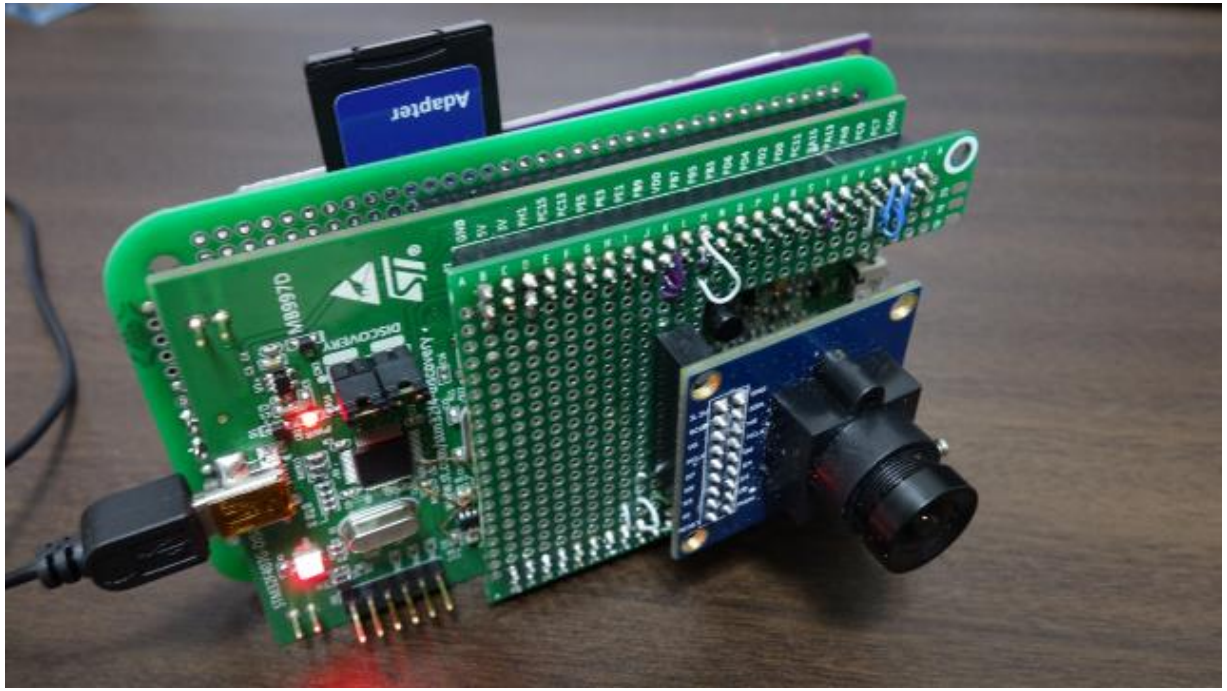
How to Create Digital Camera with STM32F4 Discovery Board

Takeshi. I



Outline

- Overview
- Device Control and Hardware Connection
- Dataflow
- Software Architecture
- Software Design
- Appendix
  - Hardware Configurations
  - Port Map

Overview

Policy

- This is just a hobby project
  - Cortex-M (STM32) is not the best choice for digital cameras
  - Better to use more powerful SoC such as Cortex-A (e.g. Raspberry Pi)

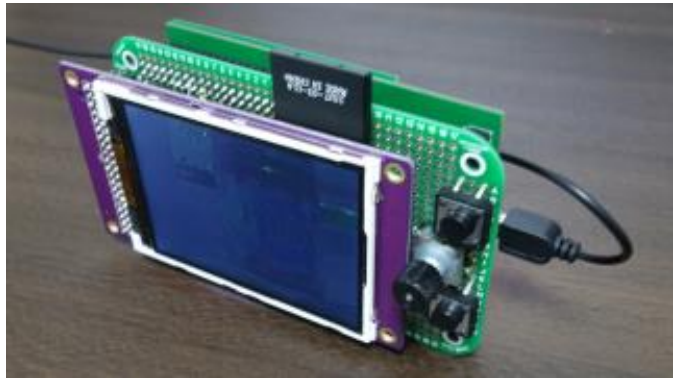
- Little effort on hardware work
  - Use discovery board

- Do not pursue performance
  - Many people already worked on this

Code and Documents

- https://github.com/take-iwiw/DigitalCamera_STM32

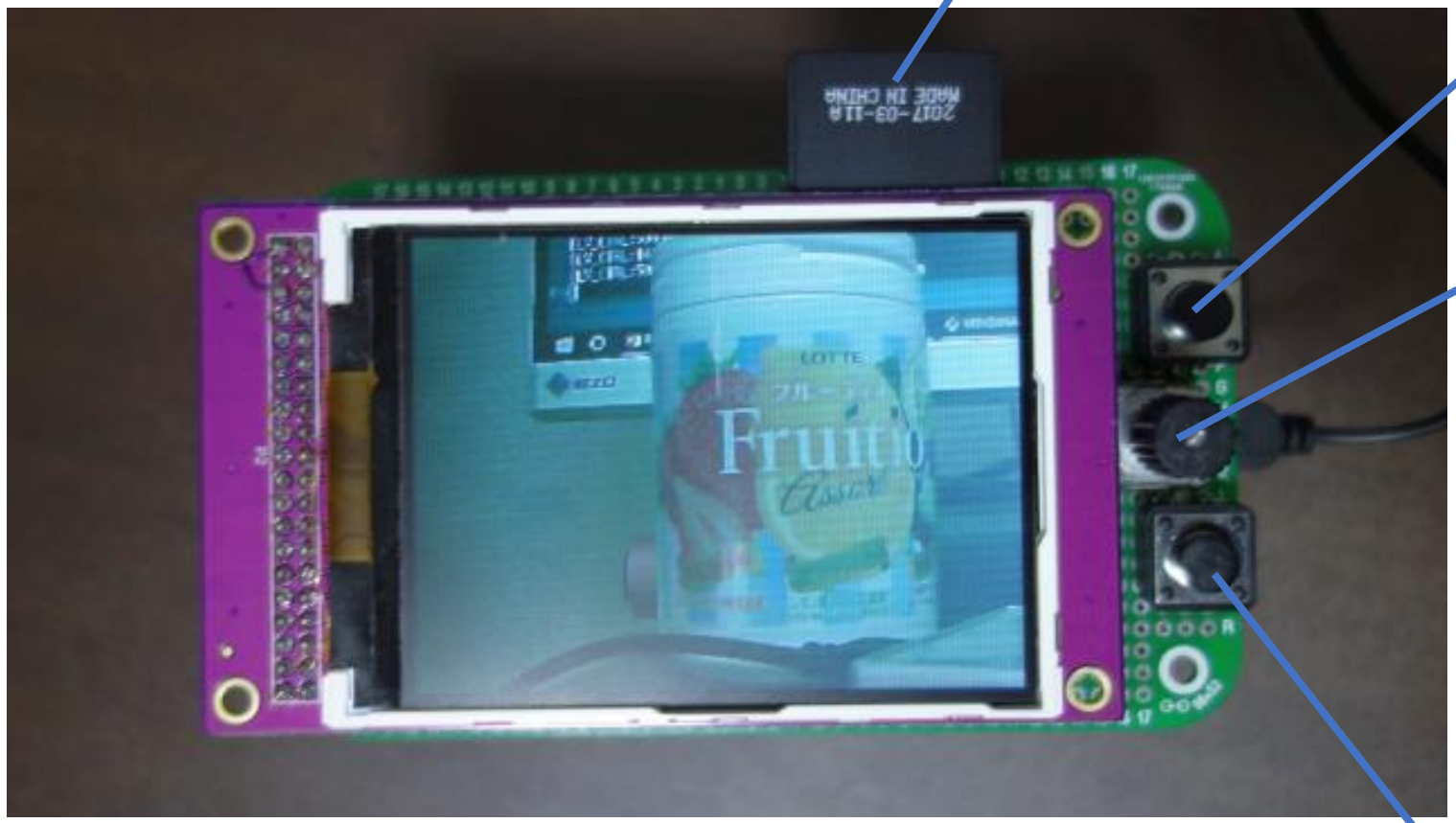
Photos



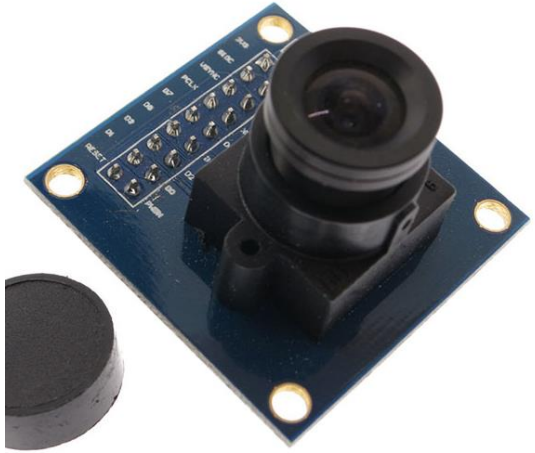
Videos

- https://www.youtube.com/watch?v=CgX3bM4v_aU

Control


SD Card
Capture button
Button
Movie Record @ Liveview mode
Movie Play/Pause @ Playback mode
Dial
JPEG Quality @ Liveview mode
Next content @ Playback mode
Mode change button

Key Components



- STM32F4 Discovery Board
  - STM32F407VGT (Cortex-M4)
  - 1-Mbyte Flash memory
  - 192-Kbyte RAM
- Display module
  - 3.2 inch LCD
  - ILI9341 controller
  - 16-bit parallel I/F
  - SD Card socket
- Camera module
  - OV7670
  - Without FIFO

Specifications

- **Still photo capture**
  - JPEG (*.jpg)
  - QVGA (320x240)

- **Movie record**
  - Motion JPEG (*.avi)
  - QVGA (320x240)
  - Around 5 fps

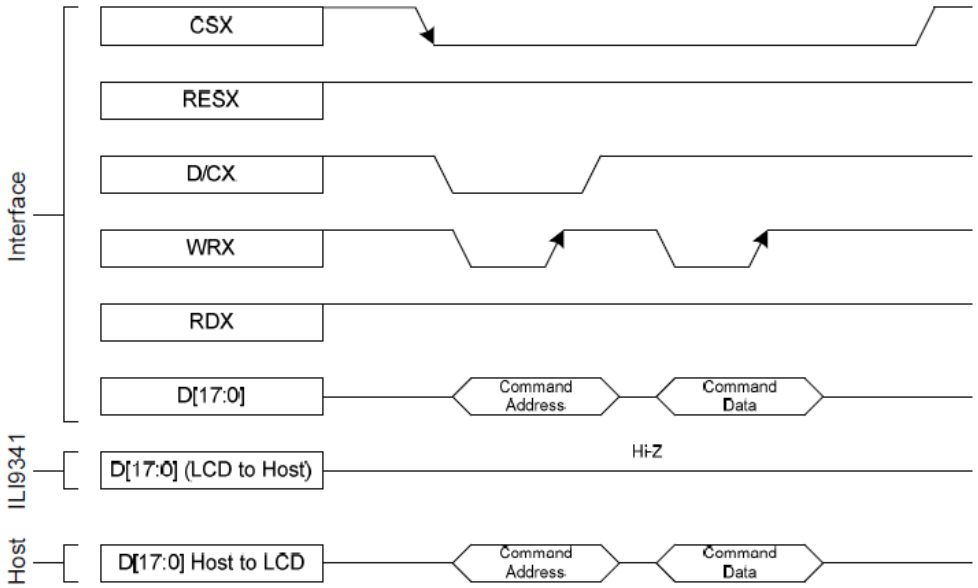
- **Playback**
  - JPEG (up to 2560x1920)
  - RGB565 (320x240)
  - Motion JPEG (around 10 fps)

- Media
  - SD Card (FAT32 format only?, 8GB ~ 16GB of SD card works well)

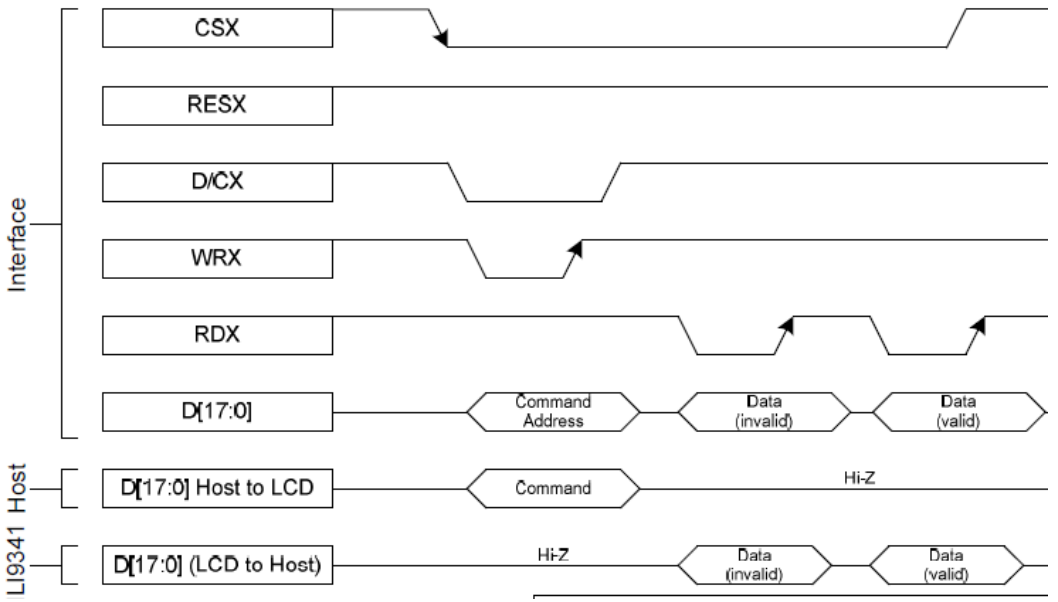
Device Control and Hardware Connection

Display module (1) – Access sequence

Write sequence of ILI9341



Read sequence of ILI9341

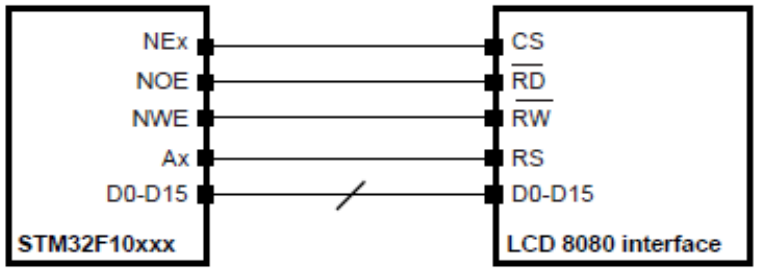


*ILI9341 datasheet

- Summary of ILI9341access sequence
  - CS (Chip Select) is low during access
  - Write access: latch at WRX rising
  - Read access: latch at RDX rising
  - Command access: DCX is low
  - Data access: DCX is high

