|  |  |
| --- | --- |
|  | Spectrometer Functional Specification |
| PFS00001 1.2 | By  Rajesh  for  Ir2Labs |

# Approvals

|  |  |
| --- | --- |
| Approval | Job Responsibility /Title |
| Rajesh | System Architect |
| Balakrishnan | Project Manager |

Table of Contents

[1. Approvals 2](#_Toc495607694)

[2. Introduction 4](#_Toc495607695)

[2.1. Purpose 4](#_Toc495607696)

[2.2. Document Scope 4](#_Toc495607697)

[2.3. Intended Audience and Reading Suggestions 4](#_Toc495607698)

[2.4. Firmware Functions 4](#_Toc495607699)

[3. Functionality 5](#_Toc495607700)

[3.1. System Overview 5](#_Toc495607701)

[3.2. Boot up 6](#_Toc495607702)

[3.3. Wi-Fi configuration 6](#_Toc495607703)

[3.4. IP address and Port number 6](#_Toc495607704)

[4. Protocol Command/Response formats 7](#_Toc495607705)

[4.1. WiFi access point configuration 7](#_Toc495607706)

[4.2. Capture Full Frame 7](#_Toc495607707)

[4.3. Exposure level configuration 7](#_Toc495607708)

[4.4. Set averaging frame count 8](#_Toc495607709)

[4.5. Set Ultra violet LED State 8](#_Toc495607710)

[4.6. Set White LED State 8](#_Toc495607711)

[4.7. ROI configuration 8](#_Toc495607712)

[4.8. Capture Cropped Region 9](#_Toc495607713)

[4.9. Capture Averaged Line 9](#_Toc495607714)

[4.10. Move strip n position command 9](#_Toc495607715)

[4.11. Save ‘A’ value command 10](#_Toc495607716)

[4.12. Get ‘A’ value command 10](#_Toc495607717)

[4.13. Save ‘B’ value command 10](#_Toc495607718)

[4.14. Get ‘B’ value command 10](#_Toc495607719)

[4.15. Reset switch 10](#_Toc495607720)

[4.16. Error Handling 10](#_Toc495607721)

[5. System Design 11](#_Toc495607722)

[5.1. Over all Application flow 11](#_Toc495607723)

[5.2. Image capture design 12](#_Toc495607724)

[5.2.1. Capture full frame 12](#_Toc495607725)

[5.2.2. Line average calculation 12](#_Toc495607726)

[5.3. Wi-Fi System design 13](#_Toc495607727)

[5.3.1. TCP connection 14](#_Toc495607728)

[5.3.2. spi\_recv\_task 14](#_Toc495607729)

[5.3.3. spi\_send\_task 15](#_Toc495607730)

[5.4. Motor system design 15](#_Toc495607731)

[6. Limitations 16](#_Toc495607732)

[6.1. General 16](#_Toc495607733)

[7. Document 17](#_Toc495607734)

[7.1. Definitions, Acronyms and Abbreviations 17](#_Toc495607735)

[7.2. References 17](#_Toc495607736)

[7.3. Revision History 17](#_Toc495607737)

[8. Appendix A - Requirement List 18](#_Toc495607738)

# Introduction

## Purpose

The purpose of this document is to provide a functional and design specification for the spectrometer based on STM32 and AR0130 image sensor.

## Document Scope

This document covers all the functionalities related to the spectrometer and configuration of the image ROI.

## Intended Audience and Reading Suggestions

The intended readers of this document are marketing people, firmware and application engineers and technical writers. This document can also form the basis for creating test cases for Test Engineers.

## Firmware Functions

List the main functions delivered from this specification.

* Wi-Fi Access Point configuration and connectivity
* TCP/IP server port definition and command/reply formats
* Application Command list and reply formats for
  + Camera configuration commands
  + Region of Interest (ROI) configuration commands
  + Position sensor and Motor driver commands
  + Access point configuration commands
* System RESET command

# Functionality

## System Overview

The overall design of spectrometer is as shown in Figure 1.

