Monocle Documentation

Release v1.6

Brilliant Labs

Jun 13, 2022

INTRODUCTION

This document was generated on 2022-06-13 at 09:32.

CHAPTER

ONE

INTRODUCTION TO MONOCLE

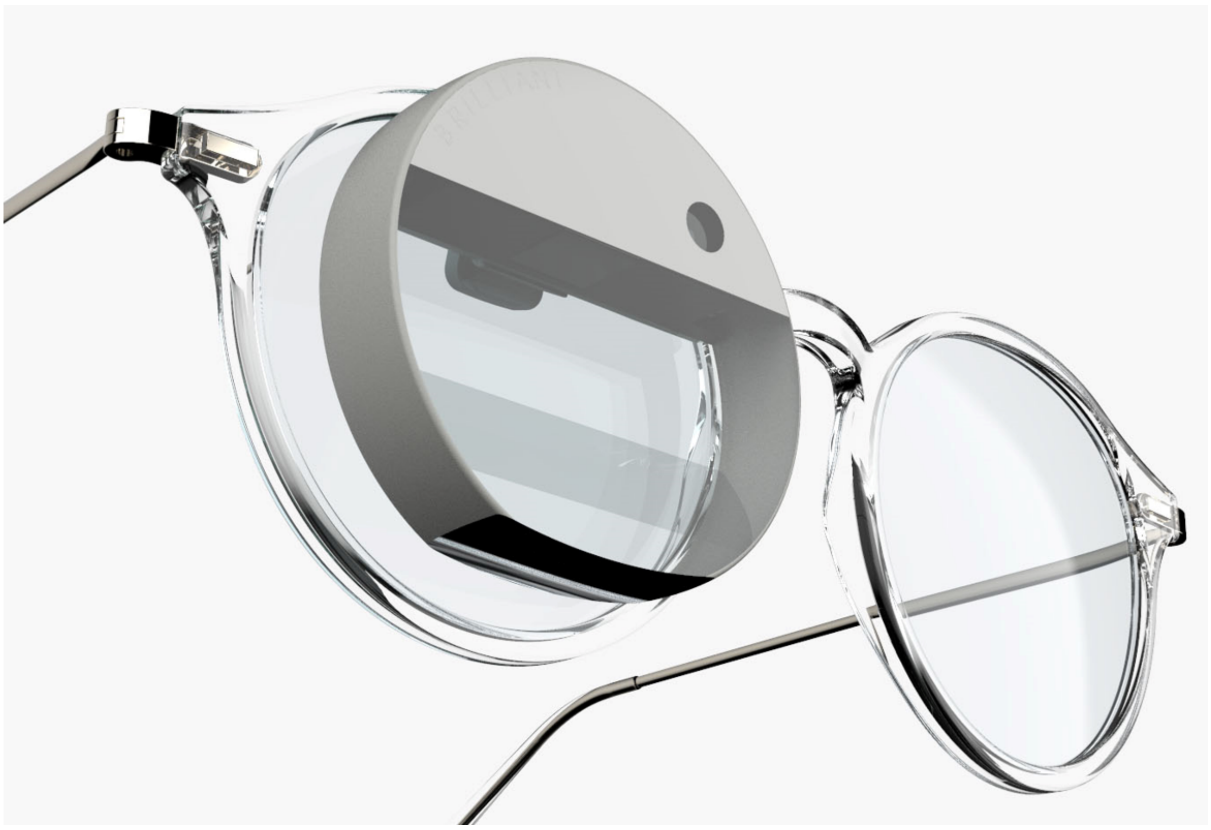


Fig. 1: *Monocle.*

AR is no longer limited to laboratories and fieldwork. Monocle is a pocket-sized device that clips right to your glasses. What it does best is help you collect memories. Share your clips and photos. Zoom in on far away objects. All without ever having to take your phone out of your pocket. Live life without looking down at your phone.

Visit Monocle to get hands on one.

This document describes the following about Monocle Product:

- *Get Started with Monocle Firmware Development.*
- *Get Started with Monocle FPGA Development.*
- *Get Started with Monocle Mobile App Development.*
- *Hardware Specifications.*
- *Highlevel Software Architecture of Monocle Firmware.*
- *Lowlevel Software Design of Monocle Firmware.*

- *Phone App Design Details*
- *Documentation tools*
- *Firmware/FPGA Release Notes*
- *Phone Application Release Notes*
- *Future implementations.*

1.1 Audience

This document is mainly for software developers to the opensource community and testers who are contributing to build firmware, FPGA and Mobile application for the monocle product.

CHAPTER

TWO

PRODUCT SPECIFICATIONS

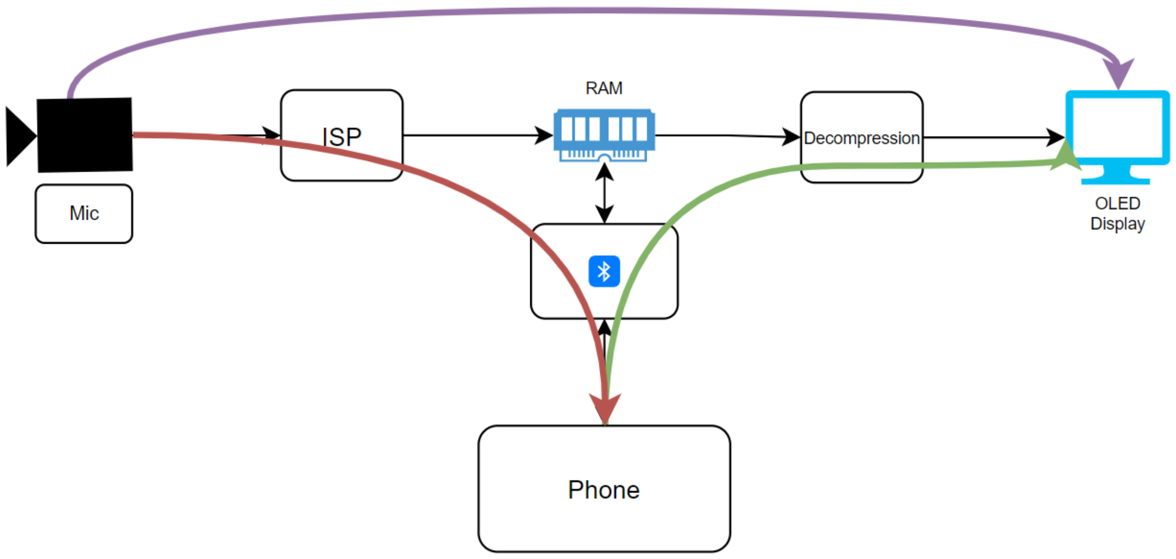


Fig. 1: *Monocle Data Flow Diagram*

2.1 Features for End Users

The video and audio is always captured by Monocle.

1. Tap -> Displays live video
2. Press -> Captures Photo and sent to Phone.
3. Double Press -> Replays video, Transfers the existing video to the phone.

There are other features like, Slow-motion replay and Zoom in functionality as well.

CHAPTER

THREE

GET STARTED WITH FIRMWARE ON MCU

This section describes all the necessary steps a developer wants to get start with monocle firmware development.

