhau của thanh ghi COM7 để chọn độ phân giải và định dạng ảnh:

SCCB_write_reg(0x12, 0x0); // VGA, YUV

SCCB_write_reg(0x12, 0x4); // VGA, RGB

SCCB_write_reg(0x12, 0x14); // QVGA, RGB

SCCB_write_reg(0x12, 0x10); // QVGA, YUV

SCCB_write_reg(0x12, 0x8); // QCIF, YUV

SCCB_write_reg(0x12, 0x0C); // QCIF, RGB

Các độ phân giải này đã được định nghĩa trước trong Camera OV7670.

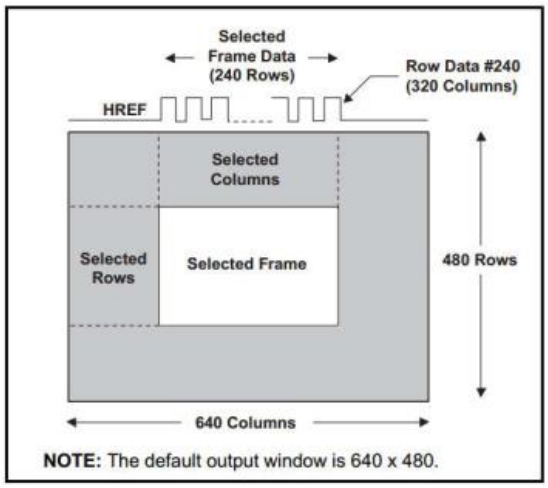
Trong cấu hình hoạt động của Camera, để đảm bảo độ phân giải ra đúng yêu cầu, có thể thay đổi các các thanh ghi:

Đặt các bit của thanh ghi COM14[3], và SCALING_PCLK_DELAY[7].

Đặt bit 0 của thanh ghi TSLB[0] (0x3Ah) về 0 và thay đổi giá trị các thanh ghi điều khiển cửa sổ theo Bảng 1.9

Bảng 1.9 Các thanh ghi thiết lập cửa sổ

Chức năng	Thanh ghi	Địa chỉ thanh ghi
Cột bắt đầu	HSTART[7:0], HREF[2:0]	0x17, 0x32
Cột kết thúc	HSTOP[7:0], HREF[5:3]	0x18, 0x32
Dòng bắt đầu	VSTART[7:0], VREF[2:0]	0x19, 0x03
Dòng kết thúc	VSTOP[7:0], VREF[5:3]	0x1A, 0x03



Hình 1.20 Ví dụ về cửa sổ 320x240

Hình 1.20 cho thấy một ví dụ về cửa sổ frame. Sau khi cài đặt các thanh ghi trong Bảng 1.9 với giá trị thích hợp, tín hiệu ra có độ phân giải là 320x240.

1.3.4.4. Tín hiệu đồng bộ

Camera OV7670 có hai tín hiệu đồng bộ VSYNC và HREF. Tín hiệu VSYNC là tín hiệu liên tục, tín hiệu HREF chỉ có giá trị khi có dữ liệu ra, nếu không có dữ liệu ra, tín hiệu HREF giữ mức thấp hoặc cao phụ thuộc vào cực tính của tín hiệu.

Cực tính của tín hiệu VSYNC, HREF được cài đặt tương ứng bởi các bit của thanh ghi COM10 (địa chỉ 0x15). Thông thường, ứng dụng sử dụng sườn lên của tín hiệu PCLK để lấy dữ liệu khi HREF ở mức cao. Tín hiệu PCLK chạy tự do theo mặc định, có thể đặt bit 5 của thanh ghi COM10[5] ở mức cao để tín hiệu PCLK chỉ chạy khi có tín hiệu HREF.

Bảng 1.10 Thanh ghi điều khiển tín hiệu đồng bộ của Camera

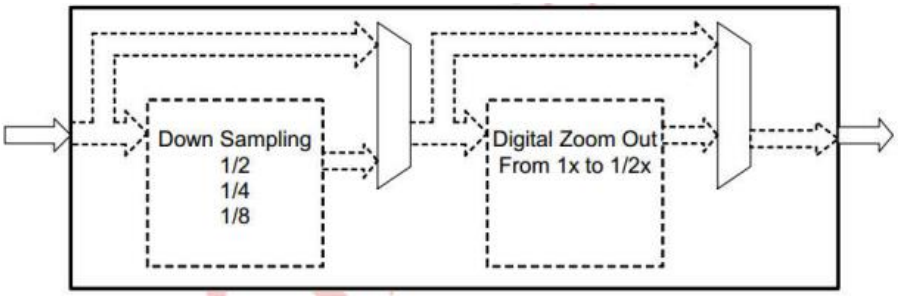
Thanh ghi	**Địa chỉ**	**Giá trị mặc định**	**Mô tả**
COM10	0x15	00	Bit[6]: HREF thay đổi theo tín hiệu HSYNC Bit[5] = 0: PCLK luôn dao động Bit[5] = 1: PCLK không dao động trong khoảng horizontal bank Bit[4] = 0: dữ liệu ra ở sườn xuống PCLK Bit[4] = 1: dữ liệu ra ở sườn lên PCLK Bit[3] = 0: HREF cực tính dương Bit[3] = 1: HREF cực tính âm