The Spectrometer consist of a CMOS camera, connected to a microcontroller via parallel port. The MCU is connected

The spectrometer reads the predefined area of the image projected by the sample. The region of interest (ROI) is set initially and the ROI of the image of the sample is read and averaged at the configured sampling rate and sent to the mobile/Desktop Application requesting it. The main features are listed below

* The Mobile/PC application connects to the spectrometer via Wi-Fi or USB Virtual COM Port
* ROI is configured via commands explained in the protocol section
* The spectrometer collects the ROI, averages it and streams it to the application requesting it
* A strip with samples moves so as to focus the camera on the sample (via motor driver and position sensor)
* The application plots the data and deduces the results as per the requirement

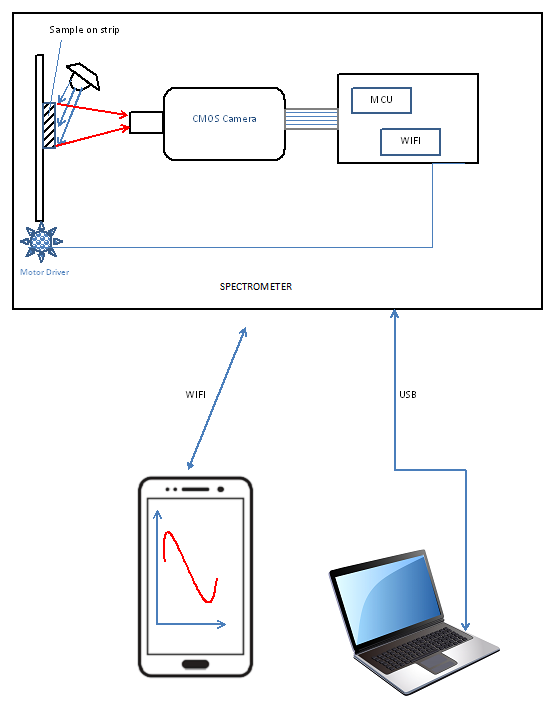


Figure 1: Overall System Architecture

## Boot up

After Power UP or RESET the board takes 1 second to initialize all peripherals and is ready to accept commands.

## Wi-Fi configuration

After Power up, the spectrometer act as a Wi-Fi access point, and any Wi-Fi client or mobile application can connect using to the board by selecting the “Spectrometer” SSID and “12345678” as the password.

## IP address and Port number

Once the Wi-Fi connection is established, the TCP server waits at port 333 at IP address 192.168.4.1. The TCP client application connects to this server.

# Protocol Command/Response formats

The spectrometer acts as a USB device when connected to a PC and also hosts a TCP server for all the configuration, control and data gathering.

The configuration includes Wi-Fi SSID and password change, configure camera ROI, stream data and move the strip based on position sensor interrupt.

All commands will have a ‘$’ as the start and ‘#’ at the end of command.

## WiFi access point configuration

$APCxxxxxxxxxxxxPWDyyyyyyyyyyyy!

Where “xxxxxxx” is the SSID of max 12 char length and “yyyyyyyyyyyy” is the password of max 12 char length.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If SSID is SpecSSID and Password is 12345678

Then send

$APCSpecSSIDPWD12345678!

## Capture Full Frame

$CFF#

Upon receiving this command a frame of size 1280 \* 960 is captured for 0 vertical and horizontal offset and sent as response.

Upon Success Response returns: ^ssss#dddddddddddd…….ddddd^EOF#

Where sssss is the size in bytes of frame

dddd….is the frame date

else

“$ERR#”

Example:

$CFF# will return

^120#yy ¡‹ŸŠ¤Œ¬|§‹‰¨‚£–Ÿ€•£y¨™ˆ£Š’£ƒŸŠž€¦†¦‚•‰£‚¡”¨ˆ˜‹ Š›”§©^EOF#

## Exposure level configuration

$ELCxxxxx#

Where ‘xxxxx’ is the exposure level in decimal format. Note max value 65535

Upon Successful Response returns: “$OK#”

else

“$ERR#”

## Digital Gain level configuration

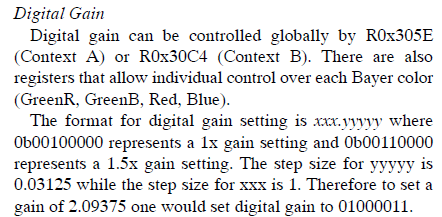
$GNVxxxx#

Where ‘xxxx’ is the exposure level in decimal format.

Upon Successful Response returns: “$OK#”

else

“$ERR#”



## Analog Gain level configuration

$AGN1X#

$AGN2X#

$AGN4X#

$AGN8X#

The above are individual commands for 1X,2X,4X,8X gain setting.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

## Global Analog Gain level multiplier

$AGN1.25E#

Enables 1.25 multiplication factor to the set analog gain.

$AGN1.25D#

Disables 1.25 multiplication factor to the set analog gain.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

## Set averaging frame count

$AFCxx#

Where ‘xx’ is the averaging count in hexadecimal format.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Minimum count is 1 and maximum is 800 frames.