3.1 Setup the Environment on your Windows Machine

1. Install Segger Embedded Studio V5.44
   - **(Optional) Licensing**
     - Request a Nordic license
     - Receive emailed license
     - 'Tools' -> 'License Manager. . . ' and click 'Activate Embedded Studio'
   - **Add software packs for Cortex-ARM and nRF**
     - Tools -> Package Manager
     - Yes to get latest list
     - Find CMSIS-CORE Support Package (not CMSIS 5 CMSIS-CORE Support Package)
     - Right-click to get context menu, and select Install Selected Package
     - the Action column will change from No Action to Install
     - Scroll down to find Nordic nRF CPU Support Package
     - Right-click and select as above (note: this will also select the CMSIS 5 CMSIS-CORE Support Package)
     - Click Next button to download & install the 3 packages
     - Click Finish
   - **(Optional)** Enable the CMSIS Configuration Wizard (provides graphical `sdk_config.h` file editing)
     - Download and install Java
     - See instructions click here & helpful video here
     - (will need to restart SES for tool to become available)
     - (if using in own project, may need to see this) instructions
2. Install nRF5 SDK v17.0.2 + Softdevice S132
   - Unizp the SDK archive (this will take ~1 hour)
   - **(Important)** Place it in the location of your choice, suggested: `C:\Nordic\`

3.2 Clone Monocle git

```
git clone git@github.com:Itsbrilliantlabs/Monocle.git
```

3.3 Folder Structure of git

```
monocle
   ├── .gitignore
   ├── docs
   ├── fpga
   └── mcu
```

- `docs/` contains reStructuredText source code for documentation.
- `fpga/` contains source code of FPGA.
- `mcu/` contains source code of MCU.

3.4 Build the code

Navigate to, and double-click on the SES project file for the hardware you wish to build for

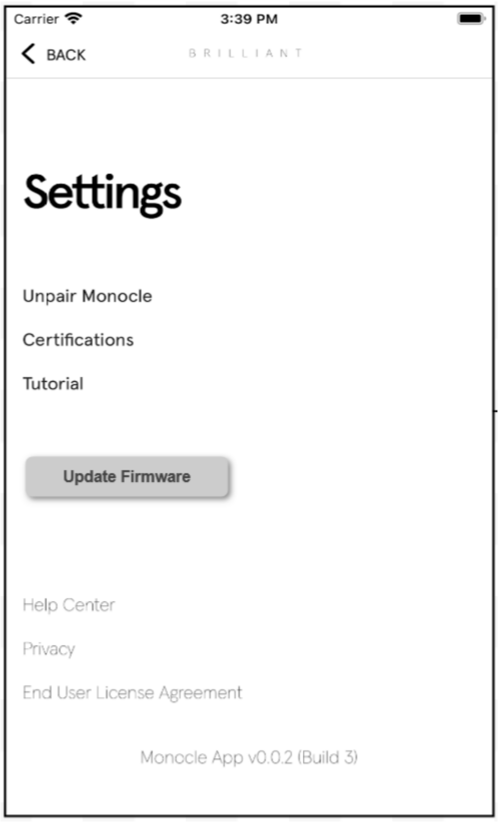
- For example, for MK11, double click: `C:\Nordic\nRF5_SDK_17.0.2_d674dde\examples\monocle\mcu\mk11\s132\ses\monocle+ble_mk11_s132.emProject`
- In Build menu, select the first action (yellow bricks icon, F7 hotkey) to build the code
- If all is well, it should build without errors or warnings

3.5 How to run the code on the Monocle Hardware

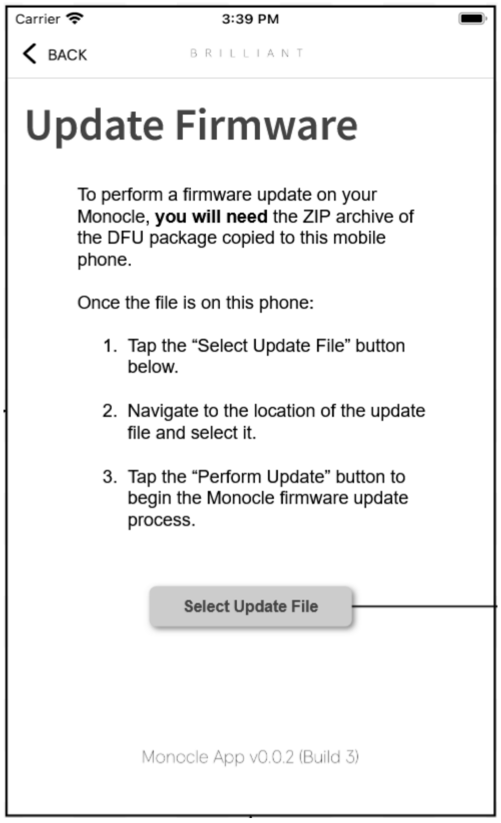
Update the Monocle firmware using Brilliant Mobile Application. Make sure the Bluetooth is paired with Monocle Hardware.

Follow the steps shown below:

1. Go to Settings and click on “Upgrade Firmware” Button.



2. Click on “Select Update File”. Your File manager will open, select the “zip” file.

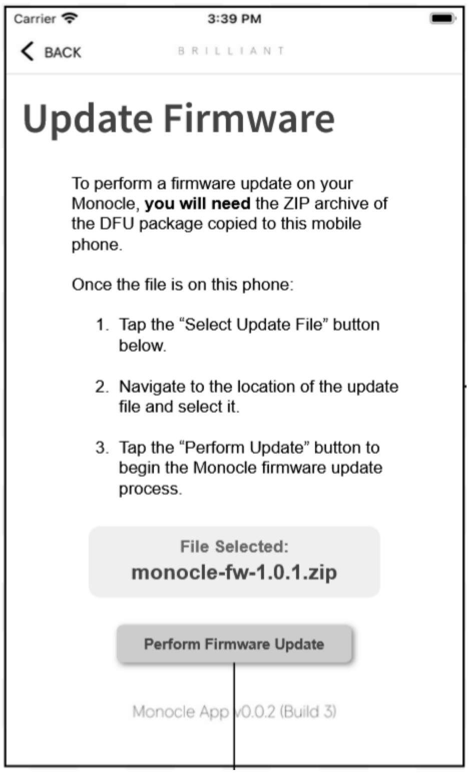


To perform a firmware update on your Monocle, **you will need** the ZIP archive of the DFU package copied to this mobile phone.

Once the file is on this phone:

1. Tap the “Select Update File” button below.
2. Navigate to the location of the update file and select it.
3. Tap the “Perform Update” button to begin the Monocle firmware update process.

3. Click on “Perform Firmware Update”


Carrier 3:39 PM

BACK BRILLIANT

Update Firmware

To perform a firmware update on your Monocle, **you will need** the ZIP archive of the DFU package copied to this mobile phone.

Once the file is on this phone:

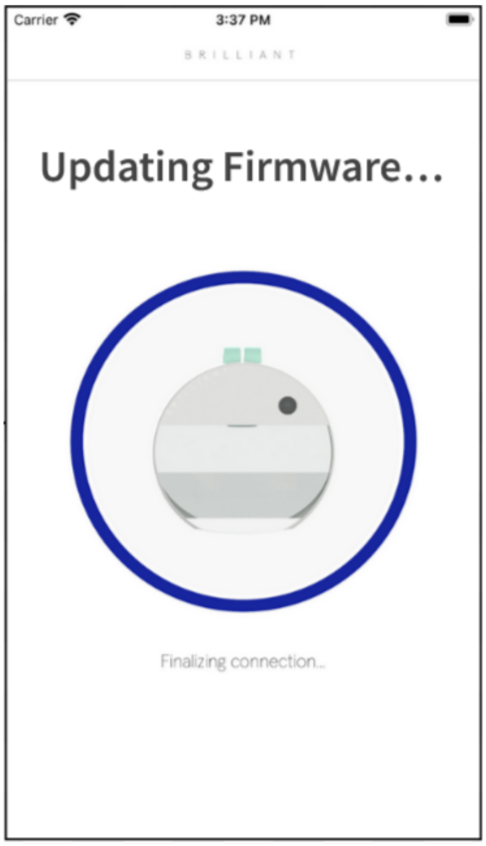
1. Tap the "Select Update File" button below.
2. Navigate to the location of the update file and select it.
3. Tap the "Perform Update" button to begin the Monocle firmware update process.