- **This is the same access sequence as SRAM**
  - **Regard DCX as address bit, then**
  - **Access LCD module as 1-bit SRAM**

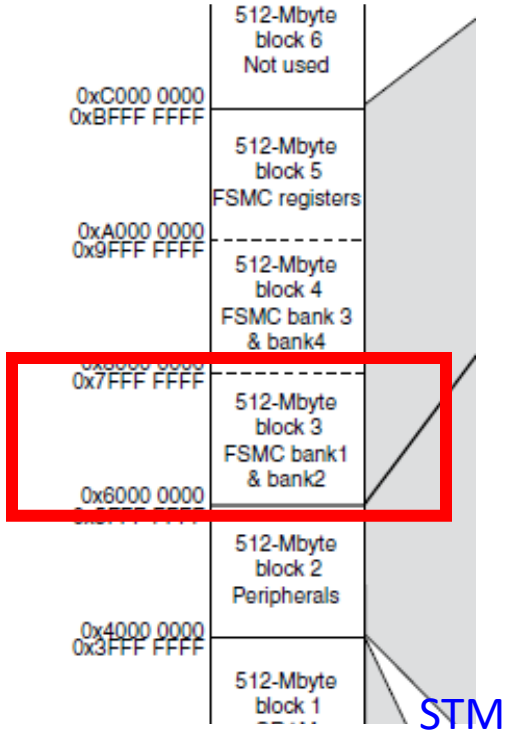
Display module (2) – FSMC function on STM32



RS = DCX

*AN2790 Application note

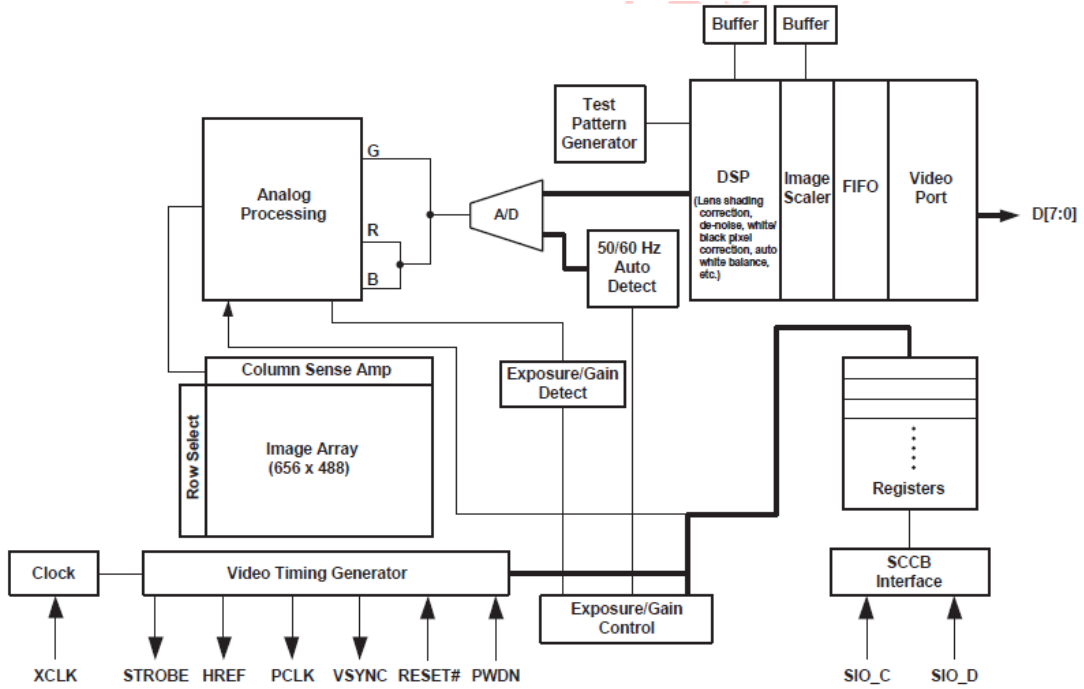
- STM32 has a function called FSMC (flexible static memory controller), which can be used to control LCD as well



STM32F407 memory map

- FSMC is assigned from 0x60000000 on STM32
- LCD access via FSMC is the same as accessing
  - * (volatile uint16_t*)0x6000_0000 for command
  - * (volatile uint16_t*)0x6002_0000 for data
- The following is the config for this
  - Memory Data Width = 16bit
  - RS(DCX) is connected to A16
  - CS is connected to NE1 (use FSMC bank1)
- Note (why 0x60020000???)
  - In 16-bit mode, data address issued to the memory is HADDR[25:1] >> 1
  - Therefore, in order to set A16 pin to high, CPU needs to access 0x6002_0000 (17-th bit is high)

Camera module- OV7670 Interface



*OV7670 datasheet

- Control I/F
  - SCCB (SIO_C, SIO_D)
- Clock supply
  - Supply around 24MHz clock (XCLK)
- Sync
  - OV7670 outputs PCLK, HREF and VSYNC
- Pixel data
  - OV7670 outputs 8bit data (D[7:0]

Use I2C (SCCB is similar to I2C, but NAK)
(Note: need a trick in implementation)

Use MCO (microcontroller clock output) with appropriate pre-scaler

Use DCMI (Digital Camera Interface)

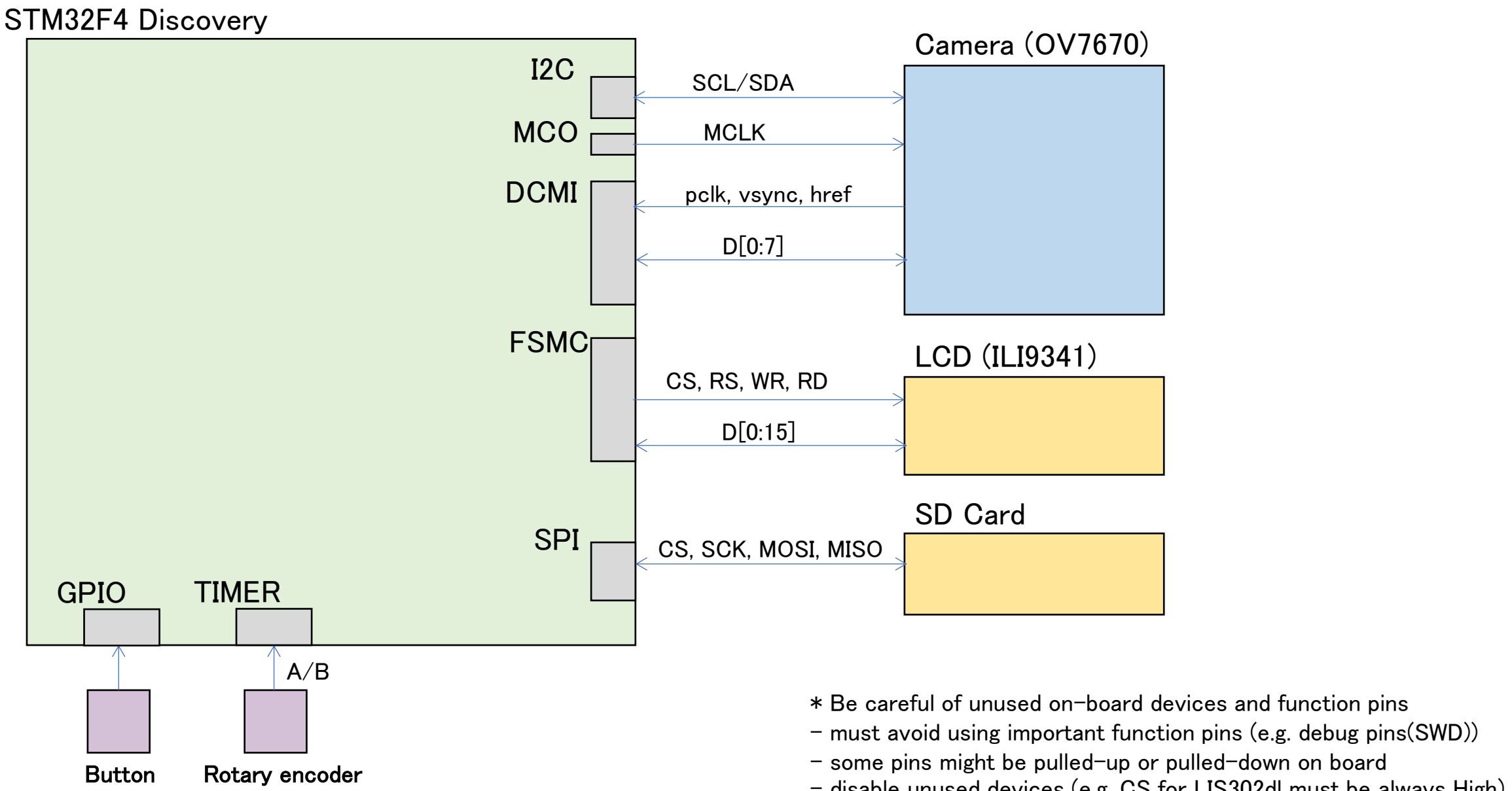
Other devices

- SD Card
  - Use SPI
  - (wanted to use SDIO, but some pins are shared with DCMI)

- Buttons
  - Use GPIO

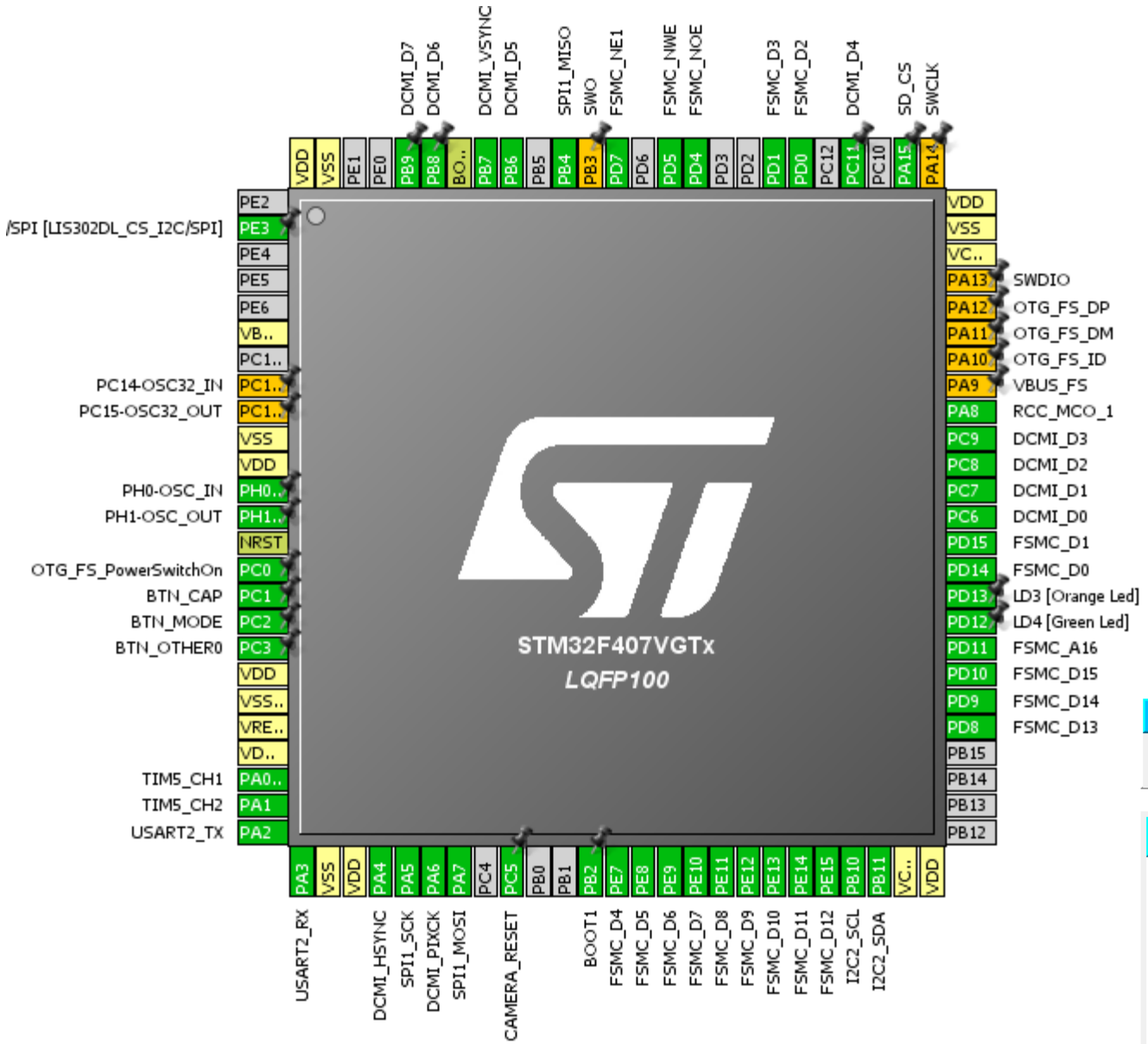
- Dial (Rotary Encoder)
  - Use TIMER as an external incremental encoder