1.3.4.5. Tỷ lệ hình ảnh

Camera OV7670 cho phép cài đặt để hình ảnh ra có độ phân giải khác nhau được lấy tỷ lệ từ độ phân giải VGA.

Mạch điều khiển tỷ lệ hình ảnh gồm hai khối, lấy mẫu xuống (Down Sampling) và co dãn ảnh số (Digital Zoom Out) được mô tả trong Hình 1.21

Khối Down Sampling làm giảm kích thước hình ảnh VGA theo tỷ lệ 1/2, 1/4, 1/8. Khối Digital Zoom Out cho phép co giãn ảnh theo hai chiều ngang dọc độc lập.



Hình 1.21 Mạch điều khiển tỷ lệ hình ảnh

Ví dụ: để lấy hình ảnh có độ phân giải 256x128. Đầu tiên, hình ảnh VGA qua khối Down Sampling xuống độ phân giải 320x240 với tỷ lệ 1/2 trên cả hai chiều ngang, dọc. Sau đó khối Digital Zoom Out lấy tỉ lệ từ 320x240 xuống 256x128 bằng cách đặt tỷ lệ 0,8 chiều ngang, 0,53 chiều dọc.

Bảng 1.11 Các thanh ghi cài đặt tỷ lệ hình ảnh

Function	Register	Address	Description
Digital Zoom Enable Bit	COM3[3]	0x0C	0: Bypass 1: Enable
Down Sampling Enable Bit	COM3[2]	0x0C	0: Disable 1: Enable
Down Sampling Related Control Register	SCALING_DCWCTR[7:0]	0x72	See Table 6-2
Pixel Clock Divider	SCALING_PCLK_DIV[3:0]	0x73	DSP Output Clock Divider Bit [3]: 0: Bypass 1: Enable Bit [2:0]: 000: Divider = 1 001: Divider = 2 010: Divider = 4 011: Divider = 8 100: Divider = 16 101: Divider = 32 110: Divider = 64 111: Divider = Not allowed
Horizontal Scaling Ratio	REG74[6:0]	0x74	From 1x (0x20) to 0.5x (0x40) Horizontal Scaling Ratio = 0x20 / (REG74[6:0])
Vertical Scaling Ratio	REG75[6:0]	0x75	From 1x (0x20) to 0.5x (0x40) Vertical Scaling Ratio = 0x20 / (REG75[6:0])
Pixel Clock Delay	SCALING_PCLK_ DELAY[3:0]	0xA2	Original H size / Pixel clock divider - New H size

Bảng 1.12 Thanh ghi điều khiển co giãn ảnh

Function	Register	Address	Description
Option for Vertical Average Calculation	SCALING_DCWCTR[7]	0x72	0: Vertical truncation 1: Vertical rounding
Option for Vertical Down Sampling	SCALING_DCWCTR[6]	0x72	0: Vertical truncation 1: Vertical rounding
Vertical Down Sampling Rate	SCALING_DCWCTR[5:4]	0x72	00: No vertical down sampling 01: Vertical down sample by 2 10: Vertical down sample by 4 11: Vertical down sample by 8
Option for Horizontal Average Calculation	SCALING_DCWCTR[3]	0x72	0: Horizontal truncation 1: Horizontal rounding
Option for Horizontal Down Sampling	SCALING_DCWCTR[2]	0x72	0: Horizontal truncation 1: Horizontal rounding
Horizontal Down Sampling Rate	SCALING_DCWCTR[1:0]	0x72	00: No horizontal down sampling 01: Horizontal down sample by 2 10: Horizontal down sample by 4 11: Horizontal down sample by 8

1.3.4.6. Lọc dải

Cường độ ánh sáng trong phòng thường thay đổi theo tần số nguồn AC. Với nguồn AC có tần số 50Hz, có thể thiết lập cấu hình bộ lọc như sau:

SCCB_write_reg(0x13, 0xe7); //banding filter enable

SCCB_write_reg(0x9d, 0xa5); //50Hz banding filter

SCCB_write_reg(0xa5, 0x02); //3 step for 50hz

SCCB_write_reg(0x3b, 0x0a); //Select 50Hz banding filter

CHƯƠNG 2. LẬP TRÌNH HỆ THỐNG

2.1 Tạo xung clock đến chân XCLK của OV7670

Để Camera OV7670 hoạt động, cần cấp một xung clock đến chân XCLK của OV7670, tần số xung clock từ 10:48 MHz theo datasheet.

