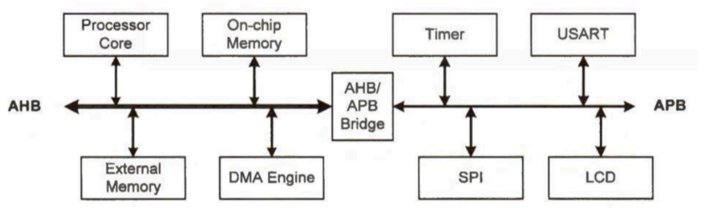
DMA (Direct Memory Access)

- hardware module
    - data source/destination include Memory (RAM/ROM) or Memory-mapped I/O (peripherals)
- improves energy efficiency of system
    - DMA simpler than CPU, less gates
    - fewer gates = less power
- on STM32, DMA connected to AHB bus and supports **2 modes**
    - *Normal* copy from start address to end address and stop when done
    - *Circular* continuously copy from start to end address in a loop

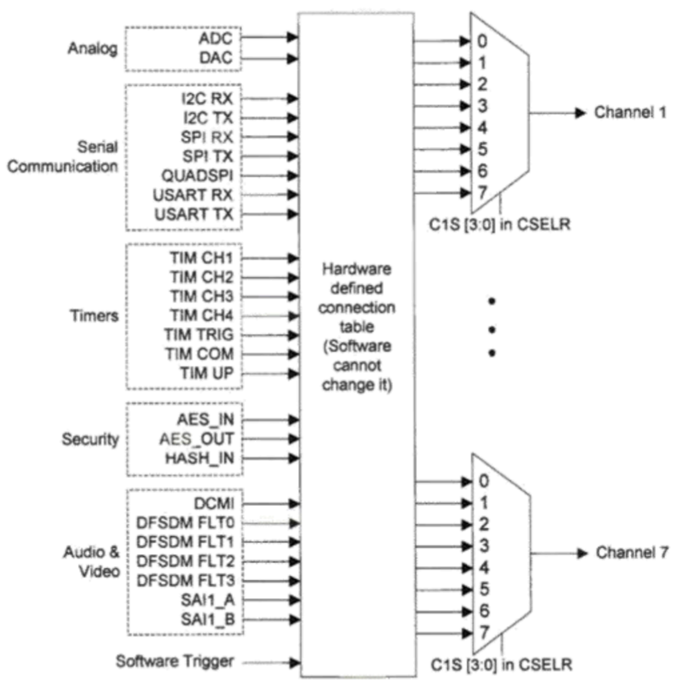


DMA Triggers

- DMA performs a transfer based on a *trigger*
    - when peripheral → memory, trigger could be new input
        - ADC has new reading
        - DMA waits for the new reading to trigger before copying to memory
    - when copying from *memory to memory*, CPU = trigger
        - copy image N+1 when CPU is working on image N
        - CPU triggers DMA to copy the next image to RAM **USE IN PROJECT**
        - calls a **software trigger**

DMA Channels

- **DMA controllers** offer multiple channels
    - each channel
        - transfers data independently
        - has a source & destination
        - amount of data to be transferred
        - data-width of transfers (how many bits are moved each time)
- one trigger is available per channel
- when using multiple channels, channel 1 is highest priority

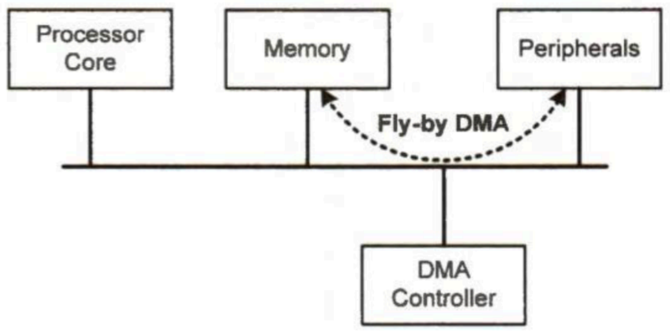


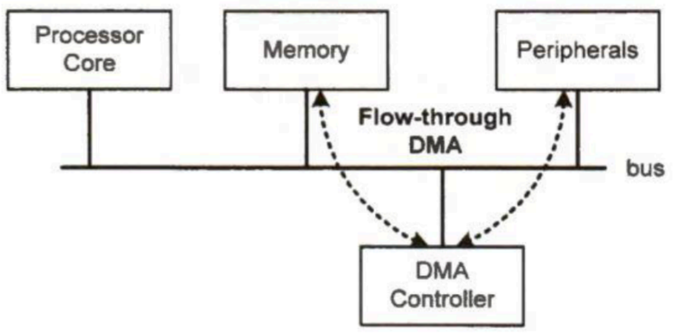
DMA Interrupts

- DMA controllers supports interrupts to update the CPU
- transfer interrupts when
  - *transfer complete* fires when total amount of data has been transferred
  - *transfer error* fires when an error occurs during the transfer
  - *half transfer* fires when half of the data has been transferred
    - useful when the CPU can start working on partial data while DMA copies the rest

DMA Transfer Types

- Fly-by DMA
  - data goes directly form source to destination
  - supported on STM32 CPU M4
- flow-through
  - uses register in the controller
    - data is read one at a time from source to register
    - data is written from register into destination
  - used when
    - devices have register of different sizes
    - when memory to memory transfer cannot be read/written in one cycle





DMA Configuration

- *which DMA controller* should be used?
- *which channel* should be used?
    - different channels have access to different resources (ADC, USART, I2C etc.)
- *which trigger* should be selected for that channel?