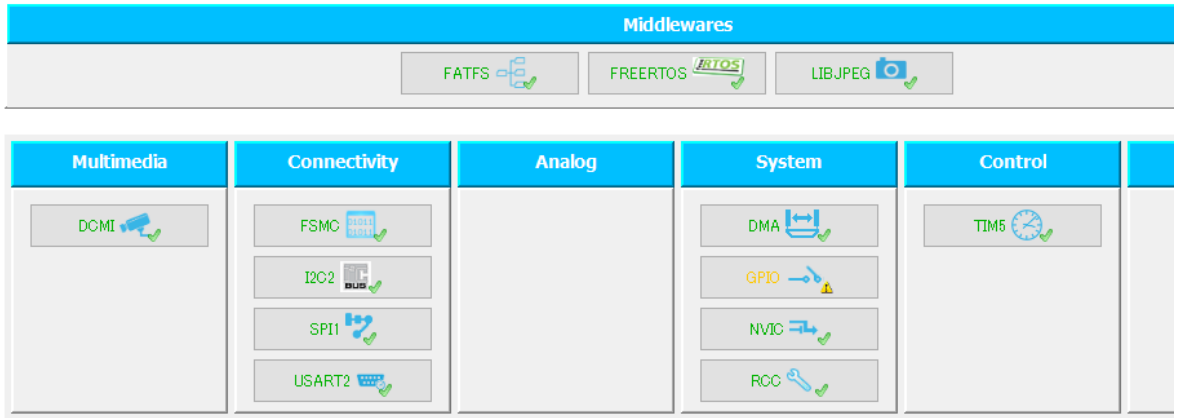
Hardware Connection



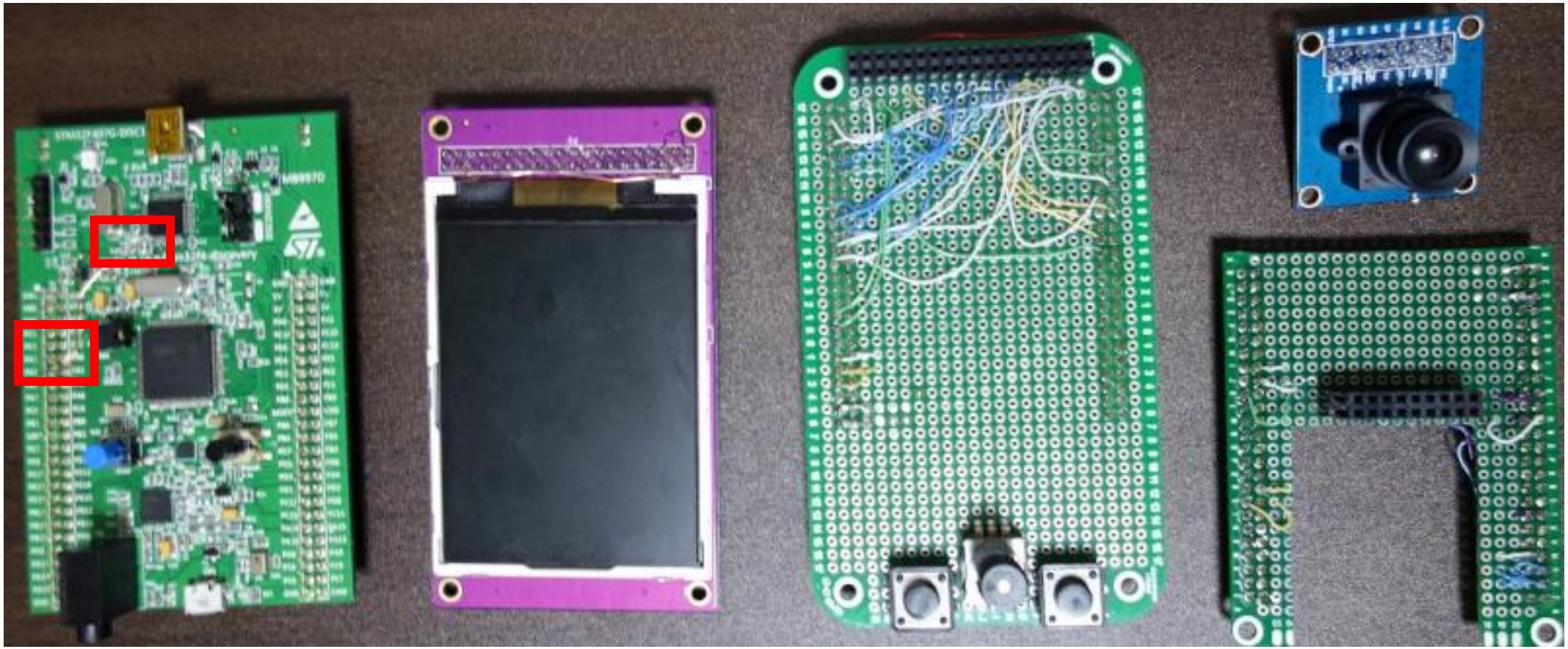
* Be careful of unused on-board devices and function pins
  - must avoid using important function pins (e.g. debug pins(SWD))
  - some pins might be pulled-up or pulled-down on board
  - disable unused devices (e.g. CS for LIS302dl must be always High)

Port map and used-modules in STM32





Breakout boards



breakout board for LCD
(connected to the bottom side of Discovery board)

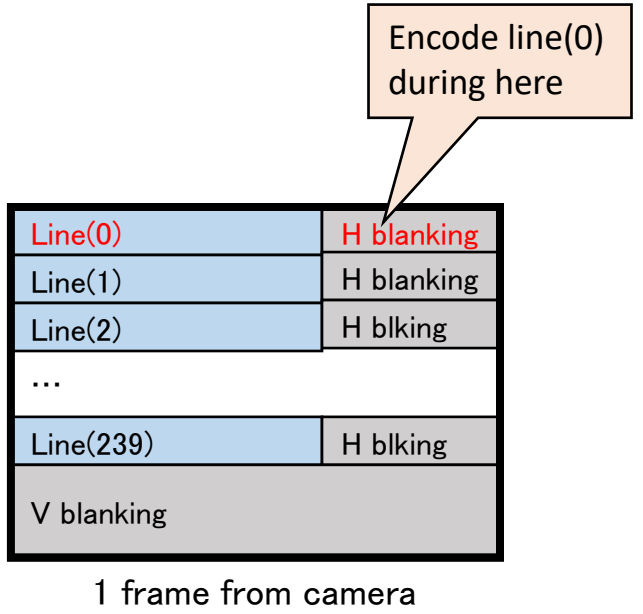
breakout board for Camera
(connected to the front side of Discovery board)

Note1: Wire USART2 on STM32
Note2: Use low profile socket for camera breakout board

Dataflow

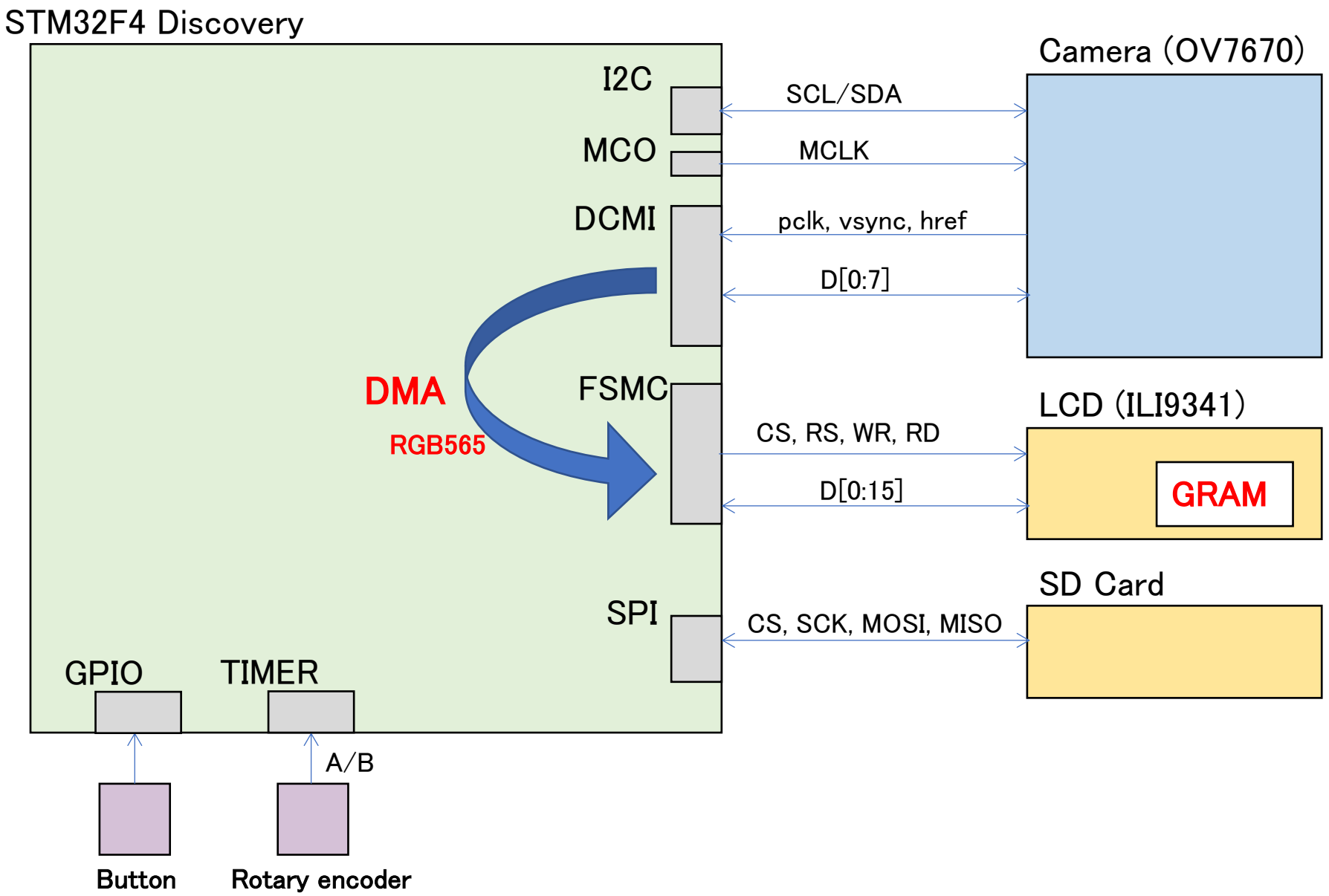
Main ideas for dataflow

- Liveview
  - Both ILI9341 and OV7670 support RGB565 format
  - So, directly transfer pixel data from OV7670 to ILI9341 using DMA
- Still photo capture
  - [Problem]
    - Not enough space for frame buffer memory (320x240x2)
  - [Solution A]
    - Increase H blanking time, then encode jpeg line by line
    - See the picture on the right
    - -> rejected because of too complex control
  - [Solution B]
    - Use RAM in ILI9341 as frame buffer
    - See the picture on the following page
    - -> adopt this idea
- Movie record
  - Repeat still photo capture every frame
  - The key point is to start/stop capture from camera frame by frame (do not use continuous capture), so that pixel data in Display is not corrupt during encoding