Cấu hình chân GPIO5 để tạo dao động 8MHz với hàm PWM_Init như sau:

```
void PWM_Init(void)
{
        ledc_timer_config_t ledc_timer = {
        .duty_resolution = LEDC_TIMER_1_BIT, // resolution of PWM duty
        .freq_hz = PWM_FREQ,                      // frequency of PWM
signal
        .speed_mode = LEDC_HIGH_SPEED_MODE,           // timer mode
        .timer_num = LEDC_TIMER_0,            // timer index
        .clk_cfg = LEDC_USE_APB_CLK,              // Auto select the
source clock
    };
    // Set configuration of timer0 for high speed channels
    ledc_timer_config(&ledc_timer);

    ledc_channel_config_t ledc_channel = {
            .channel    = LEDC_CHANNEL_0,
            .duty       = 1,
            .gpio_num   = PWM_GPIO_NUM,
            .speed_mode = LEDC_HIGH_SPEED_MODE,
            .hpoint     = 0,
            .timer_sel  = LEDC_TIMER_0,
            .intr_type = LEDC_INTR_DISABLE,
    };

    // Set LED Controller with previously prepared configuration

        ledc_channel_config(&ledc_channel);
}
```

2.2 Lập trình SCCB qua I2C

Chế độ hoạt động cho Camera OV7670 được cấu hình bằng cách thiết lập giá trị thanh ghi với giá trị thích hợp qua giao thức SCCB, giao thức này có sự tương thích với giao thức I2C. Trong luận văn sử dụng khối I2C2 giao tiếp với Camera OV7670.

Đầu tiên, phải khai báo chân nối, cấu hình hoạt động của khối I2C2 cho phù hợp với hàm SCCB_init.

```
void SCCB_Init(void)
{
    int i2c_master_port = I2C_MASTER_NUM;
    i2c_config_t conf = {
        .mode = I2C_MODE_MASTER,
        .sda_io_num = I2C_MASTER_SDA_IO,
        .sda_pullup_en = GPIO_PULLUP_ENABLE,
```

```
        .scl_io_num = I2C_MASTER_SCL_IO,
        .scl_pullup_en = GPIO_PULLUP_ENABLE,
        .master.clk_speed = I2C_MASTER_FREQ_HZ,
        // .clk_flags = 0,          /*!< Optional, you can use
I2C_SCLK_SRC_FLAG_* flags to choose i2c source clock here. */
    };
    esp_err_t err = i2c_param_config(i2c_master_port, &conf);
    if (err != ESP_OK) {
        return err;
    }
    i2c_driver_install(i2c_master_port, conf.mode,
I2C_MASTER_RX_BUF_DISABLE, I2C_MASTER_TX_BUF_DISABLE, 0);
}
```

2.3 Lập trình ghi dữ liệu vào thanh ghi OV7670

Để ghi dữ liệu vào thanh ghi của Camera OV7670 theo giao thức SCCB cần sử dụng chu kỳ truyền dữ liệu 3 pha.

Chương trình ghi dữ liệu vào thanh ghi Camera OV7670:

```
void SCCB_write_reg(uint8_t Reg_Addr, uint8_t Data)
{
    int ret;
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, OV7670_Addr << 1 | WRITE_BIT,
ACK_CHECK_EN);
    i2c_master_write_byte(cmd, Reg_Addr, ACK_CHECK_EN);
    i2c_master_write_byte(cmd, Data, ACK_CHECK_EN);
    i2c_master_stop(cmd);
    ret = i2c_master_cmd_begin(I2C_MASTER_NUM, cmd, 1000 /
portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
}
```