File Selected:
monocle-fw-1.0.1.zip

Perform Firmware Update

Monocle App v0.0.2 (Build 3)

4. Wait for the update to complete.



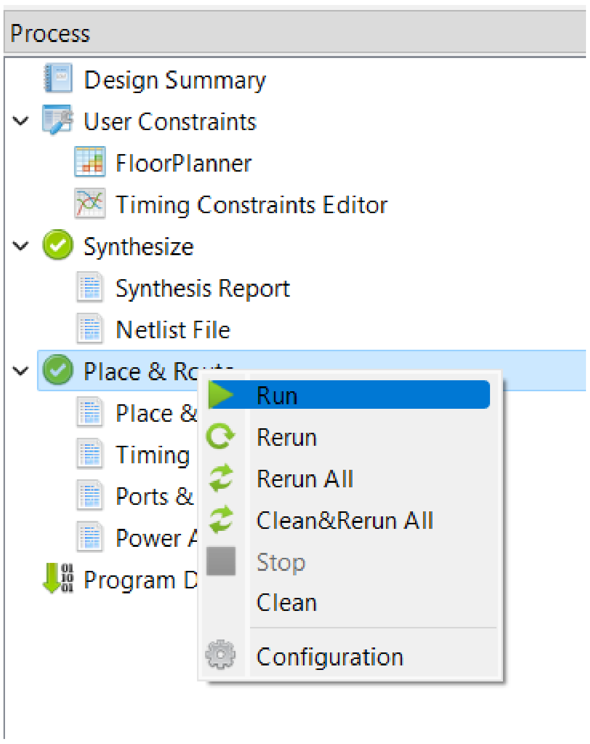
CHAPTER

FOUR

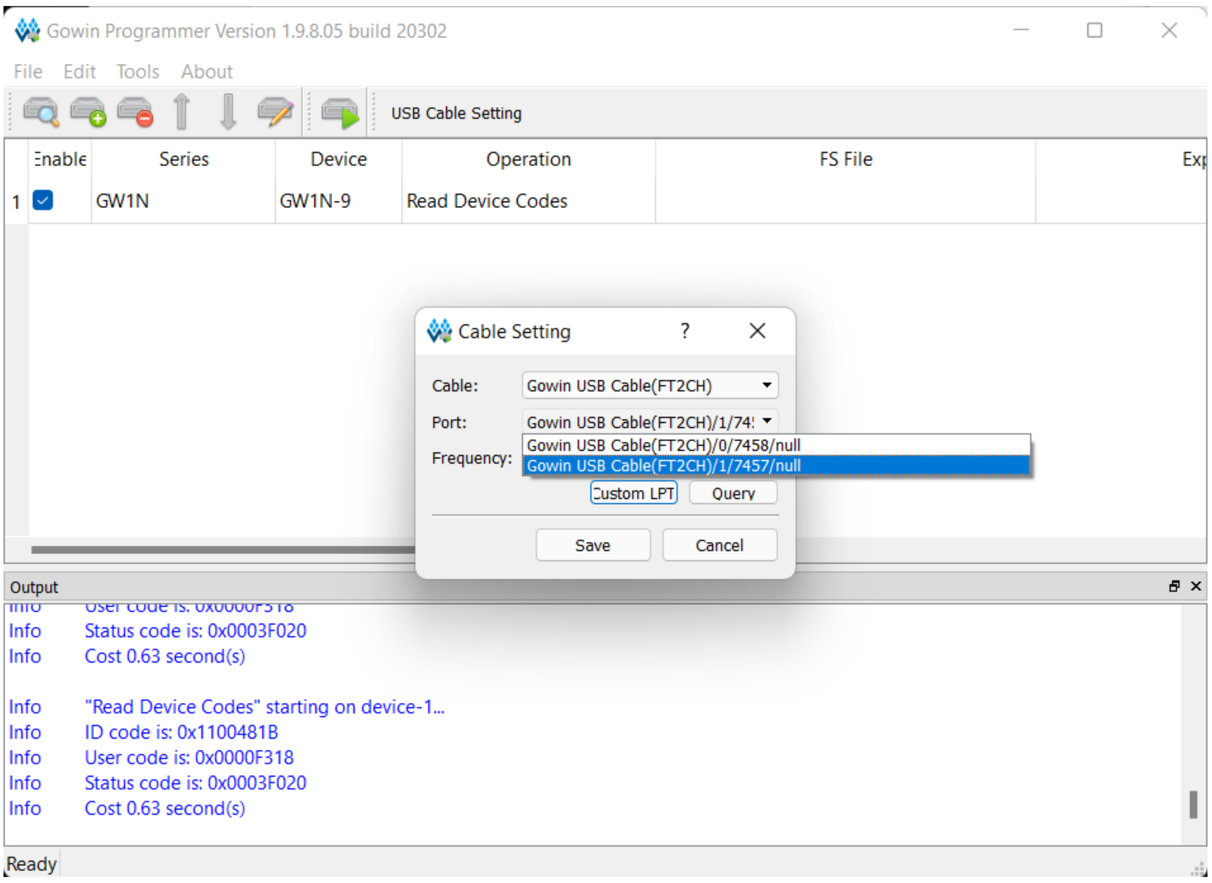
GET STARTED WITH FPGA DEVELOPMENT

4.1 Steps

1. Install the toolchain by registering for a GoWin account and download the software at: https://www.gowinsemi.com/en/support/download_eda/. This will require you to request a license.
2. Follow instructions to install the license and enable the toolchain.
3. Open the GoWin tool, open the fpga_proj.gprj file from the File -> Open menu
4. Right click on the "Place & Route" and click on Run.



5. This will result in a bitfile that can be used to program the flash on the Monocle.
6. Once the Gowin toolchain is installed, open the Gowin programmer application.
7. Select USB Cable setting and set the port to be channel 1 per the screenshot below and hit the Save button.


Gowin Programmer Version 1.9.8.05 build 20302
File Edit Tools About
USB Cable Setting
Cable Setting
Cable: Gowin USB Cable(FT2CH)
Port: Gowin USB Cable(FT2CH)/1/74!
Gowin USB Cable(FT2CH)/0/7458/null
Gowin USB Cable(FT2CH)/1/7457/null
Frequency:
Custom LPT
Query
Save
Cancel
Output
Ready

	Enable	Series	Device	Operation	FS File	Exp
1	✓	GW1N	GW1N-9	Read Device Codes		

```
Info    User code is: 0x0000F318
Info    Status code is: 0x0003F020
Info    Cost 0.63 second(s)

Info    "Read Device Codes" starting on device-1...
Info    ID code is: 0x1100481B
Info    User code is: 0x0000F318
Info    Status code is: 0x0003F020
Info    Cost 0.63 second(s)
```

8. Select the Series to be GW1N, Device to be GW1N-9 and operation to be Read Device Codes
9. Click on the Green Program/Configure button to read out the ID of the FPGA.
10. You should get the JTAG ID of the FPGA as seen in the screenshot above.

CHAPTER

FIVE

GET STARTED WITH MONOCLE MOBILE APP

5.1 Configuration

You should place a configuration file at the project root: `<project_root>/.env`. There is an example `.env.example` file to reference for the expected format. The .env file has the following params:

1. LOG_LEVEL=DEBUG|INFO|WARN|ERROR (optional, defaults to INFO)
2. ENVIRONMENT_TYPE=DEBUG|STAGE|PRODUCTION (optional, defaults to PRODUCTION)

5.2 Set Up

1. Make sure you have a recent version of `npm` installed.
2. Make sure `yarn` is installed, e.g. `npm i -g yarn`.
3. In the project root, install all the npm packages: `yarn install`.
4. For iOS, make sure you have Cocoapods installed: [https://guides.cocoapods.org/using/getting-started.html{]}(https://guides.cocoapods.org/using/getting-started.html).
5. Install the pods: `cd ios`, then `pod install`.
6. You can now open the `xcworkspace` file and then run the app using Xcode.
7. For Android, you should generate your own keystore, e.g.: `keytool -genkey -v -keystore ./android/app/debug.keystore -storepass android -alias androiddebugkey -keypass android -keyalg RSA -keysize 2048 -validity 10000`
8. You may need to run `yarn start` in a separate terminal window to manually start Metro before running Android in debug mode.
9. Run `yarn android` to start the app on Android device/emulator.