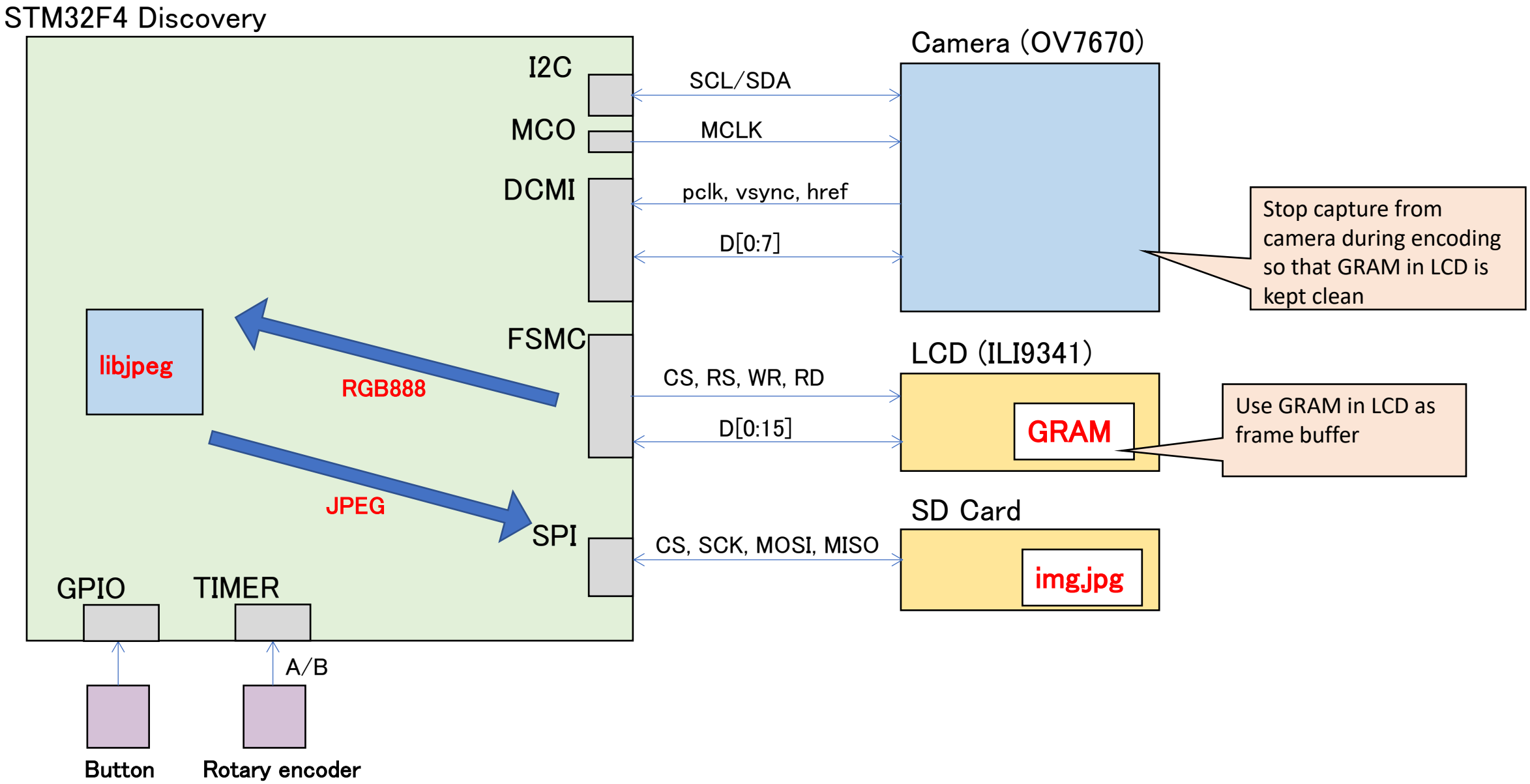


1 frame from camera

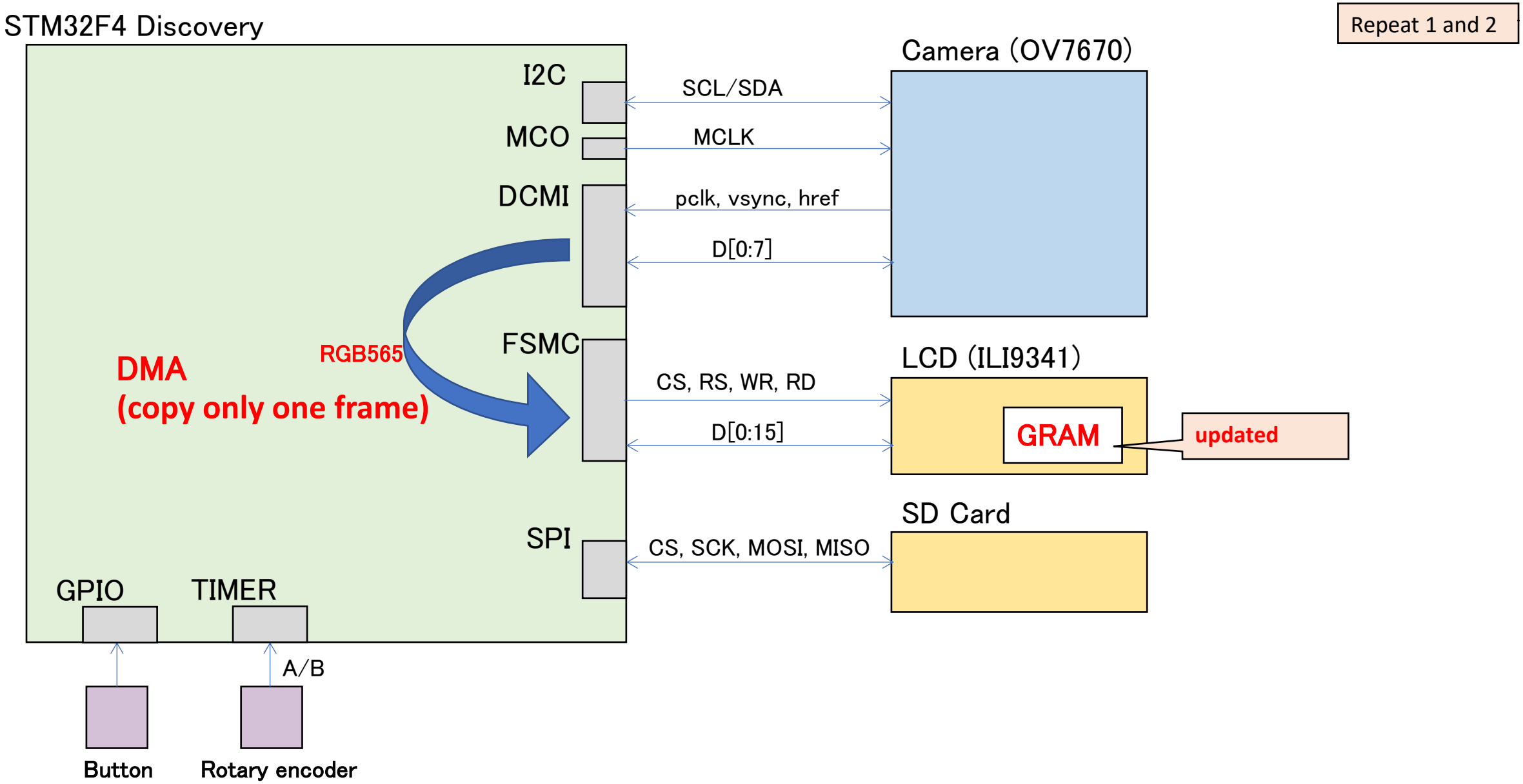
Dataflow – Liveview



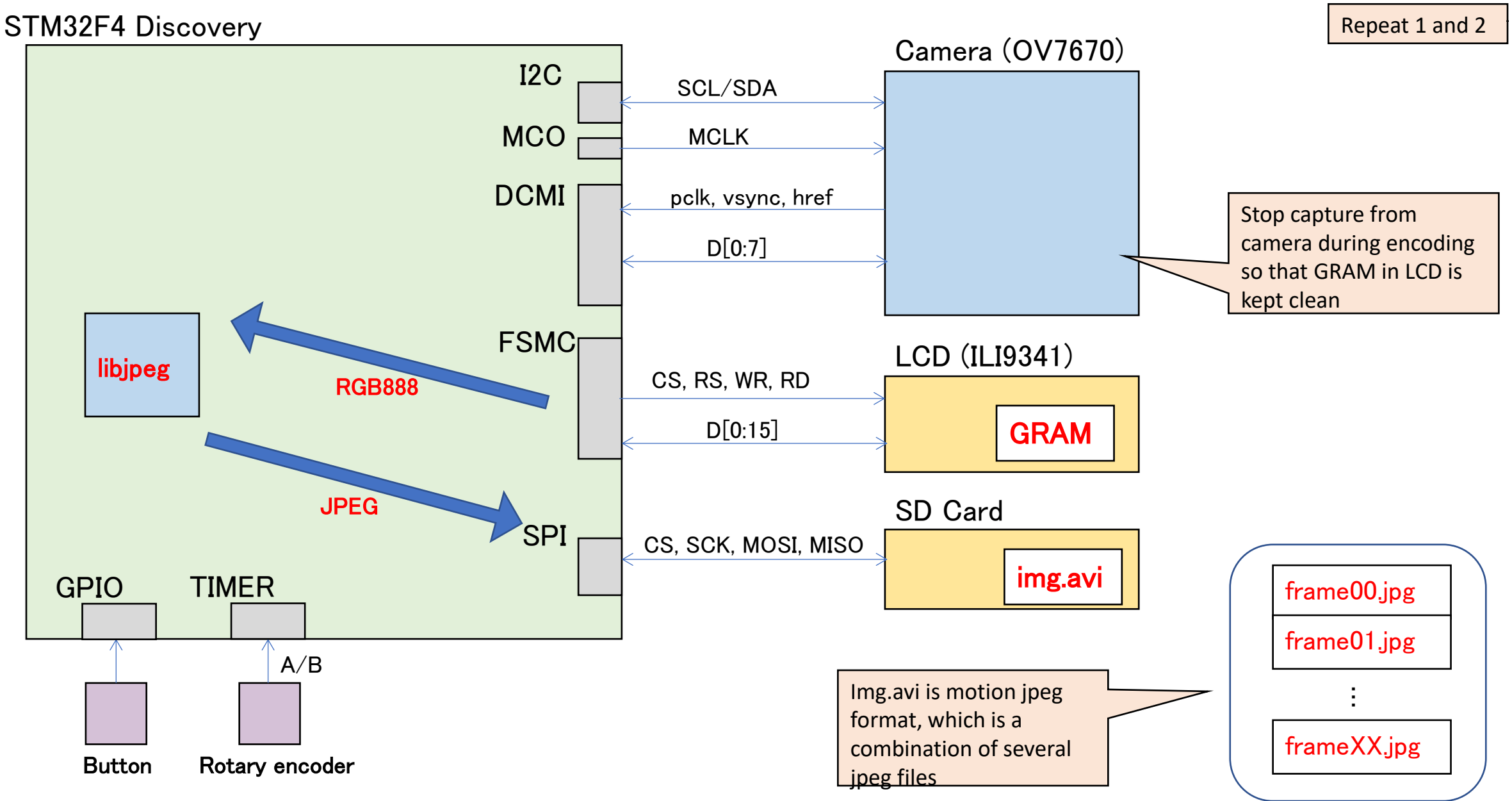
Dataflow – Still photo capture (JPEG)



Dataflow – Movie recording (Motion JPEG) – 1



Dataflow – Movie recording (Motion JPEG) – 2



Software Architecture

Software Development Environment

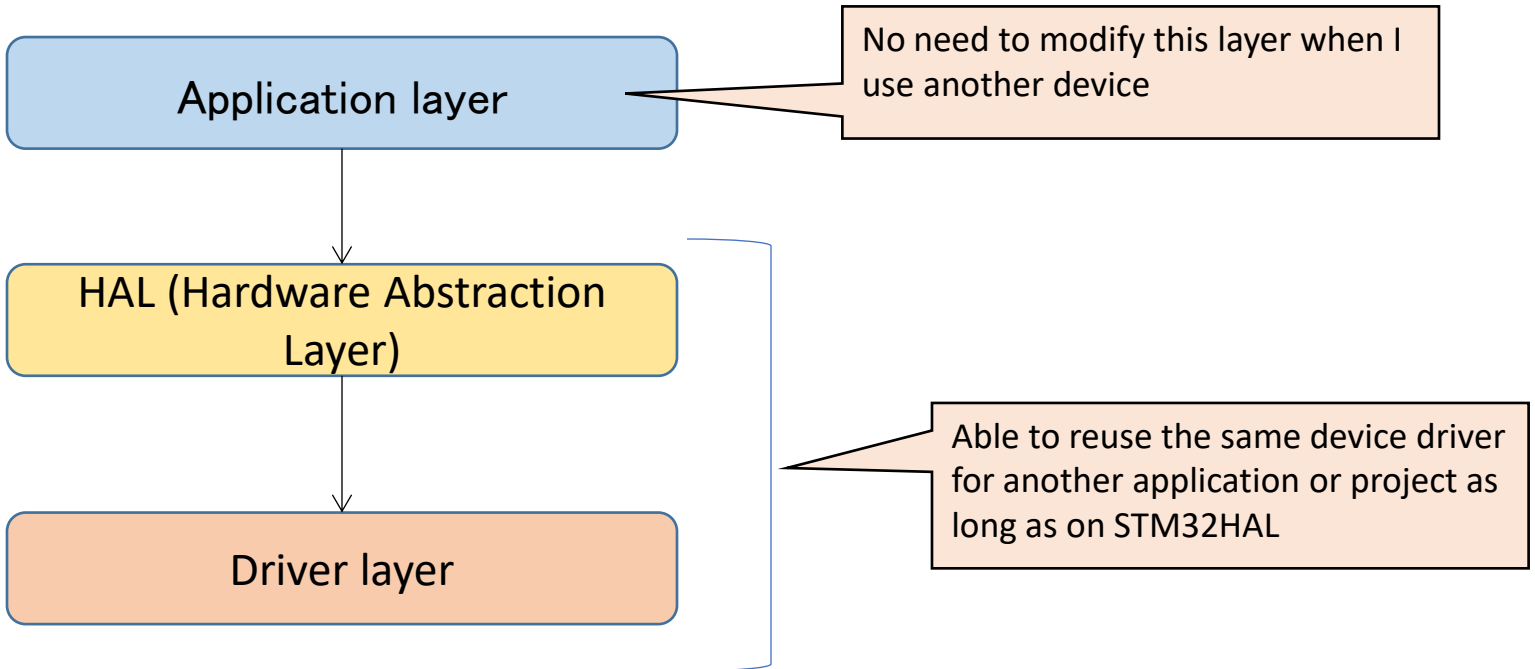
- Software Development Environment
  - IDE: sw4STM32
  - Tools: HAL with STM32 CubeMX