2.4 Bắt ảnh từ Camera

Để thu nhận ảnh từ Camera, phải cấu hình chân thu tín hiệu đồng bộ VSYNC, HREF, PCLK, bus dữ liệu D0÷D7 từ Camera ở chế độ input.

```
void OV7670_Pin_Config(void)
{
    //config gpio2 output
    gpio_config_t gpio_initstructure;
    gpio_initstructure.pin_bit_mask = RET_SEL | PWDN_SEL;
    gpio_initstructure.mode = GPIO_MODE_OUTPUT;
    gpio_initstructure.pull_up_en = GPIO_PULLUP_ENABLE;
    gpio_initstructure.pull_down_en = GPIO_PULLDOWN_DISABLE;
    gpio_initstructure.intr_type = GPIO_INTR_DISABLE;
    gpio_config(&gpio_initstructure);

    //config gpio15 input;
    gpio_initstructure.pin_bit_mask = D0_SEL | D1_SEL | D2_SEL | D3_SEL |
D4_SEL | D5_SEL | D6_SEL | D7_SEL | VSYNC_SEL | HREF_SEL | PCLK_SEL;
```

```
    gpio_initstructure.mode = GPIO_MODE_INPUT;
    gpio_config(&gpio_initstructure);
    gpio_set_level(Pin_RET, 1);
    gpio_set_level(Pin_PWDN, 0);
}
```

2.4.1 Bắt ảnh màu RGB565 lưu vào bộ nhớ đệm

Ảnh RGB565 gồm 16 bit/pixel, vi xử lý cần lấy mẫu dữ liệu trong 2 xung clock PCLK để thu được một pixel.

Chương trình bắt ảnh được viết như sau:

```
void Capture_Image(uint16_t Cot, uint16_t Hang)
{
    uint16_t i, x, y, z;
    i = 0;
    while(!gpio_get_level(Pin_VSYNC)); // tin hieu VSYN cao ket thuc 1
frame
    while(gpio_get_level(Pin_VSYNC)); // tin hieu VSYN thap bat dau 1
frame
    y = Hang; //so line
    while(y--)
        {
            x = Cot; //so pixel tren 1 line
            while(!gpio_get_level(Pin_HREF)); // tin hieu HREF cao bat
dau 1 line
            while(x--)
                {
                    while(gpio_get_level(Pin_PCLK));// doi PCLK xuong
thap
                    z = ((gpio_get_level(D7) << 7) | (gpio_get_level(D6)
<< 6) | (gpio_get_level(D5) << 5) | (gpio_get_level(D4) << 4) |
(gpio_get_level(D3) << 3) | (gpio_get_level(D2) << 2) |
(gpio_get_level(D1) << 1) | gpio_get_level(D0)) << 8; //byte 1
                    while(!gpio_get_level(Pin_PCLK));//doi PCLK len cao
                    while(gpio_get_level(Pin_PCLK));//doi PLCK xuong thap
                    //luu vao bo nho dem frame_buffer
                    Frame_Buffer[i] = ((gpio_get_level(D7) << 7) |
(gpio_get_level(D6) << 6) | (gpio_get_level(D5) << 5) |
(gpio_get_level(D4) << 4) | (gpio_get_level(D3) << 3) |
(gpio_get_level(D2) << 2) | (gpio_get_level(D1) << 1) |
gpio_get_level(D0)) | z; //byte 2
                    while(!gpio_get_level(Pin_PCLK));//doi PCLK len cao
                    i++;
                }
        }
}
```

Chương trình sẽ thu từng điểm ảnh và lưu vào frame_buffer dạng ma trận, mỗi phần tử của frame_buffer là một biến 16 bit tương đương một điểm ảnh.

2.4.2 Bắt ảnh đa mức xám lưu vào bộ nhớ đệm

Mặc định tín hiệu YUV422 do Camera OV7670 truyền đến vi điều khiển theo thứ tự YUYV, lấy trung bình là 2 byte/pixel.

Do đặc điểm tín hiệu chói Y trong định dạng YUV chính là ảnh đa mức xám, khi Camera truyền dữ liệu định dạng YUV, chỉ cần lấy tín hiệu Y là đã có được hình ảnh đa mức xám.