5.3 iOS Distribution Builds

You can use the standard approach in Xcode to make builds–nothing special is required.

5.4 Android Distribution Builds

Put a file `android/app/gradle.properties` with the following contents:

```
release_keystore=your_key_name.keystore
release_keystore_password=your_key_store_password
release_key_alias=your_key_alias
release_key_password=your_key_password
```

Optionally, you could pass the parameters from command-line with the `-P` option. For example, `./gradlew assembleRelease -Prelease_key_password=keypass`.

There is a bash script that's provided, `build_android.sh`, which will automate making builds (there are a few extra steps that are necessary to make the build work, which this script takes care of).

CHAPTER

SIX

FUTURE CHANGES TO MONOCLE

6.1 Monocle Firmware Side

- Audio transfer from Monocle Hardware to Phone Application
- Reliable transfer of data to phone
- Data tranfer from Phone to Monocle Hardware
- FPGA Upgrade feature

6.2 Monocle FPGA Side

- Video Compression for faster Video transfer from Monocle to Phone

6.3 Monocle Phone Application Side

- Application on phone displaying Video/Photo transferred from monocle

CHAPTER

SEVEN

HARDWARE SPECIFICATIONS

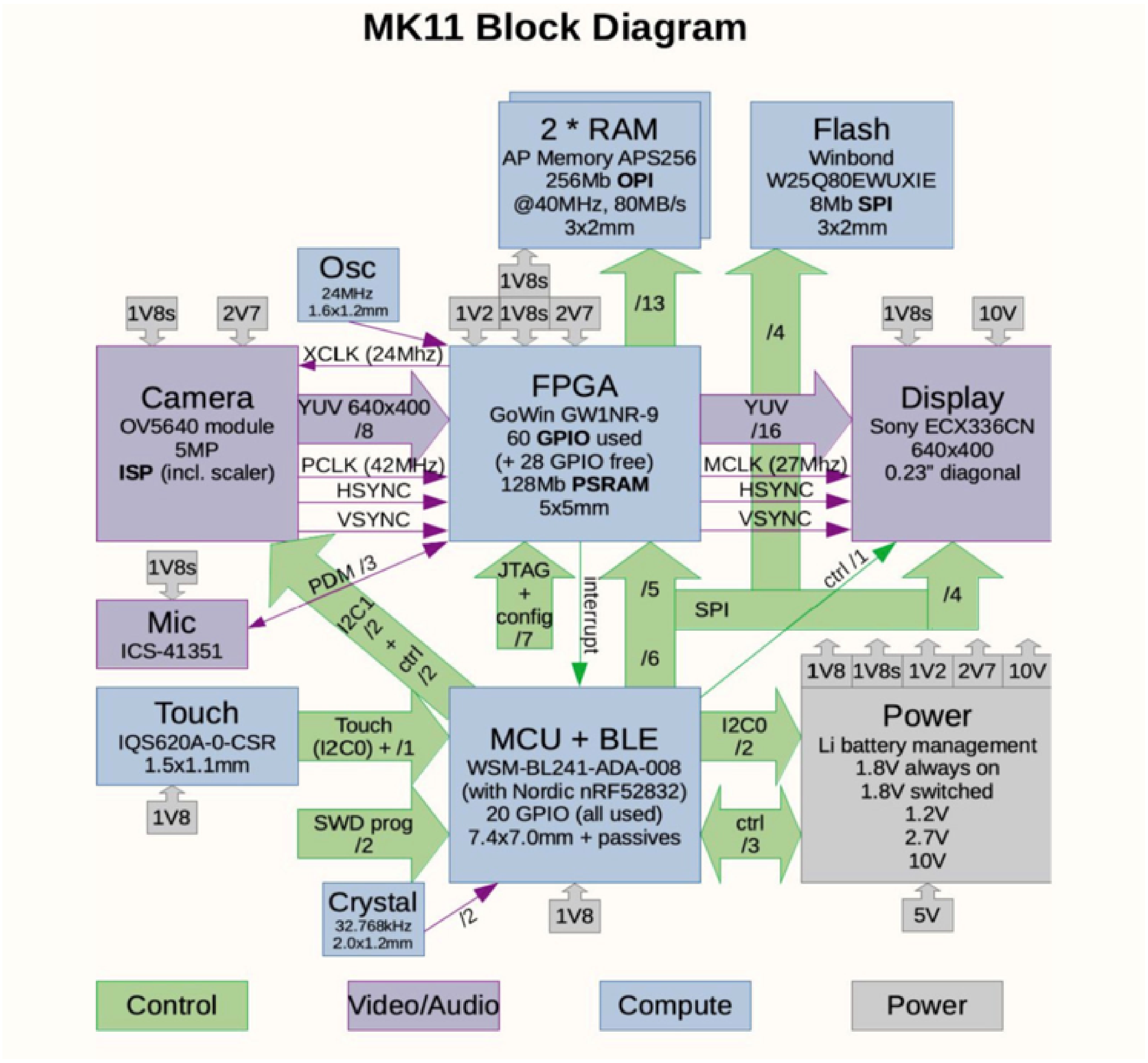
Evolutions of Monocle hardware:

- MK9B - Evaluation Board - **(No Support anymore)**
- MK10 - First Formfactor Board - **(No Support anymore)**
- *MK11 - Second Formfactor Board* - **(Currently in production)**

7.1 MK11 Hardware Specification

Overview of monocle hardware MK11 REV board.

`MK11 Board Schematics`



- MCU runs on NRF52832 SOC.
- Monocle also has Gowin GW1NR-9 FPGA on its Data Path.

7.2 Hardware Features

1. The data path of Camera and Mic is connected to FPGA and the output stored in SRAM. This is done infinitely from Monocle Bootup till monocle sleeps.
2. If any Touch event is detected, MCU commands FPGA to freeze the Frame buffer and Audio buffer stored in SRAM and queries the Frame to Transfer through BLE and/or Display the image to OLED.