- Libraries (Middlewares)
  - FreeRTOS
  - FatFS
  - LibJPEG

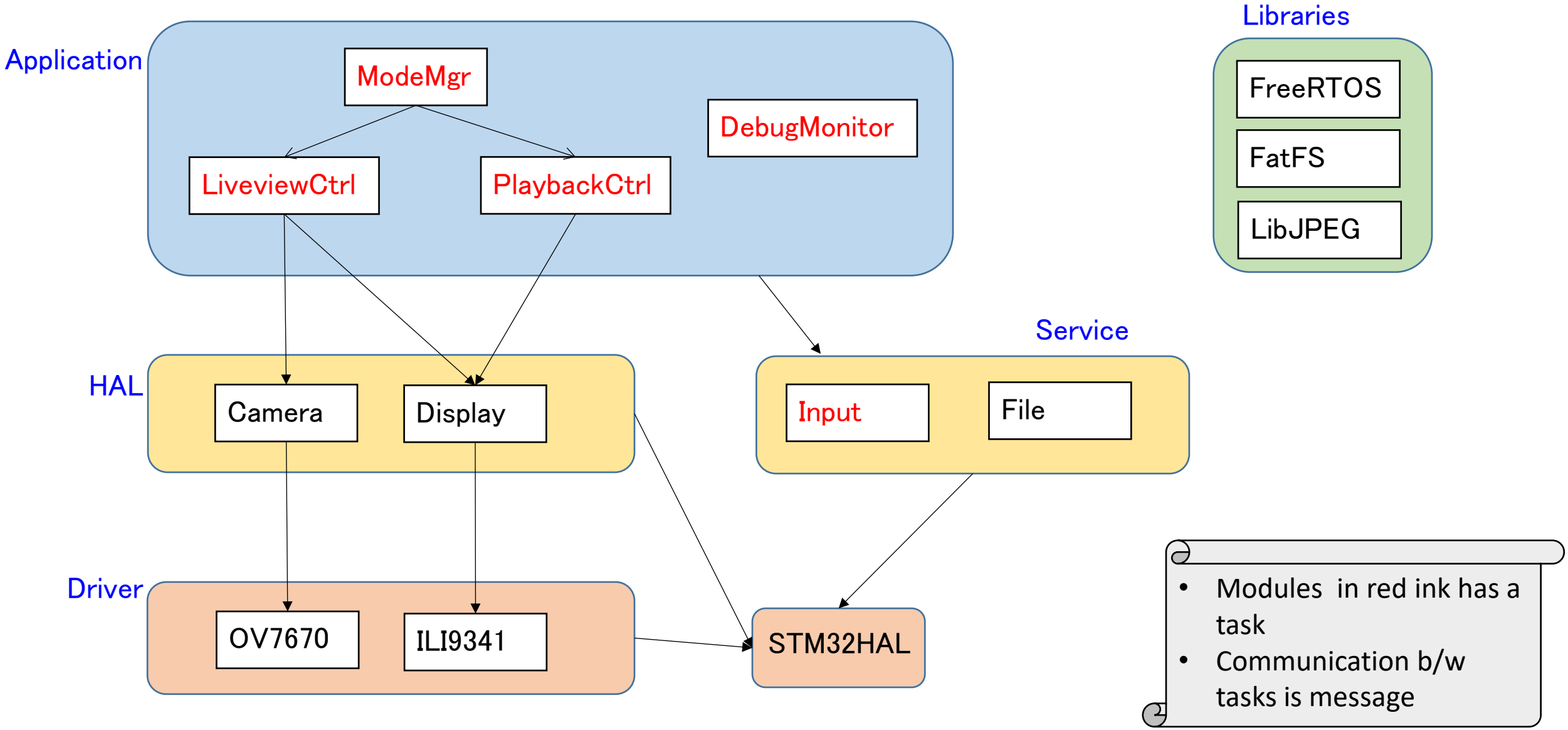
- Porting and modification needed (not described in this document)
  - FatFS porting to use SPI
  - LibJPEG modification to use appropriate buffer size
  - printf (putc, getc) porting to use UART

Software Architecture (1) – Policy

- Software architecture policy
  - Try to abstract device access
    - Create my own HAL layer (Display, Camera), which is different from “HAL” from STM32 (call this Stm32HAL)
      - STM32HAL abstracts hardware access on SoC such as GPIO
      - “My” HAL abstracts external device access such as display (more like BSP)
    - Application layer never access driver layer nor Stm32HAL directly



Software Architecture (2) – Overview



Software Architecture (3) – Responsibility

- Application
  - ModeMgr: controls mode (Liveview and Playback), runs sequence during mode change
  - LiveviewCtrl, PlaybackCtrl: main process in each mode, including enc/dec process
  - DebugMonitor: offers test command on terminal (UART). Able to access any module exceptionally

- HAL
  - Display: abstracts ILI9341 driver
  - Camera: abstracts OV7670 driver

- Service
  - Input: notifies key/dial device status to application (like “C” in MVC)
  - File: easy access to filesystem(FatFS)

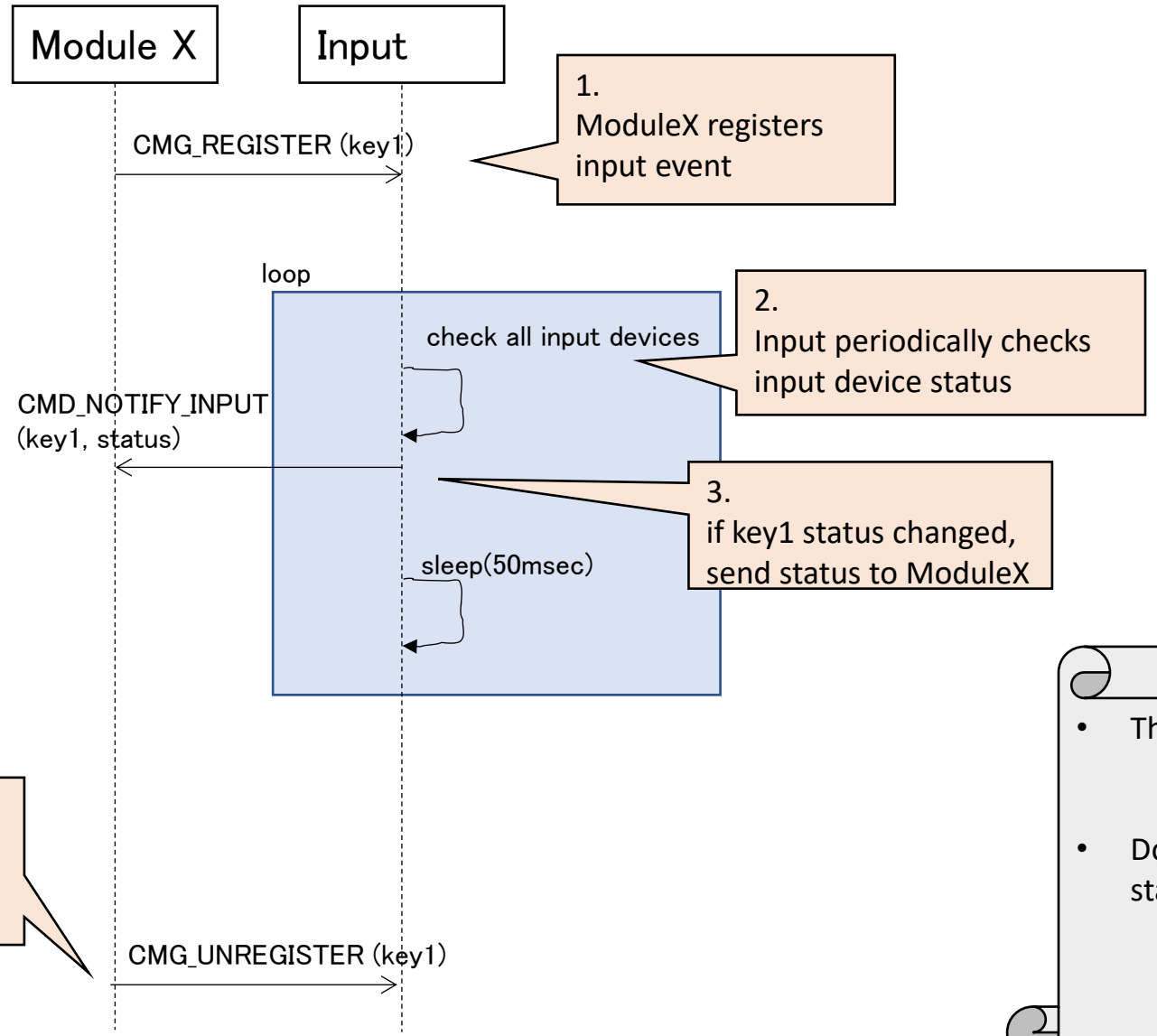
- Driver
  - ILI9341: is a device driver for ILI9341
  - OV7670: is a device driver for OV7670

Software Design

APIs

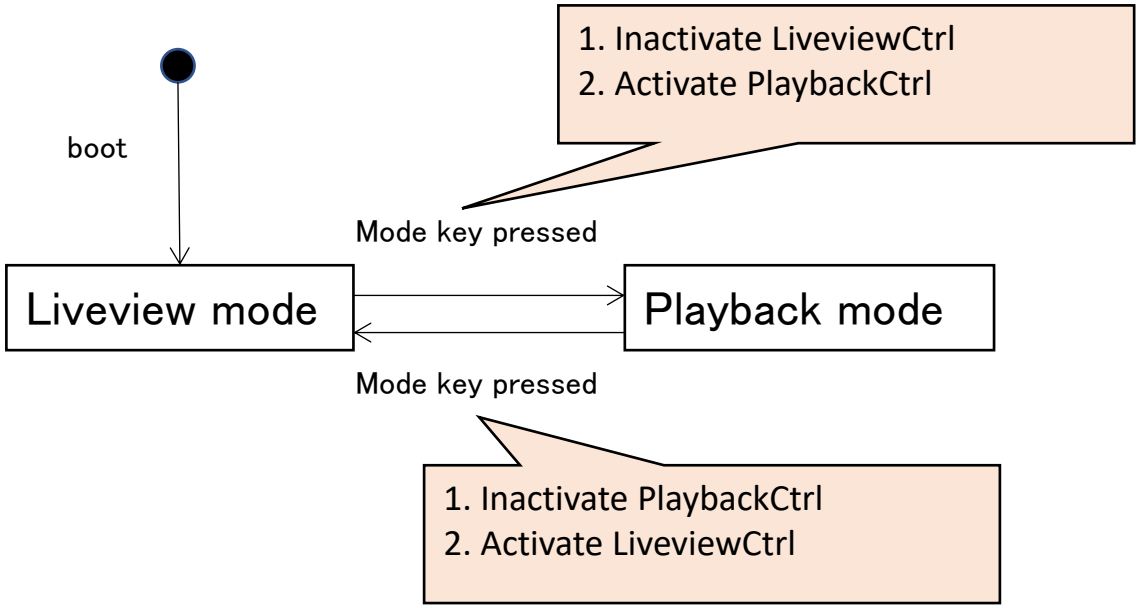
Module	Queue ID	Command
ModeMgr	QUEUE_MODE_MGR	
LiveviewCtrl	QUEUE_LIVEVIEW_CTRL	CMD_START CMD_STOP
PlaybackCtrl	QUEUE_CAPTURE_CTRL	CMD_START CMD_STOP
Input	QUEUE_INPUT	CMD_REGISTER CMD_UNREGISTER

Input

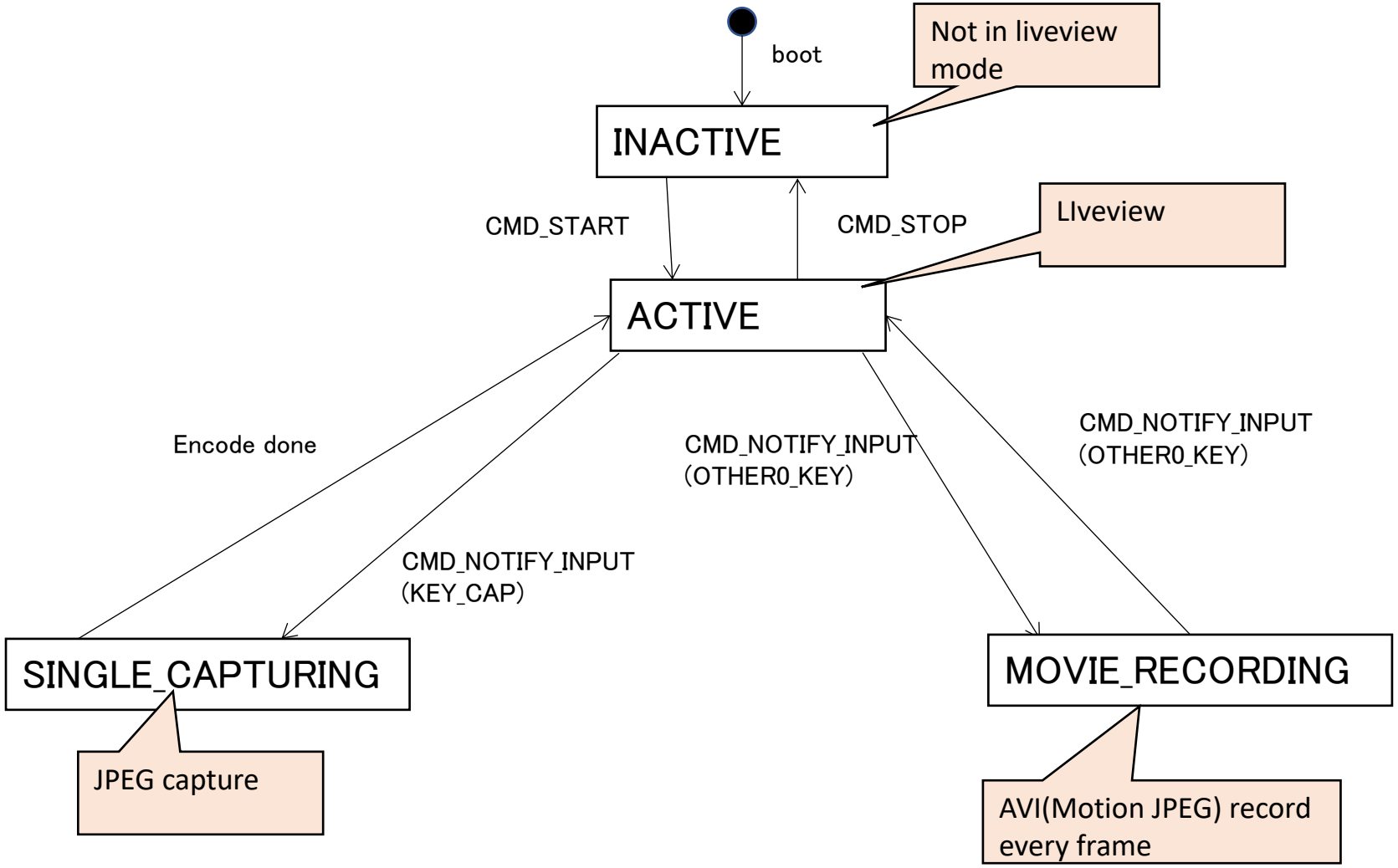


- The input module abstracts input devices
  - Button(GPIO) → Key
  - Rotary Encoder → Dial
- Don’t use interrupt to check input device status
  - To avoid disturbing other critical process such as camera control