Chương trình bắt ảnh đa mức xám tương tự như chương trình bắt ảnh màu RGB565, chỉ khác ở chỗ trong quá trình lấy mẫu dữ liệu trong 2 xung clock PCLK, vi xử lý chỉ lấy một byte ở xung clock đầu tiên (thu tín hiệu Y).

```
void CaptureVGA(uint16_t Cot, uint16_t Hang)
{
    uint16_t i = 0, x, y;
    while(!gpio_get_level(Pin_VSYNC)); // tin hieu VSYN cao ket thuc 1
frame
    while(gpio_get_level(Pin_VSYNC)); // tin hieu VSYN thap bat dau 1
frame
    y = Hang; //so line
    while(y--)
        {
            x = Cot; //so pixel tren 1 line
            while(!gpio_get_level(Pin_HREF)); // tin hieu HREF cao bat
dau 1 line
            while(x--)
                {
                    while(gpio_get_level(Pin_PCLK));// doi PCLK xuong
thap
                    frame_buffer[i] = ((gpio_get_level(D7) << 7) |
(gpio_get_level(D6) << 6) | (gpio_get_level(D5) << 5) |
(gpio_get_level(D4) << 4) | (gpio_get_level(D3) << 3) |
(gpio_get_level(D2) << 2) | (gpio_get_level(D1) << 1) |
gpio_get_level(D0)); //byte 1
                    while(!gpio_get_level(Pin_PCLK));//doi PCLK len cao
                    while(gpio_get_level(Pin_PCLK));//doi PLCK xuong thap
                    while(!gpio_get_level(Pin_PCLK));//doi PCLK len cao
                    i++;
                }
        }
}
```

Ảnh đa mức xám được lưu vào frame_buffer dạng ma trận, mỗi phần tử của frame_buffer là một biến 8 bit tương đương một điểm ảnh.

Ảnh đa mức xám (8 bit/pixel) làm giảm kích thước bộ nhớ một nửa so với ảnh màu RGB565 (16 bit/pixel) trong cùng độ phân giải, phù hợp với vi xử lý có bộ nhớ đệm hạn chế.

2.5 Cấu hình hoạt động Camera OV7670

Hình 2.1 Lưu đồ cấu hình chế độ hoạt động của Camera OV7670

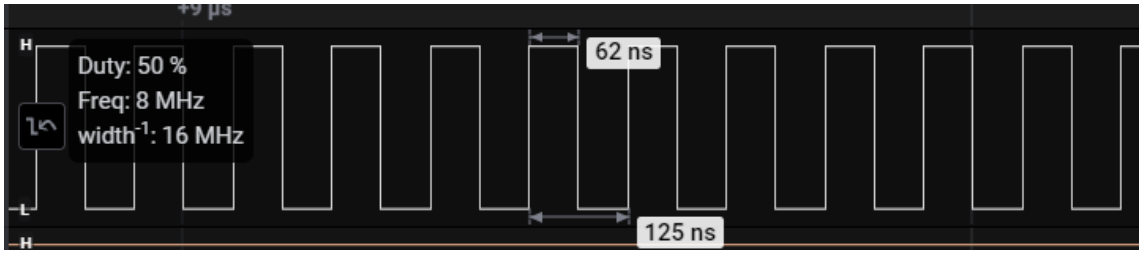
Hình 2.1 mô tả các bước cơ bản khi cấu hình chế độ hoạt động của Camera OV7670, trong quá trình thực nghiệm có thể thay đổi giá trị thanh ghi CLKRC (địa chỉ 0x11) để điều chỉnh tần số PCLK cho đến khi hệ thống bắt ảnh thành công.

Trong báo cáo, thực nghiệm cấu hình Camera OV7670 hoạt động với chế độ QQVGA, RGB565. (xem phụ lục A1)

CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM

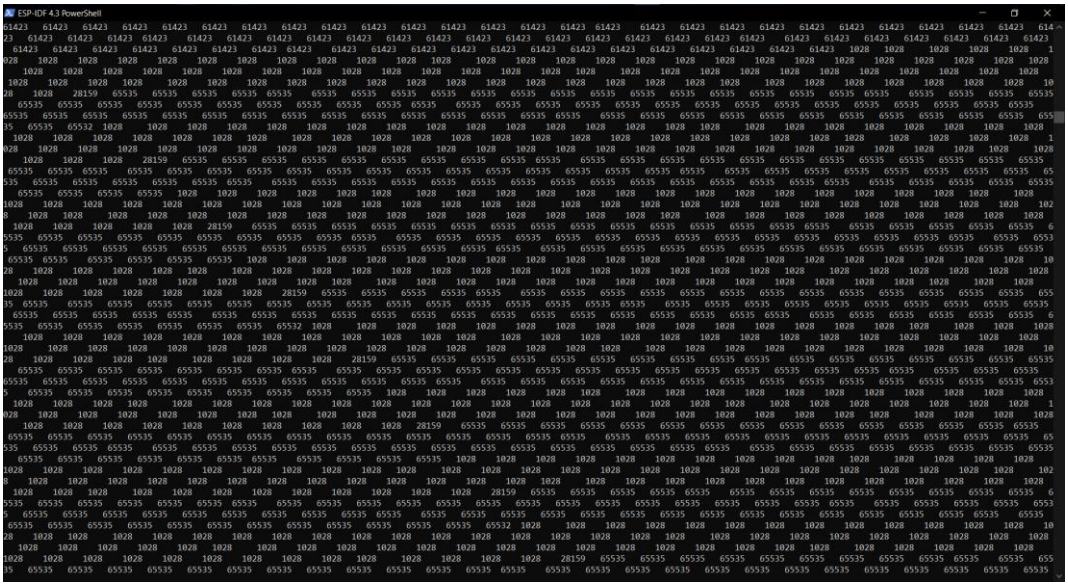
3.1 Đo tần số xung clock XCLK cấp cho Camera OV7670