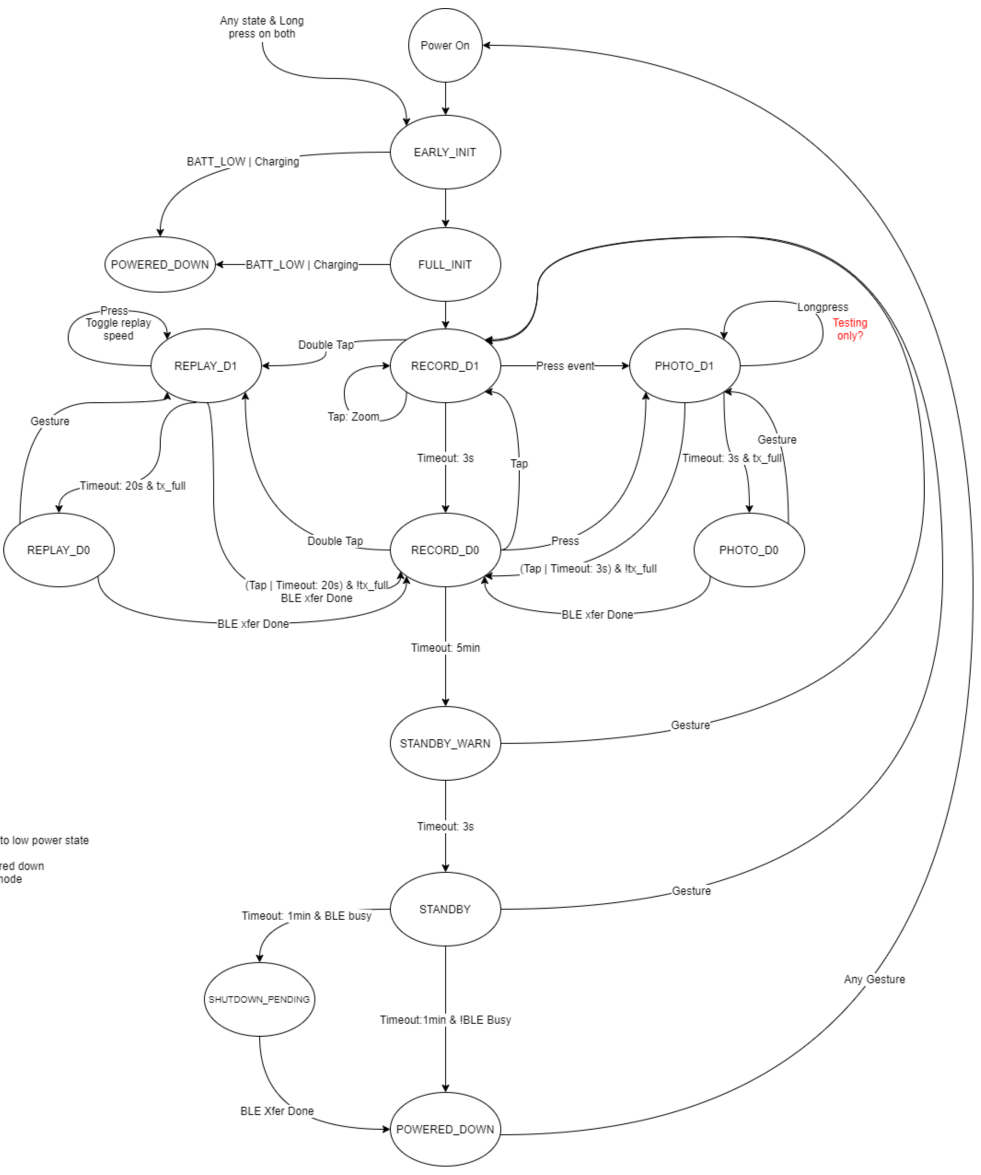
CHAPTER

EIGHT

SOFTWARE ARCHITECTURE

This section describes about how is the software developed for monocle, various events that monocle software should handle and the Internal statemachine.

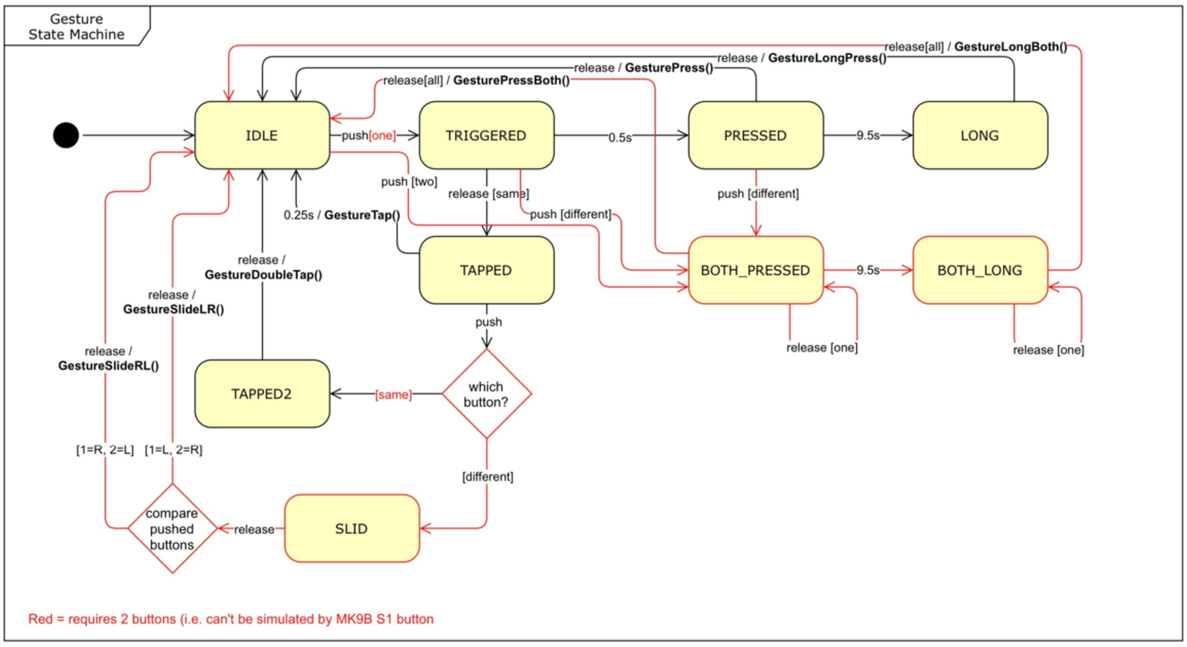
8.1 Monocle Firmware State Machine

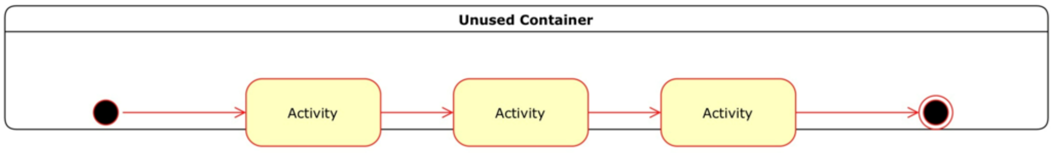


State Descriptions

- EARLY_INIT: Ultra low power mode
- FULL_INIT: Low power mode
- RECORD_D1: Active Recording
- RECORD_D0: Recording Home
- PHOTO_D1: Active Photo
- PHOTO_D0: Photo Home
- REPLAY_D1: Active Photo
- REPLAY_D0: Replay Home
- STANDBY_WARN: Prelude to going into low power state
- STANDBY: Low power
- SHUTDOWN_PEND: Prelude to powered down
- POWERED_DOWN: Ultra low power mode

8.2 Monocle Gesture Events





CHAPTER

NINE

SOFTWARE DESIGN

This section describes design details of Monocle Firmware on the High level architecture discussed in the previous section. This section goes deep in the architecture to understand how each module/component is implemented.

9.1 Data structure

1. Events to be Handled are mentioned below:

```
MONOCLE_EVENT_G_TAP
MONOCLE_EVENT_G_DOUBLE_TAP
MONOCLE_EVENT_G_PRESS
MONOCLE_EVENT_G_LONGPRESS
MONOCLE_EVENT_G_LONGBOTH
MONOCLE_EVENT_TIMEOUT
MONOCLE_EVENT_BATT_LOW
MONOCLE_EVENT_BATT_CHARGE
MONOCLE_EVENT_BLE_CONNECT
MONOCLE_EVENT_BLE_DISCONNECT
MONOCLE_EVENT_BLE_TRANSFER_DONE
```

2. States of Monocle Firmware.

```
MONOCLE_EARLY_INIT
MONOCLE_FULL_INIT
MONOCLE_POWERED_DOWN
MONOCLE_SHUTDOWN_PENDING
MONOCLE_RECORD_D1
MONOCLE_RECORD_D0
MONOCLE_PHOTO_D1
MONOCLE_PHOTO_D0
MONOCLE_REPLAY_D1
MONOCLE_REPLAY_D0
```

3. monocle_state variable in main.c represents the current state of monocle firmware.

9.2 I2C Module Design

One of the interface to achive High Speed Inter Module comminication is via TWO Wire interface such as I2C. In Monocle *MK11* form factor board, there are 2 I2C interfaces. I2C Interfaces are connected to 3 main modules.

- Camera Module (OV5640) Connected on I2C1, Slave Address 0x3C.
- Touch Module (IQS620A) Connected on I2C0, Slace Address 0x44.
- Power Module (MAX77654) Connected on I2C0, Slave Address 0x48.

1. Initializing each I2C instaces are done using the below `init` APIs. This API typically initializes the 2 I2C interfaces appropriately to communicate to the corresponding attached mocules.