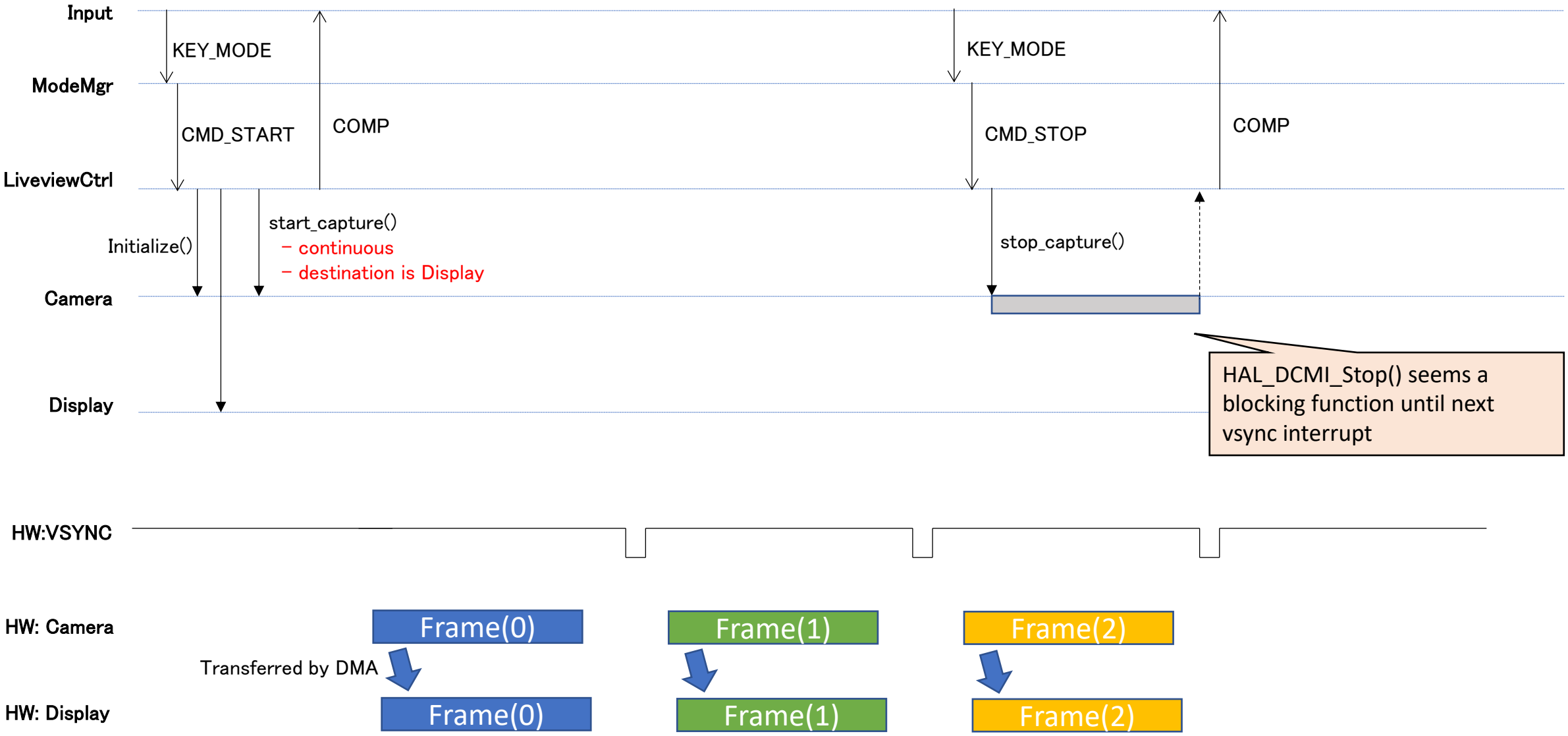
Mode Manager



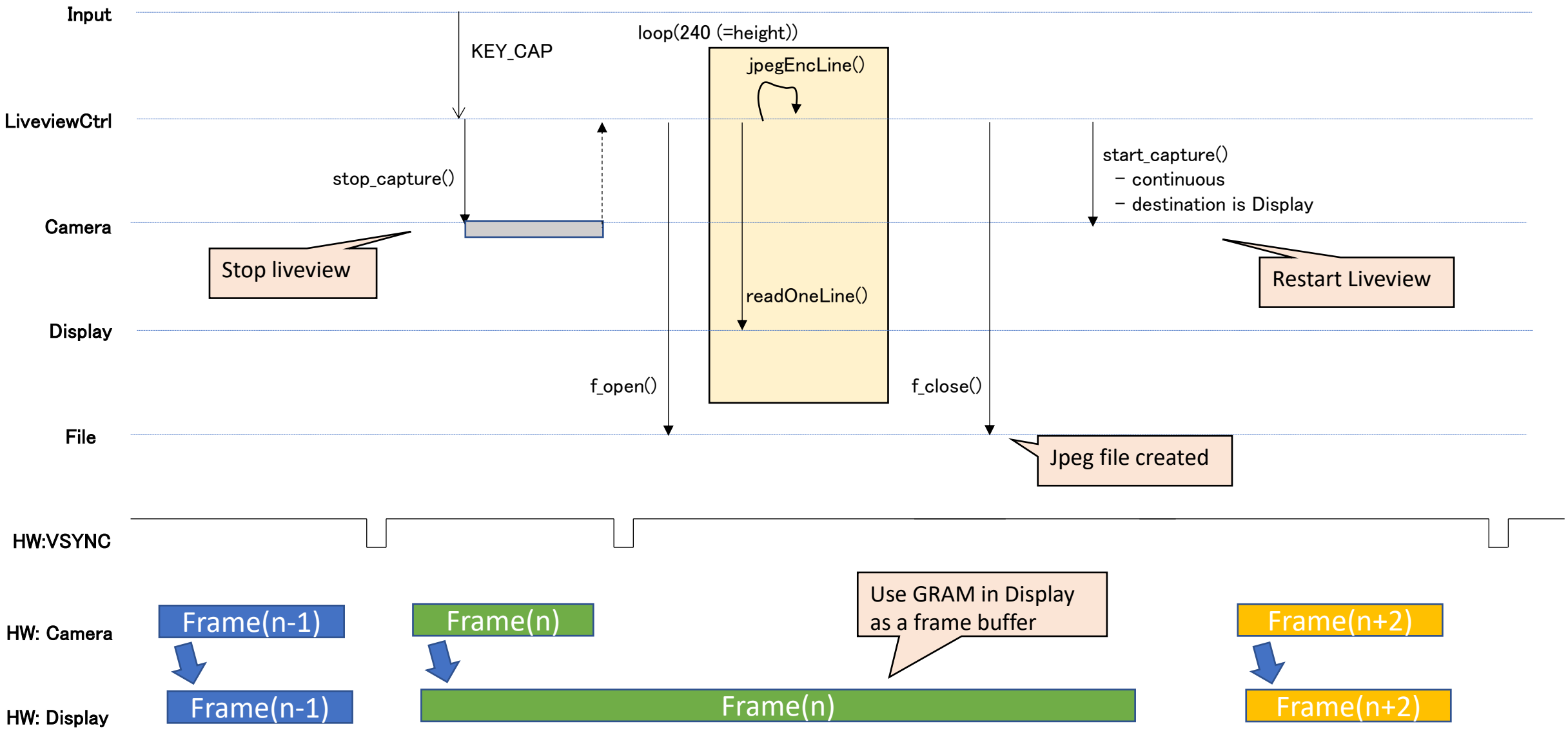
LiveviewCtrl : State Machine



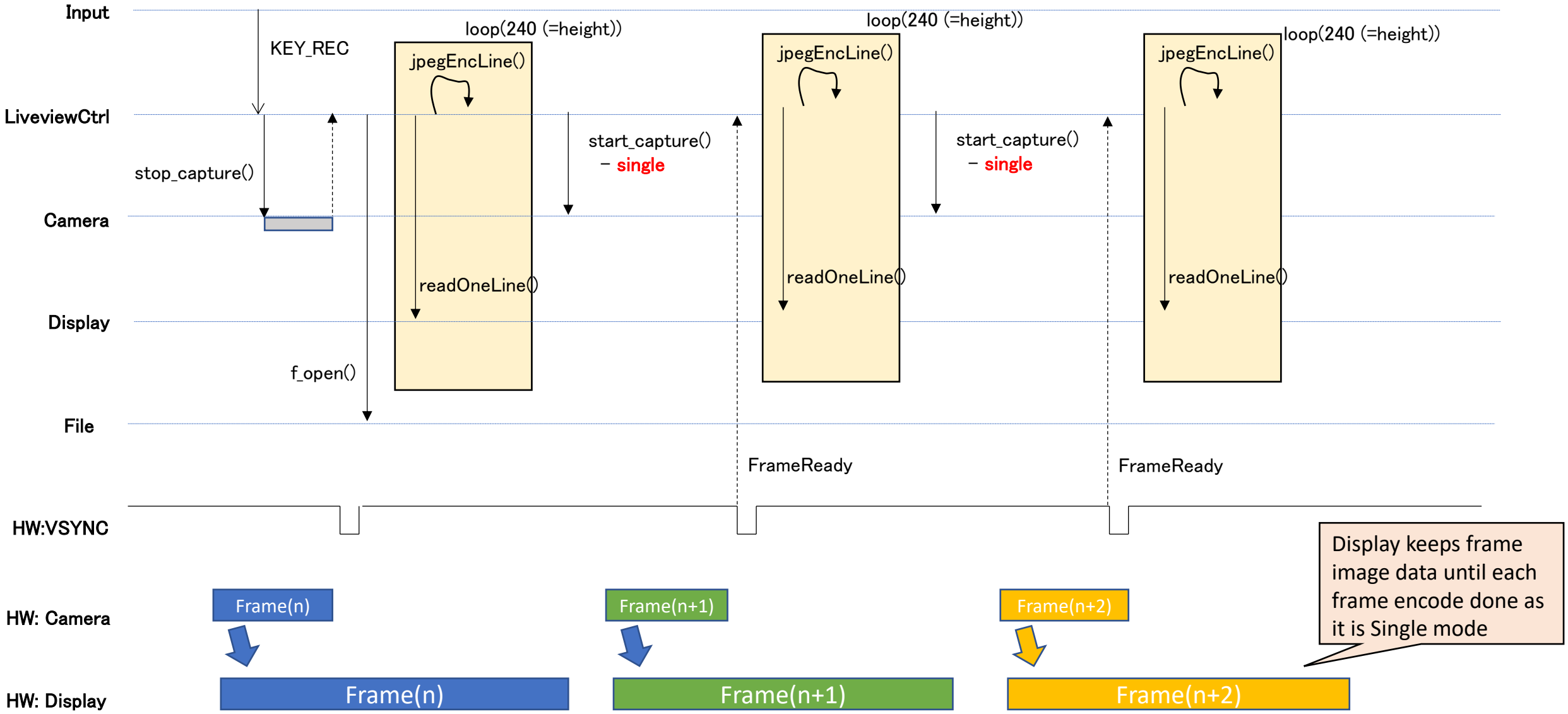
LiveviewCtrl : Liveview sequence



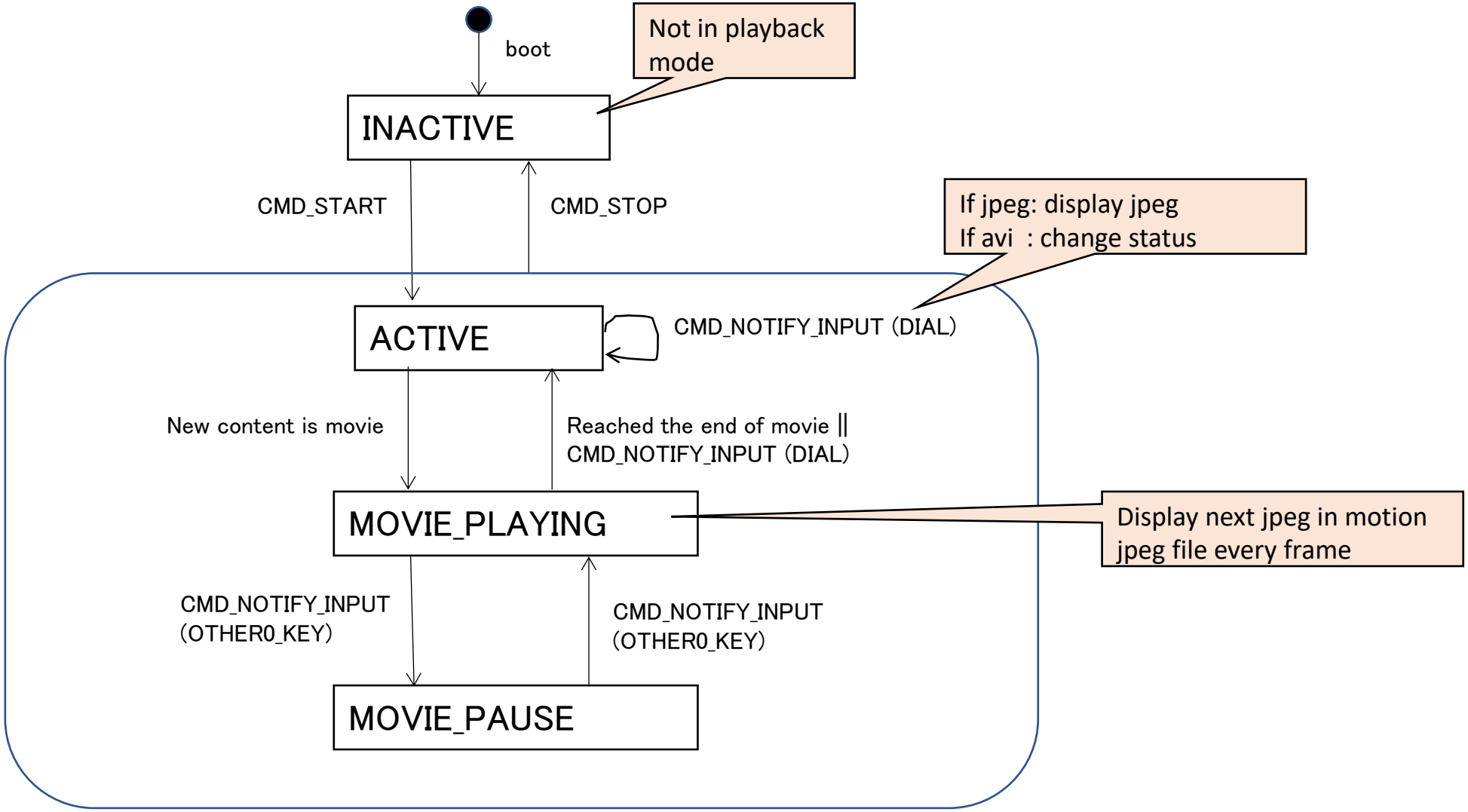
LiveviewCtrl : JPEG Capture sequence



LiveviewCtrl : Movie Record sequence



PlaybackCtrl ：StateMachine



Hardware Configurations

FSMC (for LCD(ILI9341))

```
FSMC_NORSRAM_TimingTypeDef Timing;

 /** Perform the SRAM1 memory initialization sequence
 */
 hsram1.Instance = FSMC_NORSRAM_DEVICE;
 hsram1.Extended = FSMC_NORSRAM_EXTENDED_DEVICE;
 /* hsram1.Init */
 hsram1.Init.NSBank = FSMC_NORSRAM_BANK1;
 hsram1.Init.DataAddressMux = FSMC_DATA_ADDRESS_MUX_DISABLE;
 hsram1.Init.MemoryType = FSMC_MEMORY_TYPE_SRAM;
 hsram1.Init.MemoryDataWidth = FSMC_NORSRAM_MEM_BUS_WIDTH_16;
 hsram1.Init.BurstAccessMode = FSMC_BURST_ACCESS_MODE_DISABLE;
 hsram1.Init.WaitSignalPolarity = FSMC_WAIT_SIGNAL_POLARITY_LOW;
 hsram1.Init.WrapMode = FSMC_WRAP_MODE_DISABLE;
 hsram1.Init.WaitSignalActive = FSMC_WAIT_TIMING_BEFORE_WS;
 hsram1.Init.WriteOperation = FSMC_WRITE_OPERATION_ENABLE;
 hsram1.Init.WaitSignal = FSMC_WAIT_SIGNAL_DISABLE;
 hsram1.Init.ExtendedMode = FSMC_EXTENDED_MODE_DISABLE;
 hsram1.Init.AsynchronousWait = FSMC_ASYNCHRONOUS_WAIT_DISABLE;
 hsram1.Init.WriteBurst = FSMC_WRITE_BURST_DISABLE;
 hsram1.Init.PageSize = FSMC_PAGE_SIZE_NONE;
 /* Timing */
 Timing.AddressSetupTime = 2;
 Timing.AddressHoldTime = 15;
 Timing.DataSetupTime = 4;
 Timing.BusTurnAroundDuration = 1;
 Timing.CLKDivision = 16;
 Timing.DataLatency = 17;
 Timing.AccessMode = FSMC_ACCESS_MODE_A;
```

Need tuning depending on display device spec

DCMI (for Camera(OV7670))

```
hdcmi.Instance = DCMI;
 hdcmi.Init.SynchroMode = DCMI_SYNCHRO_HARDWARE;
 hdcmi.Init.PCKPolarity = DCMI_PCKPOLARITY_RISING;
 hdcmi.Init.VSPolarity = DCMI_VSPOLARITY_HIGH;
 hdcmi.Init.HSPolarity = DCMI_HSPOLARITY_LOW;
 hdcmi.Init.CaptureRate = DCMI_CR_ALL_FRAME;
 hdcmi.Init.ExtendedDataMode = DCMI_EXTEND_DATA_8B;
 hdcmi.Init.JPEGMode = DCMI_JPEG_DISABLE;
```

```
hdma_dcmi.Instance = DMA2_Stream1;
   hdma_dcmi.Init.Channel = DMA_CHANNEL_1;
   hdma_dcmi.Init.Direction = DMA_PERIPH_TO_MEMORY;
   hdma_dcmi.Init.PeriphInc = DMA_PINC_DISABLE;
   hdma_dcmi.Init.MemInc = DMA_MINC_DISABLE;
   hdma_dcmi.Init.PeriphDataAlignment = DMA_PDATAALIGN_WORD;
   hdma_dcmi.Init.MemDataAlignment = DMA_MDATAALIGN_WORD;
   hdma_dcmi.Init.Mode = DMA_NORMAL;
   hdma_dcmi.Init.Priority = DMA_PRIORITY_LOW;