Hình 3.1 cho thấy tần số xung clock đo tại PA8 đưa đến chân XCLK của OV7670 là 8MHz.



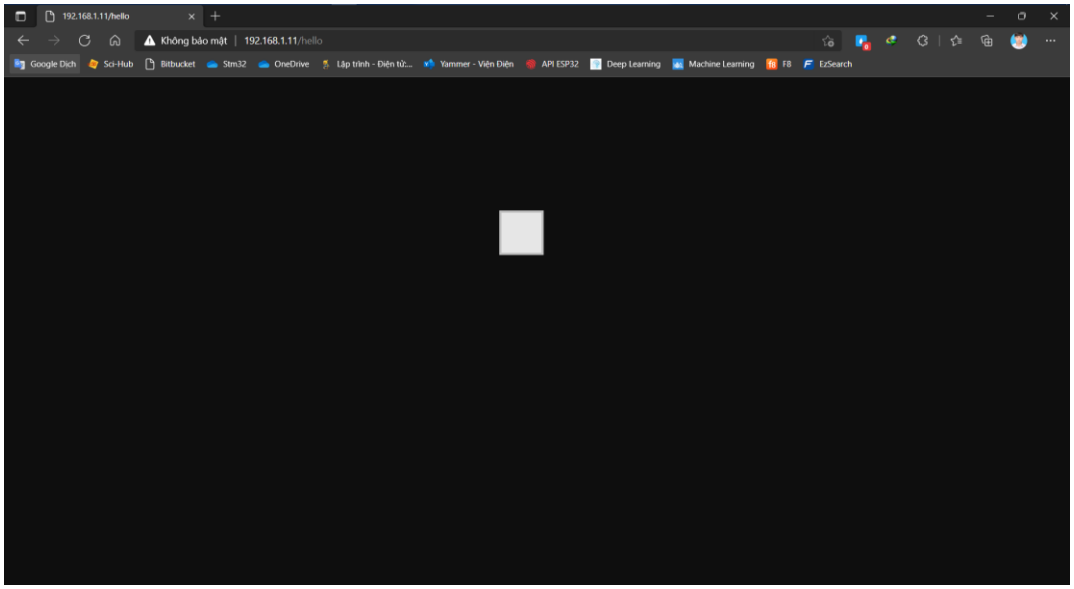
Hình 3.1 Xung Clock XCLK cấp cho Camera

3.2 Camera OV7670 hoạt động ở chế độ QQVGA, RGB565



Hình 3.2 Giá trị của từng pixel trong một khung hình

3.3 Gửi hình ảnh từ Camera OV7670 lên Server



Hình 3.3 Ảnh gửi từ Camera OV7670 lên Server

CHƯƠNG 4. KẾT LUẬN

4.1 Kết luận

Hệ thống đã giao tiếp được với Camera OV7670, đã đọc được giá trị trả về từ Camera nhưng không hiển thị được hình ảnh lên Server.

4.2 Phân công công việc trong nhóm

Họ và tên	MSSV	Nhiệm vụ	Đánh giá
Nguyễn Tiến Đạt	20173727	Code giao tiếp Camera OV7670, làm slide	50%
Trần Quang Huy	20173952	Code gửi dữ liệu lên Server, viết báo cáo	50%

TÀI LIỆU THAM KHẢO

[1] ESP32 API Reference, "espressif," [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html.

[2] ESP32 Technical Reference Manual, "espressif," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[3] OV7670 Datasheet.