APIs for I2C0 and I2C1 respectively:

```
void i2c_init    (void);
void i2c_sw_init (void);
```

- ***Params:***
    - None
- ***Return***
    - None

2. Writing to a module which are connected to I2C Interface are done using the below `write` APIs.

APIs for I2C0 and I2C1 respectively:

```
bool i2c_write    (uint8_t slaveAddress, uint8_t *writeBuffer, uint8_t numberOfBytes);
bool i2c_sw_write (uint8_t slaveAddress, uint8_t *writeBuffer, uint8_t numberOfBytes);
```

- ***Params***
    - slaveAddress -> Unique Address of the Module to be written to.
    - writeBuffer -> Write Data Buffer.
    - numberOfBytes -> Number of bytes to be written.

3. Reading from a mocule which are connected to I2C interface are done using the below `read` APIs.

APIs for I2C0 and I2C1 respectively:

```
bool i2c_read    (uint8_t slaveAddress, uint8_t *readBuffer, uint8_t numberOfBytes);
bool i2c_sw_read (uint8_t slaveAddress, uint8_t *readBuffer, uint8_t numberOfBytes);
```

- ***Params***
    - slaveAddress -> Unique Address of the Module to be read from.
    - readBuffer -> Read Data Buffer.
    - numberOfBytes -> Number of bytes to be read.

9.3 SPI Module Design

The Serial Peripheral interface (SPI) is connected to 3 Modules internally in Monocle MK11 Board. The MCU acts as Master always and the other modules as Slave.

- FPGA connected with Chip Select GPIO PIN 8
- OLED connected with Chip Select GPIO PIN 6
- FLASH connected with Chip Select GPIO PIN 4 (Not accessed in the Firmware)

1. Initializing SPI instace is using the below `init` APIs. After executing this API the SPI Interface should be configured as Master and be ready to communicate to each module.

```
void spi_init (void);
```

- ***Params:***
    - None
- ***Return***
    - None

2. Selecting the CHIP Select PIN. This API is called before calling `read` or `write` to any module.

```
void spi_set_cs_pin (uint8_t cs_pin);
```

- ***Params***
  - cs_pin -> CS PIN Number of the corresponding module connected to SPI Interface.

3. Writing to a module which are connected to SPI Interface is done using `write` API.

```
void spi_write_burst (uint8_t addr, const uint8_t *data, uint16_t length);
```

- ***Params***
  - addr -> Write the Data starting from this Address.
  - data -> Data Buffer.
  - length -> Data Length to be written.

4. Reading to a module which are connected to SPI Interface is done using `read` API

```
uint8_t *spi_read_burst (uint8_t addr, uint16_t length);
```

- ***Params:***
  - addr -> Read the data from this Address.
  - length -> Length of data to be read.
- ***Return***
  - Read Data Buffer.

9.4 ADC Module Design

ADC on Monocle Hardware is used for sampling the Battery Voltage reading. Every 1 Sec the ADC is kick-started to sample the Battery Voltage. The Voltage is averaged for 5 sec or 5 Samples.

There are 2 types of APIs is ADC.

- Quick APIs -> Used only to calculate Battery Voltage quickly in the initial bootup of Monocle Firmware.
- Full APIs -> Used to calculate Battery Voltage in the Normal operation.

1. Init ADC API: To initialize the ADC Sampling intervals with NRF SDK.

```
void adc_init       (void);
void adc_quick_init (void);
```

- ***Params:***
  - None
- ***Return***
  - None

2. Deinitializing ADC APIs: Deinitializing is only required in the initial phase of the Firmware bootup.

```
void adc_quick_uninit(void)
```

- ***Params:***
  - None
- ***Return***

- None

3. ADC Start Sampling: This API is called every Second to check the Battery Voltage

```
void adc_start_sample (void);
```

- ***Params:***
  - None
- ***Return***
  - None

4. API to get the present Battery Voltage.

```
float adc_get_batt_voltage      (void);
float adc_quick_get_batt_voltage(void)
```

- ***Params:***
  - None
- ***Return***
  - ADC Battery Voltage

4. API To get the present Battery Percentage.

```
uint8_t adc_get_batt_soc(void)
```

- ***Params:***
  - None
- ***Return***
  - ADC Battery Percentage

9.5 FPGA Interface Design

The FPGA is interfaced with SPI with the MCU. FPGA Interface modues is the highlevel APIs to get the FPGA LEDs, Mic, Camera and OLED functionality to MCU.

Refer the latest FPGA Register specification.

Following are the APIs defined for the same.

1. **FPGA read / write APIs.** These APIs are interfaced through SPI read/write funtions. Any FPGA register can be read or written using these functions.

```
void fpga_write_byte (uint8_t addr, uint8_t data);
```

- ***Params:***
  - addr -> FPGA Register Address.
  - data -> 1 Byte Data to be written to FPGA Register.
- ***Return***
  - None

```
void fpga_write_burst (const uint8_t *data, uint16_t length);
```

- ***Params:***
  - data -> Data to be written to FPGA Register.

    - Length ->

- ***Return***
    - None

```
uint8_t fpga_read_byte (uint8_t addr);
```

- ***Params:***
    - addr ->
- ***Return***
    - 1 Byte Data read.

```
uint8_t *fpga_read_burst (uint16_t length);
```

- ***Params:***
    - Length ->
- ***Return***
    - Array of data read.

2. **FPGA Control APIs**:

```
void fpga_soft_reset (void);
bool fpga_xclk_on    (void);
```

3. **FPGA Test APIs**:

```
bool fpga_test_reset             (void);
bool fpga_ram_check              (void);
bool fpga_spi_exercise_register (uint8_t addr);
void fpga_set_luma               (bool turn_on);
```

3. **FPGA LED APIs**: On MK9B and MK10 evaluation boards there are physical LEDs that MCU can toggle. On *MK11 Form Factor Board* there are no Physical LEDs.

```
void fpga_led_on      (uint8_t led_number);
void fpga_led_off     (uint8_t led_number);
void fpga_led_toggle  (uint8_t led_number);
void fpga_led_on_all  (void);
void fpga_led_off_all (void);
```

- ***Params:***
    - led_number
- ***Return***
    - None

4. **FPGA Camera APIs**:

```
bool fpga_camera_on     (void);
void fpga_image_capture (void);
void fpga_video_capture (void);
void fpga_set_zoom      (uint8_t level);
```

5. **FPGA Mic APIs**:

```
bool fpga_mic_on          (void);
bool fpga_mic_off         (void);
void fpga_prep_read_audio (void);
```

6. **FPGA OLED APIs**:

```
void fpga_disp_live   (void);
void fpga_disp_busy   (void);
void fpga_disp_bars   (void);
void fpga_disp_off    (void);
void fpga_replay_rate (uint8_t repeat);
void fpga_set_display (uint8_t mode);
```

7. **FPGA Buffer APIs**:

```
uint32_t fpga_get_capture_size (void);
uint32_t fpga_get_bytes_read   (void);
uint16_t fpga_get_checksum     (void);
uint16_t fpga_get_and_clear_checksum(void);
bool fpga_is_buffer_at_start  (void);
bool fpga_is_buffer_read_done (void);
```

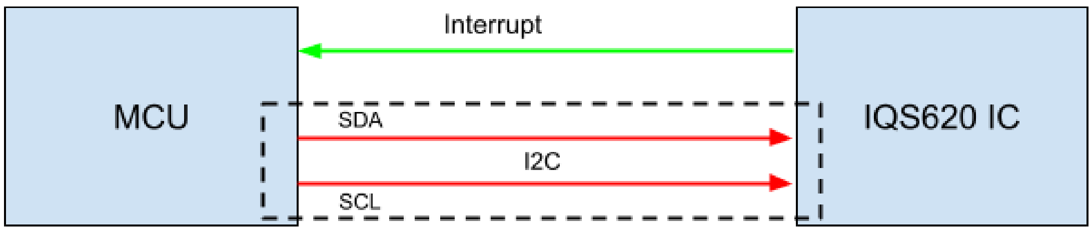
8. **MCU Checksum APIs.**

```
uint16_t fpga_calc_checksum (uint8_t *bytearray, uint32_t length);
uint16_t fpga_checksum_add  (uint16_t checksum1, uint16_t checksum2);
```

9.6 iQS620 Touch Module Design

Touch IC are different on different hardware. On *MK11 Form Factor Board* iQS620 is the touch IC. There are 2 files in focus for the Touch Module design.