   hdma_dcmi.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
```

Port Map

Port Map

```
## IO
PA00 = ROTARY_A (TIM5_CH1)       ! pulled-up on board
PA01 = ROTARY_B (TIM5_CH2)
PA02 = USART2 (TX)
PA03 = USART2 (RX)
PA04 = CAMERA_HS(DCMI_HSYNC)
PA05 = SD_CARD(SPI1_SCK)
PA06 = CAMERA_PCLK(DCMI_PIXCK)
PA07 = SD_CARD(SPI1_MOSI)
PA08 = CAMERA_MCLK(MCO1)
PA09 = [VBUS_FS]
PA10 = [OTG_FS_ID]
PA11 = [OTG_FS_DM]
PA12 = [OTG_FS_DP]
PA13 = [SWDIO]
PA14 = [SWCLKDebug]
PA15 = SD_CARD(SPI1_NSS(SW control))

PB00 = N/A
PB01 = N/A
PB02 = [BOOT]
PB03 = [SWO]
PB04 = SD_CARD(SPI1_MISO)
PB05 = N/A
PB06 = CAMERA_D5(DCMI_D5)
PB07 = CAMERA_VS(DCMI_VSYNC)
PB08 = CAMERA_D6(DCMI_D6)
PB09 = CAMERA_D7(DCMI_D7)
PB10 = CAMERA(I2C2_SCL)
PB11 = CAMERA(I2C2_SDA)
PB12 = N/A
PB13 = N/A
PB14 = N/A
PB15 = N/A

PC00 = [OTG_FS_PowerSW]  !Don't use
PC01 = BUTTON1
PC02 = BUTTON2
PC03 = BUTTON3
PC04 = N/A
PC05 = CAMERA_RESET
PC06 = CAMERA_D0(DCMI_D0)
PC07 = CAMERA_D1(DCMI_D1)
PC08 = CAMERA_D2(DCMI_D2)
PC09 = CAMERA_D3(DCMI_D3)
PC10 = [CS43L22_SCLK]   !Don't use
PC11 = CAMERA_D4(DCMI_D4)
PC12 = [CS43L22_SDIN]   !Don't use
PC13 = N/A
PC14 = [OSC32_IN]
PC15 = [OSC32_IN]

PD00 = LCD_D2(FSMC_D2)
PD01 = LCD_D3(FSMC_D3)
PD02 = N/A
PD03 = N/A
PD04 = LCD_RD(FSMC_NOE)
PD05 = LCD_WD(FSMC_NWE)
PD06 = N/A
PD07 = LCD_CS(FSMC_NE1)
PD08 = LCD_D13(FSMC_D13)
PD09 = LCD_D14(FSMC_D14)
PD10 = LCD_D15(FSMC_D15)
PD11 = LCD_RS(FSMC_A16)
PD12 = LED
PD13 = LED
PD14 = LED,  LCD_D0(FSMC_D0)
PD15 = LED,  LCD_D1,(FSMC_D1)

PE00 = [LIS302DL_INT1]  !Don't use
PE01 = [LIS302DL_INT2]  !Don't use
PE02 = [LIS302DL_CS]    !Always HIGH
PE03 = N/A
PE04 = N/A
PE05 = N/A
PE06 = N/A
PE07 = LCD_D4(FSMC_D4)
PE08 = LCD_D5(FSMC_D5)
PE09 = LCD_D6(FSMC_D6)
PE10 = LCD_D7(FSMC_D7)
PE11 = LCD_D8(FSMC_D8)
PE12 = LCD_D9(FSMC_D9)
PE13 = LCD_D10(FSMC_D10)
PE14 = LCD_D11(FSMC_D11)
PE15 = LCD_D12(FSMC_D12)


## Function
USART2 = TERMINAL
SPI1   = SD CARD
I2C2   = CAMERA(OV7670)
FSMC   = LCD(ILI9341)
DCMI   = CAMERA(OV7670)
TIMER5_CH1/2 = Rotary Encoder

## DMA
DMA1_5 = USART2_RX
DMA2_1 = CAMERA(DCMI->FSMC(LCD))

## OSC
PH0 = OSC_IN   (external 8MHz)
PH1 = OSC_OUT  (external 8MHz)

## Note
LCD_RESET    = VDD
CAMERA_PWDN  = GND
```

backup

APIs

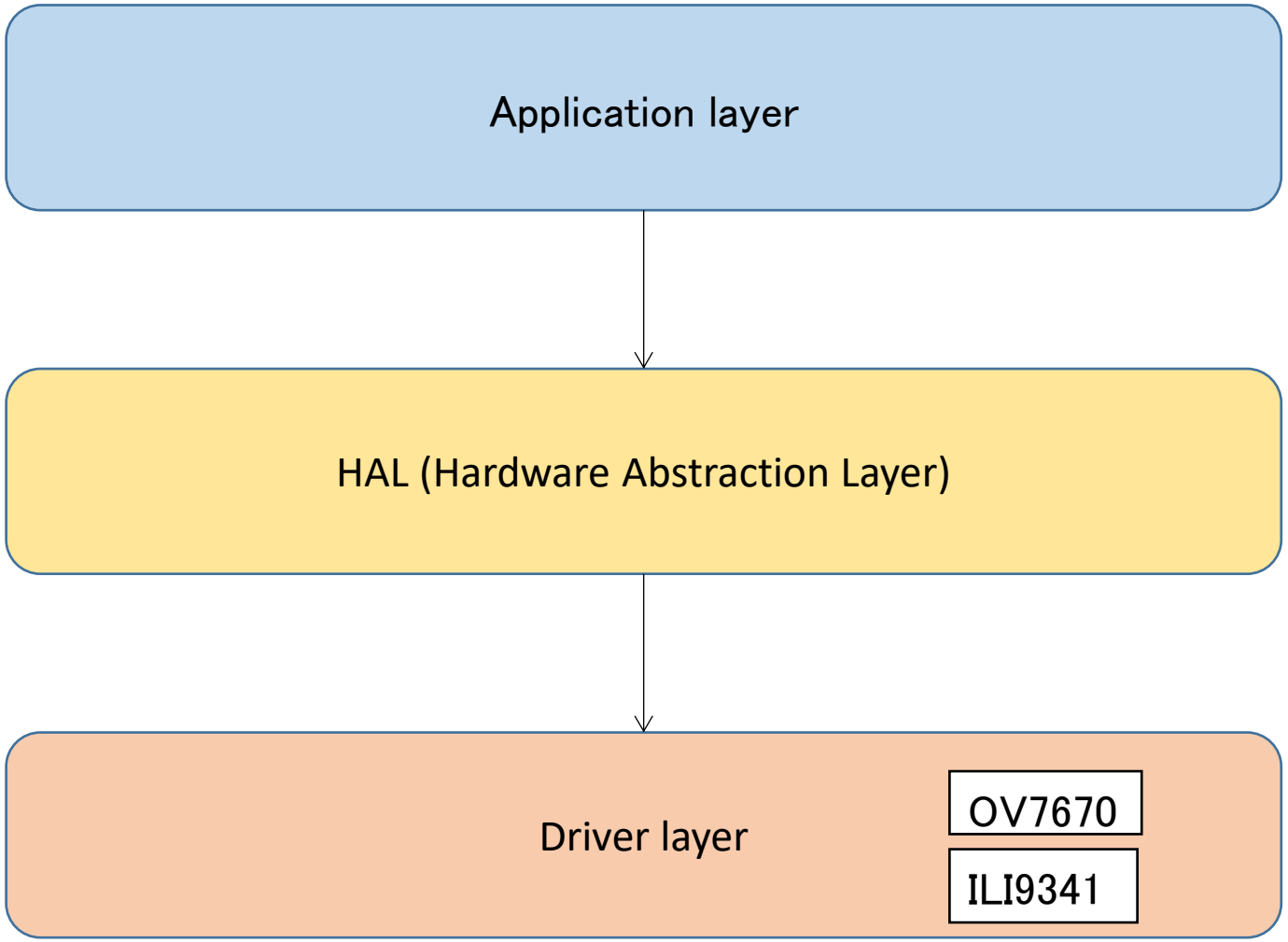
Module	Queue ID	COMMAND
modeMgr	QUEUE_MODE_MGR	NOTIFY_EXIT
liveviewCtrl	QUEUE_LIVEVIEW_CTRL	START STOP
captureCtrl	QUEUE_CAPTURE_CTRL	START
playbackCtrl	QUEUE_CAPTURE_CTRL	START STOP
input	QUEUE_INPUT	REGIST UNREGIST