[4] ESP32 Series Datasheet, "espressif," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

PHỤ LỤC

A1. Cấu hình chế độ QQVGA, RGB565

```
void OV7670_QQVGA_RGB565_Init(void)
{
    SCCB_write_reg(COM7, 0x80); //COM7, RESET camera
    vTaskDelay(500 / portTICK_PERIOD_MS);
    //Video format RGB565
    SCCB_write_reg(COM7, 0x04); // COM7, output format RGB
    SCCB_write_reg(COM15, 0xD0); // COM15, output format RGB565
    SCCB_write_reg(RGB444, 0x00); // disable RGB444
    SCCB_write_reg(COM1, 0x00); // COM1, disable CCIR656
    //Tan so dao dong noi camrea
    SCCB_write_reg(DBLV, 0x80); //nhan tan so voi 6
    SCCB_write_reg(CLKRC, 0x07); //chia tan so 8 => Fin = 6MHz
    // scalling windows
    SCCB_write_reg(COM3, 0x04); // COM3, DCW enable
    SCCB_write_reg(COM14, 0x1A); // COM14, chia tan so PCLK: chia 4
    SCCB_write_reg(SCALLING_XSC, 0x3A); // SCALLING_XSC
    SCCB_write_reg(SCALLING_YSC, 0x35); // SCALLING_YSC
    SCCB_write_reg(SCALLING_DCWCTR, 0x22); // downsample VGA by 4 ->
QQVGA
    SCCB_write_reg(SCALLING_PCLK_DIV, 0xF2); //
    SCCB_write_reg(MVFP, 0x0); // MVFP, Mirror/Flip hinh anh
    //SCCB_write_reg(COM10, 0x32); // COM10, tat tin hieu PCLK khi HREF
bank
    // Hardware window QQVGA
    SCCB_write_reg(TSLB, 0x04); // TSLB, cho phep thay doi tham so
hardware windows
    SCCB_write_reg(HSTART, 0x16); // REG_HSTART
    SCCB_write_reg(HSTOP, 0x04); // REG_HSTOP
    SCCB_write_reg(REG_HREF, 0x24); // REG_HREF
    SCCB_write_reg(VSTART, 0x02); // REG_VSTART
    SCCB_write_reg(VSTOP, 0x7A); // REG_VSTOP
    SCCB_write_reg(VREF, 0x0A); // REG_VREF
    // Matrix coefficients, saturation = 0,
    SCCB_write_reg(MTX1, 0x80); // MTX1
    SCCB_write_reg(MTX2, 0x80); // MTX2
    SCCB_write_reg(MTX3, 0x00); // MTX3
    SCCB_write_reg(MTX4, 0x22); // MTX4
    SCCB_write_reg(MTX5, 0x5E); // MTX5
    SCCB_write_reg(MTX6, 0x80); // MTX6
    SCCB_write_reg(MTXS, 0x9E ); // MTXS
    // Gamma
    SCCB_write_reg(SLOP, 0x20); // SLOP
    SCCB_write_reg(GAM1, 0x10); // GAM1
    SCCB_write_reg(GAM2, 0x1E); // GAM2
    SCCB_write_reg(GAM3, 0x35); // GAM3
    SCCB_write_reg(GAM4, 0x5A); // GAM4
    SCCB_write_reg(GAM5, 0x69); // GAM5
```

```
    SCCB_write_reg(GAM6, 0x76); // GAM6
    SCCB_write_reg(GAM7, 0x80); // GAM7
    SCCB_write_reg(GAM8, 0x88); // GAM8
    SCCB_write_reg(GAM9, 0x8F); // GAM9
    SCCB_write_reg(GAM10, 0x96); // GAM10
    SCCB_write_reg(GAM11, 0xA3); // GAM11
    SCCB_write_reg(GAM12, 0xAF); // GAM12
    SCCB_write_reg(GAM13, 0xC4); // GAM13
    SCCB_write_reg(GAM14, 0xD7); // GAM14
    SCCB_write_reg(GAM15, 0xE8); // GAM15
    //While balance, can bang trang
    SCCB_write_reg(COM8, 0xA7); // COM8, AWB on | FASTAEC | AECSTEP | AGC
on | AEC on
    SCCB_write_reg(AWBC1, 0x0A); // AWBC1
    SCCB_write_reg(AWBC2, 0xF0); // AWBC2
    SCCB_write_reg(AWBC3, 0x34); // AWBC3
    SCCB_write_reg(AWBC4, 0x58); // AWBC4
    SCCB_write_reg(AWBC5, 0x28); // AWBC5
    SCCB_write_reg(AWBC6, 0x3A); // AWBC6
    SCCB_write_reg(AWBC7, 0x88); // AWBC7
    SCCB_write_reg(AWBC8, 0x88); // AWBC8
    SCCB_write_reg(AWBC9, 0x44); // AWBC9
    SCCB_write_reg(AWBC10, 0x67); // AWBC10