- iqs620.c -> defines the low level functions to interface with IQS620 IC using I2C interface. (Not explained in this document)
- touch.c -> defines High level functions to start or stop the TouchIC. (APIs explained below)



MCU to IQS620 connection on the board

The interrupt line (IO_TOUCHED_PIN GPIO Number 2) is the indication to MCU that there is an event in IQS620 IC. Once the Interrupt occurs, MCU should quiry using I2C to get what event was it. There are 2 Buttons on MK11. Both the touch instance are indicated using this IC.

Below are the Events that will be generated.

- TOUCH_GESTURE_TAP : 0.25 Sec
- TOUCH_GESTURE_DOUBLETAP : 0.25 Sec (Both Tapped)
- TOUCH_GESTURE_PRESS : 0.5 Sec
- TOUCH_GESTURE_LONGPRESS : 9.5 Sec
- TOUCH_GESTURE_LONGBOTH : 9.5 Sec (Both Pressed)

Refer *Gesture State machine* to know more about the what actions are taken in each Gesture types.

1. **Touch Init API**: Initializes Touch IC. Quick Init API initializes the Interrupt GPIO. And the Full Init initializes the remaining intialization like statemachine and timer.

```
void touch_quick_init (void);
bool touch_init       (touch_gesture_handler_t handler);
```

- ***Params:***
  - handler -> Callback handler to indicate the Gesture events.
- ***Return***
  - None

2. **Reset function:** To reprogram the IC.

```
bool touch_reprogram (void);
```

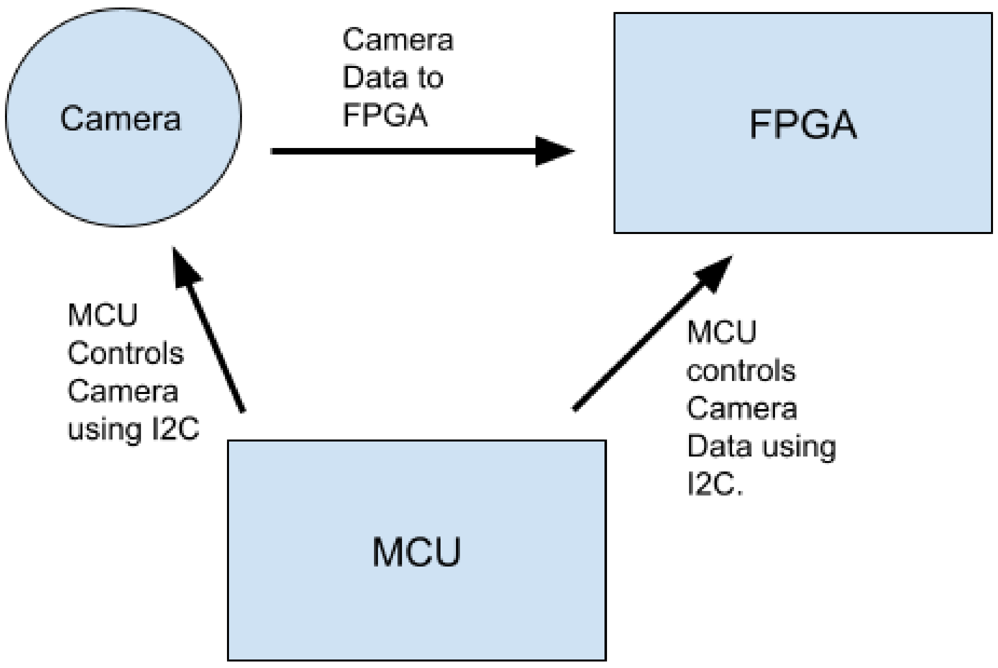
- ***Params:***
  - None
- ***Return***
  - None

3. **Internal ISR and Statemachine functions**

```
static void touch_pin_handler   (void *iqs620, iqs620_button_t button, iqs620_event_t event);
static void touch_event_handler (bool istimer);
```

9.7 OV5640 Camera Module Design

Camera module is connected and controlled by MCU using I2C and the Data path is connected to FPGA. The following image shows the same.


Camera
Camera Data to FPGA
FPGA
MCU Controls Camera using I2C
MCU controls Camera Data using I2C.
MCU

MCU to OV5640 connection on the board

Few important points about Camera module:

- Camera resolution: 5MP
- Records in 15 FPS. While replaying the captured buffer, FPGA replays the same frame thrice to match the 50 FPS rate on OLED.
- Camera Outputs 640X400 resolution frames by 4x Digital Zoom.
- Register configuration of camera is done by MCU.

Interface files:

- ov5640.h -> APIs exposed to Application (main.c).
- ov5640cfg.h -> Internal Config Header files.
- ov5640af.h -> Internal Config Header files.

Camera Bootup APIs:

```
bool ov5640_init      (void);
bool ov5640_pwr_on    (void);
void ov5640_pwr_sleep (void);
void ov5640_pwr_wake  (void);
```

Camera Control APIs:

```
void ov5640_YUV422_mode (void);
void ov5640_mode_1x     (void);
void ov5640_mode_2x     (void);
void ov5640_reduce_size(uint16_t Hpixels, uint16_t Vpixels);
void ov5640_light_mode       (uint8_t mode);
void ov5640_color_saturation (uint8_t sat);
void ov5640_brightness       (uint8_t bright);
void ov5640_contrast         (uint8_t contrast);
void ov5640_sharpness        (uint8_t sharp);
void ov5640_special_effects  (uint8_t eft);
void ov5640_test_pattern     (uint8_t mode);
void ov5640_flash_ctrl       (uint8_t sw);
void ov5640_mirror           (uint8_t on);
void ov5640_flip             (uint8_t on);
bool ov5640_outsize_set  (uint16_t offx, uint16_t offy, uint16_t width, uint16_t height);
bool ov5640_imagewin_set (uint16_t offx, uint16_t offy, uint16_t width, uint16_t height);
bool ov5640_focus_init     (void);
bool ov5640_focus_single   (void);
bool ov5640_focus_constant (void);
```

9.8 Power MAX77654 Module Design

PMIC (Power management IC, External to MCU/SDK), Only used in *MK11 Board*. Right now a little underpowered, we can withness this while running the code on the board, there is a 18 Sec delay after turning the power for every module. Occassionally boot failure is also noticed right now. PMIC is connected to MCU via I2C Interface.

There are at least 5 Different voltage outputs.

- 1.8V Power Rail: MCU and the Touch IC, Always on.

Auxillary Power Rails: All are being used to power other modules.

- 1.2V Power Rail
- 2.7V Power Rail

- 10V Power Rail
- LED Power Rail: Can be tured OFF.

Auxillary Power rails should be tured ON by MCU in a specific routine, which is noted in main.c.

Important files:

- max77654.c -> PMIC Interface API definitions.
- max77654.h -> PMIC Interface APIs.

Note: power.c -> power_management_init function is used to initilize the internal power management using NRF SDK APIs.

Power Init API:

```
bool max77654_init (void);
```

Power Control API:

```
bool max77654_rail_1v8sw_on (bool on);
bool max77654_rail_2v7_on   (bool on);
bool max77654_rail_1v2_on   (bool on);
bool max77654_rail_10v_on   (bool on);
bool max77654_rail_vled_on  (bool on);
bool max77654_led_red_on    (bool on);
bool max77654_led_green_on  (bool on);
max77654_status max77654_charging_status (void);
max77654_fault  max77654_faults_status   (void);
bool max77654_set_charge_current (uint16_t current);
bool max77654_set_charge_voltage (uint16_t voltage);
bool max77654_set_current_limit  (uint16_t current);
```

9.9 OLED Module Design

Sony OLED, is a micro OLED (very small, smaller than a finger nail). Conned directly to the data path to FPGA. MCU is connected to OLED via SPI Interface. Configuration of OLED is done using SPI Interface.