When a control module exits (by itself?), the control module sends this message

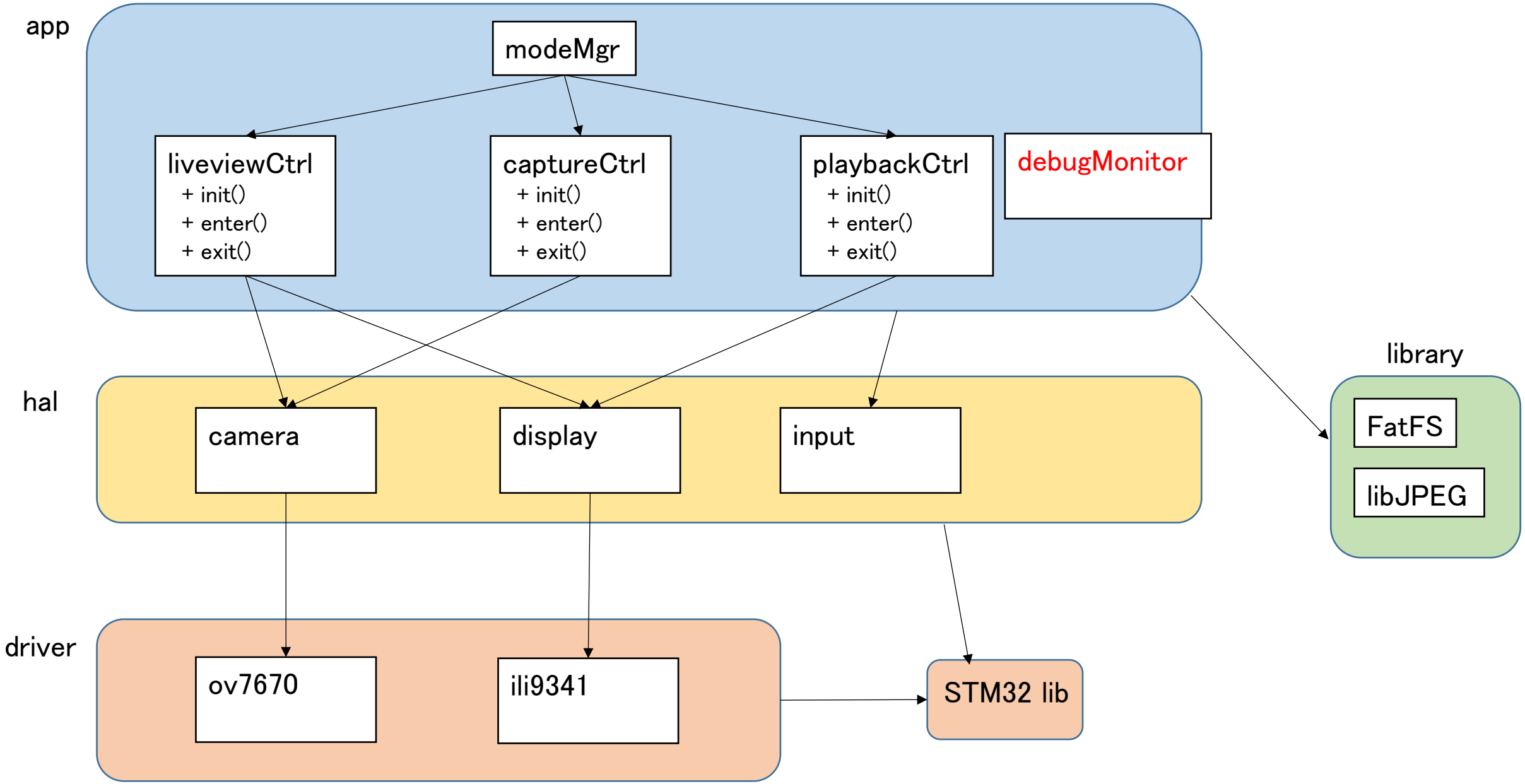
modeMgr sends these messages when mode changed

Any module which wants to be notified input status calls this

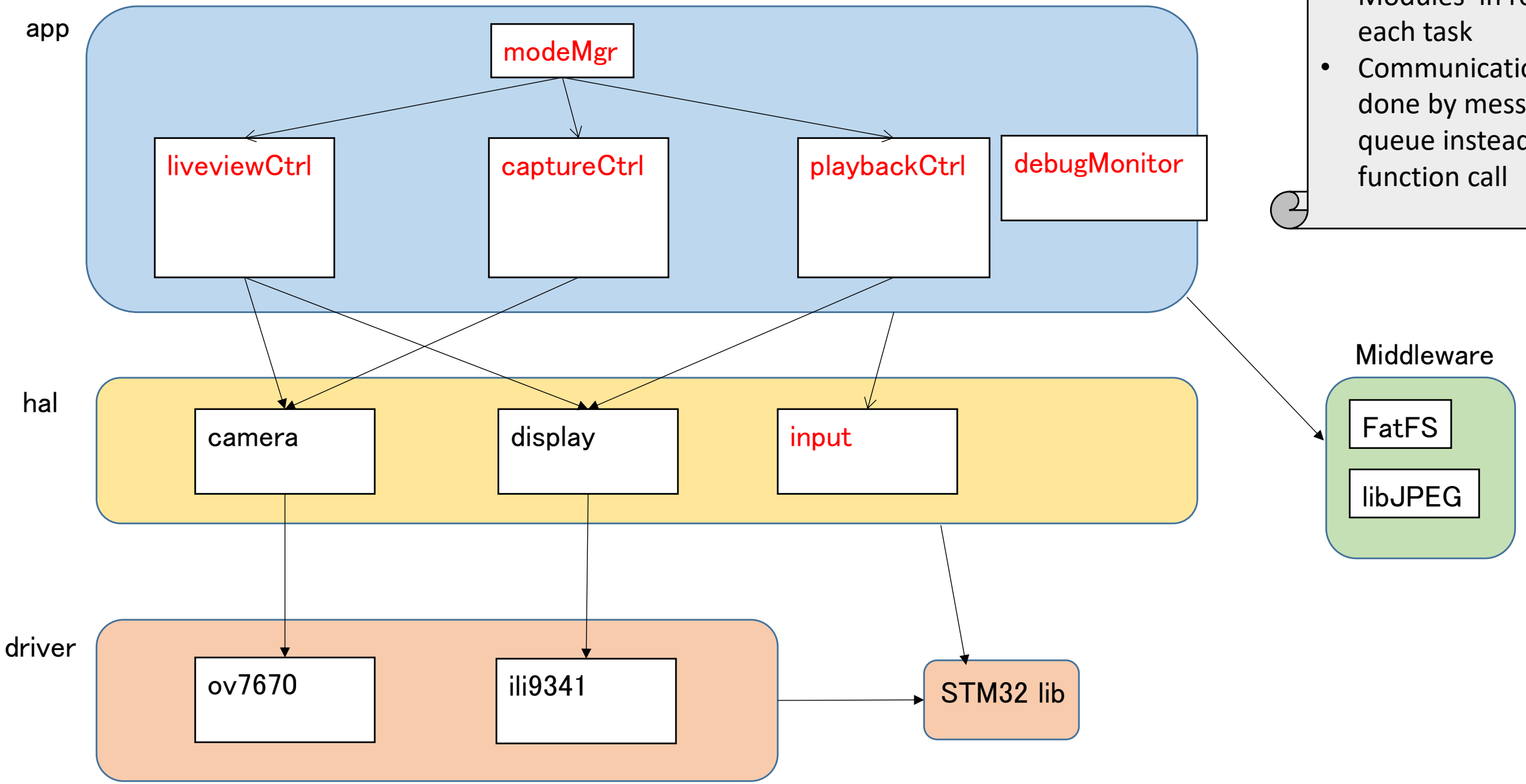
Software structure (to be)



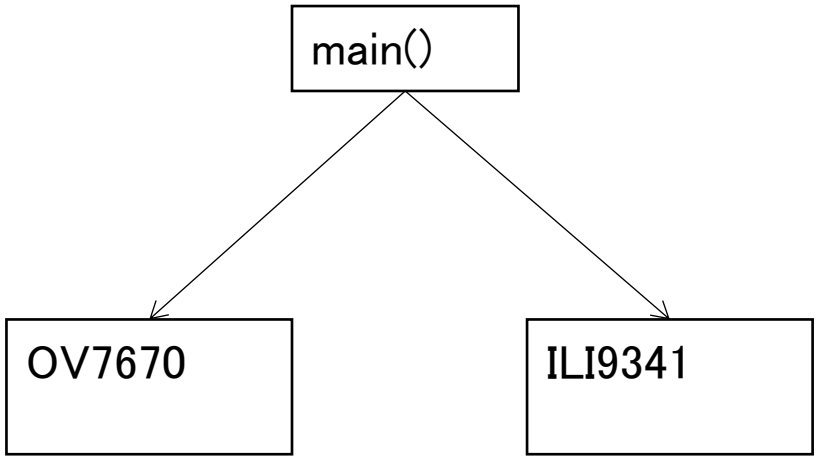
Software structure



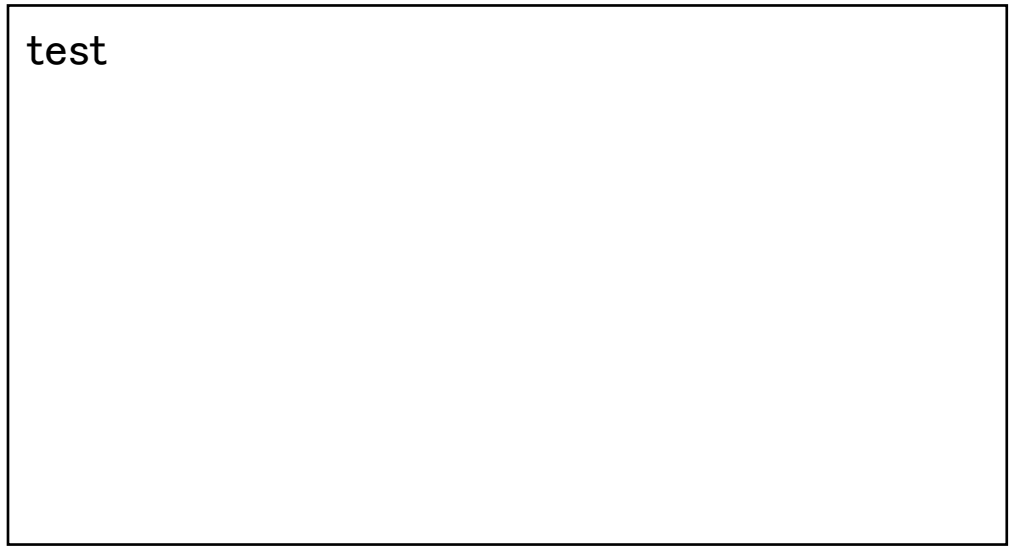
Software structure



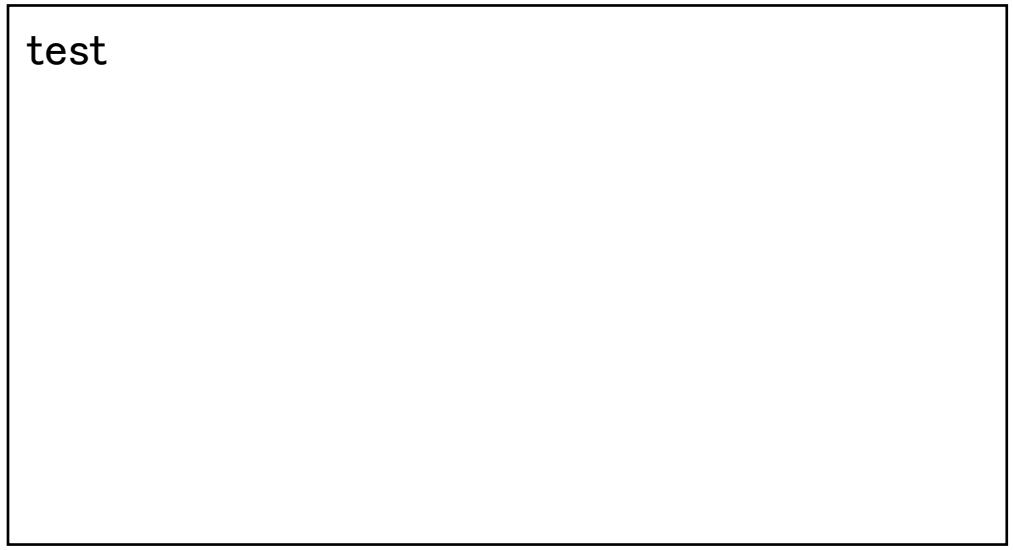
Software structure (current test program)



Overview (HW)



Overview (HW)



Overview (HW)