    SCCB_write_reg(AWBC11, 0x49); // AWBC11
    SCCB_write_reg(AWBC12, 0x0E); // AWBC12
    SCCB_write_reg(AWBCTR3, 0x0A); // AWBCTR3
    SCCB_write_reg(AWBCTR2, 0x55); // AWBCTR2
    SCCB_write_reg(AWBCTR1, 0x11); // AWBCTR1
    SCCB_write_reg(AWBCTR0, 0x9F); // AWBCTR0, 9e -> advance AWB, 9f ->
simple AWB
    SCCB_write_reg(0x6a, 0x40); // G chanel gain
    SCCB_write_reg(0x01, 0x40); // B chanel gain
    SCCB_write_reg(0x02, 0x40); // R chanel gain
    SCCB_write_reg(0x14, 0x68); // REG_COM9
    // Bright, do sang hinh anh
    SCCB_write_reg(0x55, 0x00); // Bright =0
    //SCCB_write_reg( 0x55, 0x18 ); // Bright =1
    //SCCB_write_reg( 0x55, 0x30 ); // Bright =2
    //SCCB_write_reg( 0x55, 0x98 ); // Bright =-1
    //SCCB_write_reg( 0x55, 0xb0 ); // Bright =-2
    // Contrast, do tuong phan hinh anh
    SCCB_write_reg(0x56, 0x40); // contrast =0
    //SCCB_write_reg( 0x56, 0x50 ); // contrast =1
    //SCCB_write_reg( 0x56, 0x60 ); // contrast =2
    //SCCB_write_reg( 0x56, 0x38 ); // contrast =-1
    //SCCB_write_reg( 0x56, 0x40 ); // contrast =-2
    // Hieu ung: normal
    SCCB_write_reg(0x67, 0xc0);
    SCCB_write_reg(0x68, 0x80);
    // Banding filter 50Hz
    SCCB_write_reg(0x3B, 0x0A); // COM11, chon bo loc 50Hz
```

```
    SCCB_write_reg(0x9d, 0x4c); // BD50ST, 50Hz banding filter value,
active khi COM8[5] high, COM11[3] high
    SCCB_write_reg(0xA5, 0x05); // BD50MAX, max banding filter step
    SCCB_write_reg(0x0E, 0x61); // COM5
    SCCB_write_reg(0x0F, 0x4b); // COM6
    SCCB_write_reg(0x16, 0x02); //
    SCCB_write_reg(0x21, 0x02); // ADCCTR1
    SCCB_write_reg(0x22, 0x91); // ADCCTR2
    SCCB_write_reg(0x29, 0x07); // RSVD
    SCCB_write_reg(0x33, 0x0b); // CHLF
    SCCB_write_reg(0x35, 0x0b); //
    SCCB_write_reg(0x37, 0x1d); // ADC control
    SCCB_write_reg(0x38, 0x71); // ADC and Analog Common mode control
    SCCB_write_reg(0x39, 0x2a); // ADC Offset control
    SCCB_write_reg(0x4d, 0x40); // DM Pos, dummy row position
    SCCB_write_reg(0x4e, 0x20); //
    SCCB_write_reg(0x8d, 0x4f); //
    SCCB_write_reg(0x8e, 0x0); //
    SCCB_write_reg(0x8f, 0x0); //
    SCCB_write_reg(0x90, 0x0); //
    SCCB_write_reg(0x91, 0x0); //
    SCCB_write_reg(0x96, 0x0); //
    SCCB_write_reg(0x9a, 0x0); //
    SCCB_write_reg(0xb0, 0x84); // No document
    SCCB_write_reg(0xb1, 0x0c); // ABLC1, 1100, enable ABLC function
    SCCB_write_reg(0xb2, 0x0e); //
    SCCB_write_reg(0xb3, 0x82); // ABLC target
    SCCB_write_reg(0xb8, 0x0a); //
    SCCB_write_reg(0x3F, 0x0); // REG_EDGE Enhancement Adjustment
    SCCB_write_reg(0x74, 0x10); // REG74 0001 0000, digital gain manual
control bypass.
    SCCB_write_reg(0x75, 0x05); // REG75, Edge enhancement lower limit
    SCCB_write_reg(0x76, 0xe1); // REG76, 1110 0001, [6:5]enable
black/white pixel correct, [4:0]Edge enhancement higher limit
    SCCB_write_reg(0x77, 0x01); // REG77, de-noise range control
    SCCB_write_reg(0x4c, 0x0); // De-noise strength
    SCCB_write_reg(0x4b, 0x09); // UV average enable
    SCCB_write_reg(0xc9, 0x60); // Saturation control, bao hoa
    SCCB_write_reg(0x34, 0x11); // Array reference control
}
```