After configuration is done by OLED, luminance should be configured (Setting the brightness).

Important Files:

- oled.c -> Control APIs are defined in this file.
- oled.h -> OLED Interface APIs.

OLED Bootup APIs

```
void oled_pwr_sleep (void);
void oled_pwr_wake  (void);
```

OLED Control APIs

```
void oled_config             (void);
void oled_config_burst       (void);
bool oled_verify_config      (void);
bool oled_verify_config_full (void);
void oled_set_luminance (enum oled_luminance_t level);
```

CHAPTER

TEN

PHONE APPLICATION ARCHITECTURE

This is a React Native app, and the libraries most pertinent to software architecture are the following:

- React Native 0.67.4 https://www.npmjs.com/package/react-native/v/0.67.4
- React Native Navigation 7.26.0 https://www.npmjs.com/package/react-native-navigation/v/7.26.0
- Redux Saga 1.1.3 https://www.npmjs.com/package/redux-saga/v/1.1.3
- React Native SQLite Storage 6.0.1 https://www.npmjs.com/package/react-native-sqlite-storage/v/6.0.1

Coding convention notes:

- Class components are used as opposed to hooks and functional components.
- Promise Resolve is used as opposed to async await.
- Events are processed using Redux Saga.

Please see RNN docs on https://wix.github.io/react-native-navigation/docs/before-you-start for navigation architecture. Navigation entrypoint is https://gitlab.com/brilliant_dave/monocle-phone-app/-/blob/main/src/screens/Navigation.tsx

Database operations are all local, as opposed to API calls to a web backend:

- SQL queries are contained in https://gitlab.com/brilliant_dave/monocle-phone-app/-/blob/main/src/database/MainDao.ts
- MainDao.ts runs queries by calling functions in https://gitlab.com/brilliant_dave/monocle-phone-app/-/blob/main/src/database/DatabaseManager.ts
- Some queries trigger migrations and transactions which are mostly called from DatabaseManager.ts and mainly configured in https://gitlab.com/brilliant_dave/monocle-phone-app/-/blob/main/src/database/migrations/DatabaseMigrationManager.ts

Custom Native Modules:

- Image/audio/video/media assets are processed for compression/decompressing/encoding/decoding (collectively "AssetProcessing") using custom Java and Obective-C code that is linked by React Native's native modules interface.
- AssetProcessing interfaced with React Native via https://gitlab.com/brilliant_dave/monocle-phone-app/-/blob/main/src/natives/AssetProcessing.ts

10.1 Android Application Architecture

See the contents of this directory for related Java code https://gitlab.com/brilliant_dave/monocle-phone-app/-/tree/main/android/app/src/main/java/com/monocle

10.2 iPhone Application Architecture

See the content of this directory https://gitlab.com/brilliant_dave/monocle-phone-app/-/tree/main/ios/AssetProcessing and this directory https://gitlab.com/brilliant_dave/monocle-phone-app/-/tree/main/ios/Encoder

10.3 Phone Application API References

Currently there is no web/cloud API for operations such as checking for Monocle firmware updates. These need to be added.

A custom BLE library is currently being built and is due to be integrated into this app. As such, API details for that can be added to this doc when implementation is complete.

CHAPTER

ELEVEN

DOCUMENTAION TOOLS

Monocle documents are powered by sphinx and readthedocs.org using reStructuredText. This section gives a detailed explaination on how to install tools to generate documentation for Monocle. And also describes how to contribute to the documents.

11.1 First time Install On a Windows system using WSL Ubuntu

- Enable WSL on windows.
- Install Ubuntu 18.04.
- Give appropriate password and configure Ubuntu. By running it.
- APT Update and upgrade on Ubuntu: `$ sudo apt update; sudo apt upgrade`
- Install python on Ubuntu: `$ sudo apt install python python-pip`
- Change directory to `monocle` folder: `$ cd /mnt/c/Nordic/nRF5_SDK_17.0.2_d674dde/examples/monocle`
- Create a Python Virtual Environment for Sphinx: `$ python -m virtualenv monocle_env`
- Activate Python Virtual Environment: `$ source monocle_env/bin/activate`
- Install Sphinx: `$ pip install sphinx`
- Install ReadTheDocs theme: `$ pip install sphinx_rtd_theme`
- Change Directory to docs: `$ cd docs`
- Install PDF Tools: `$ sudo apt-get install latexmk texlive-latex-recommended texlive-latex-extra texlive-fonts-recommended`

11.2 Generate documents

- Build the document to get HTML: `$ make html`
- Build the document to get PDF: `$ make latexpdf`
- The documents will be built on `build` directory under `docs` folder.

11.3 Updating the document and rebuilding

- All the document source code is present in `source` folder.
- `index.rst` file is the home file. The source files are written in ReStructuredText format.
- After modifying the source code (Document source code), execute `$ make html` to generate the html documents.

CHAPTER

TWELVE

CURRENT RELEASE

12.1 Monocle Fimrware Version V1.6

Hardware	MK11 V1.0
MCU	MK11_MCU_V1.6.hex
FPGA	MK11_FPGA_V0.12.fs

12.2 New feature implementaion details

- BLE Stack update to SWL BLE Release 2022-05-09
- 5 Sec Live Video change
- Camera code made to be a Lib (support_lib.a)

12.3 Bugs Solved

- Factory Mode implemented (Removed OLED notification during Battery Low condition, instead added LED Toggle)
- 1 Sec Delay instead of 18 Sec bootup delay

12.4 TAG Version

v1.6

12.5 Milestone

Release v1.6

12.6 New features in the next Release

- Code Structuring
- Implement Scheduler
- Update FPGA Support

12.7 Dependancies

- Swaralink BLE Library Release 2022-03-16-A
- nRF SDK 17.0.2

CHAPTER

THIRTEEN

PREVIOUS RELEASES

13.1 Monocle Fimrware Version V1.4

Hardware	MK11 V1.0
MCU	MK11_MCU_V1.4.hex
FPGA	MK11_FPGA_V0.12.fs

13.2 New feature implementaion details

- Flip image to correct optics
- Increase battery termination current cutoff from 5% to 10% (6.8mA) to eliminate CC6 green LED flicker
- Prevent reboot loops by requiring battery be at >20% SoC for boot to proceed
- OTA Protocol change - Added Media Header
- Update LED Blink Codes for Production.

13.3 Bugs Solved

- Bluetooth stack bugfix (allow factory reset before pairing)
- Upgrade Bluetooth stack (note: breaks compatibility with previous releases)
- Standby after 5 min or Battery Low

13.4 TAG Version

v1.4

13.5 Milestone

Release v1.4

13.6 New features in the next Release

- Code Structuring
- Modularize
- Implement Scheduler

13.7 Dependancies

- Swaralink BLE Library Release 2022-03-16-A
- nRF SDK 17.0.2

CHAPTER

FOURTEEN

MONOCLE PHONE APP VERSION 0.0.3

Build files are not included as part of the repository. They will need to be built in each of Android Studio for Android, and XCode for iOS. Actual releases are currently made available for internal testers on the Google Play Store and Apple App Store developer accounts.

14.1 New feature implementation details

No new features.

14.2 Bugs Solved

- Fixed a major crash on iOS
- Image and video media now displays on Android API 32

14.3 TAG Version

v0.0.3

14.4 Milestone

Release v0.0.3

14.5 Bugs to be resolved in the next Release

- Audio/video playback on iOS needs to improve.
- Image and video media doesn't always display on Android API version lower than 32
- Image and video display dimensions do not position and resize correctly on Android devices that have a larger screen.

14.6 Dependancies

Please see https://gitlab.com/brilliant_dave/monocle-phone-app/-/blob/main/package.json