Priorities

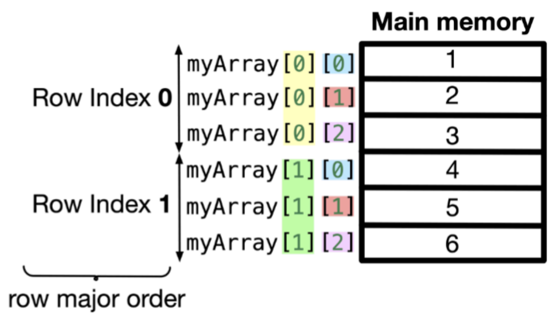
- hardware priority is higher; channel 1 highest
- If using software priority, highest should be channel demanding highest BW

DCMI (Digital Camera Interface)

- OV7670 supports image sizes
    - VGA: 640 × 480
    - CIF: 352 × 288
    - Custom down to 40×30
- Uses I2C compatible protocol (SCCB)
- 30fps
- use DMA to copy the image to memory without CPU

Images

- each image frame is rows, row major order
    - 8 bit greyscale image - 0 is black, 0xFF is white



RGB

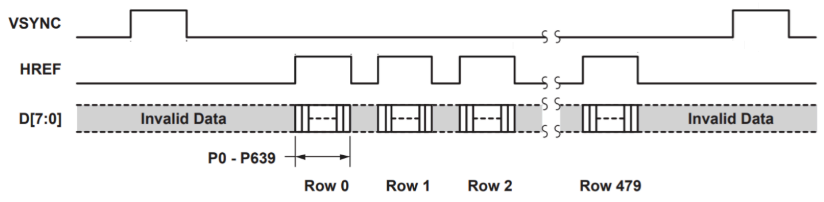
- color-space
- OV7670 supports RGB444, RGB555 and RGB565
  - 444 is 4 bits red, 4 bits green, 4 bits blue

YCbCr

- Y = avg of R, G, B
- Cb = blue difference, Cr = green difference
- Just the Y channel gives greyscale image
- OV7670 uses 8 bits for Y, Cb, Cr
  - to save BW, skips Cb and Cr for odd-numbered pixels
  - If Y_i, Cb_i and Cr_i are the components for pixel i, the bytes sent from the camera are

$$Cb_0, Y_0, Cr_0, \underline{Y_1}, Cb_2, Y_2, Cr_2, \underline{Y_3}, Cb_4, Y_4, Cr_4, \underline{Y_5}\ldots$$

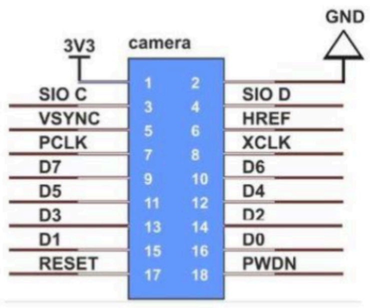
Timing

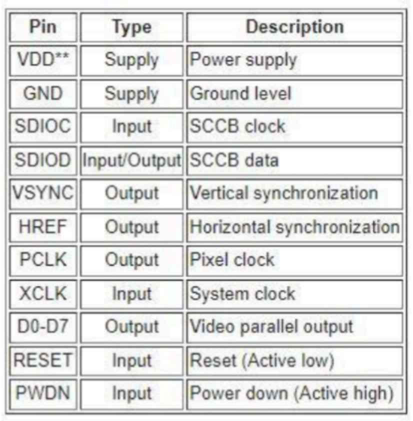


- VSYNC - pulse to pulse; indicates 1 whole frame
- HREF - high when outputting each row
- $D[7:0]$ outputs each pixel of the row, byte by byte
- each pixel output is aligned to PCLK

OV7670 - I/O

- SCCB (Serial Camera Control Bus) Interface - based on I2C
- configure camera by writing to registers using I2C writes
- OV7670 has no crystal oscillator, so cannot produce a clock
  - need to provide clock (XCLK) for the camera, camera creates PCLK from XCLK (need it for fps)





Pin	Type	Description
VDD**	Supply	Power supply
GND	Supply	Ground level
SDIOC	Input	SCCB clock
SDIOD	Input/Output	SCCB data
VSYNC	Output	Vertical synchronization
HREF	Output	Horizontal synchronization
PCLK	Output	Pixel clock
XCLK	Input	System clock
D0-D7	Output	Video parallel output
RESET	Input	Reset (Active low)
PWDN	Input	Power down (Active high)

DCMI

- STM proviudes DCMI module which handles all the signalling and timing
- connect DCMI to DMA, CPU isn't involved in transfer