Example:

If you want to average 40 frames then send

$AFC40#

## Set Ultra violet LED State

$SUV1# for ON or $SUV0# for OFF

This command switches ON or OFF Ultra violet.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

## Set White LED State

$SWL1# for ON or $SWL0# for OFF

This command switches ON or OFF white led.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

## Set Reflection LED State

$RLD1# for ON or $RLD0# for OFF

This command switches ON or OFF white led.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

## Set UV LED intensity

$IUVxxx#

This command gives 0 to 100% intensity.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example

To set to 100 % intensity send the command,

$IUV100#

## Set WL LED intensity

$IWLxxx#

This command gives 0 to 100% intensity.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example

To set to 100 % intensity send the command,

$IWL100#

## Set Reflection LED intensity

$IRLxxx#

This command gives 0 to 100% intensity.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example

To set to 100 % intensity send the command,

$IRL100#

## ROI configuration

$ROIho,hc,vo,lc#

Where ROI parameters are separated by,

ho = Horizontal offset

hc = Horizontal pixel count

vo = vertical offset

lc = number of lines to be captured

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Note:

1. The horizontal pixel offset shall not exceed 1280 (ho + hc)
2. The vertical line offset shall not exceed 960 (vo + lc)
3. The line count has to be an even number and total pixel count (hc \* lc) should not exceed 25600.

Example:

1. If you want to set ROI at 20 horizontal offset, 1000 horizontal pixels, at 400 vertical offset and capture 20 lines then send

$ROI20,1000,400,20#

1. If you want to set ROI at 40 horizontal offset, 800 horizontal pixels, at 370 vertical offset and capture 14 lines then send

$ROI40,800,370,14#

## Capture Cropped Region

$CCR#

Upon receiving this command a frame is captured as per the ROI sent as response.

Upon Success Response returns: ^ssss#dddddddddddd…….ddddd^EOF#

Where sssss is the size in bytes of frame

dddd….is the frame date

else

“$ERR#”

Example:

$CCR# will return

^120#yy ¡‹ŸŠ¤Œ¬|§‹‰¨‚£–Ÿ€•£y¨™ˆ£Š’£ƒŸŠž€¦†¦‚•‰£‚¡”¨ˆ˜‹ Š›”§©^EOF#

## Capture Averaged Line

$CAL#

Upon receiving this command, N frames are captured (N is set via $AFCxx# command) as per the ROI and then all frames are averaged. Then a vertical average of all the lines in the ROI is calculated and sent as response.

Upon Success Response returns: ^ssss#dddddddddddd…….ddddd^EOF#

Where sssss is the size in bytes of averaged line data

dddd….is the averaged line data

else

“$ERR#”

Example:

$CAL# will return

^120#yy ¡‹ŸŠ¤Œ¬|§‹‰¨‚£–Ÿ€•£y¨™ˆ£Š’£ƒŸŠž€¦†¦‚•‰£‚¡”¨ˆ˜‹ Š›”§©^EOF#

## Move strip n position command

$MSPxx#

The strip moves xx sample position or the the motor is rotated for 10 seconds and stopped

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If you want to move the strip 10 samples ahead, then send

$MSP10#

## Move strip n steps command in clock wise direction

$MRSxxx#

The strip moves xx steps position in the clock wise direction

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If you want to move the strip 999 steps in clock wise direction, then send

$MRS999#

## Move strip n steps command in counter clock wise direction

$MLSxxx#

The strip moves xx steps position in the counter clock wise direction

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If you want to move the strip 999 steps in counter clock wise direction, then send

$MLS999#

## Move strip with “start command” in clock wise direction and return the steps moved once the “stop command” is issued.

$MSR1#

The strip starts to move in clock wise direction till command $MSR0# is issued.

Upon Successful Response returns: “$no of steps#”

else

“$ERR#”

Example:

If you want to start the strip in clock wise direction issue command

$MSR1#   
here 1 indicated start.   
  
If you want to Stop the strip issue command   
$MSR0#   
here 0 indicated stop.

## Motor Speed 0 to 100%

$MSSxxx#

The strip speed can be controlled with this command from 0 % to 100% full speed.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If you want to move the strip in full speed, then send

$MSS100#

## Set Macro Item

$MASxx,yy,xx#

Used this command to set macro item

|  |  |  |
| --- | --- | --- |
| xx | integer | Macro step |
| yy | integer | Macro item type(ref. Below table) |
| zz | integer | Macro item value |

|  |  |  |  |
| --- | --- | --- | --- |
| Item type | Type name | Item value | Description |
| 0 | MSP | n | Same as MSP command |
| 1 | MRS | n | Same as MRS command |
| 2 | MLS | n | Same as MLS command |
| 3 | SWL | 0 or 1 | Same as SWL command |
| 4 | DLY | n | Delay n ms |
| 5 | CAL | n | Capture frame and set as index n |
| 6 | DIR | n | Same as DIR command |

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If you want to add index 0 and “MSP” macro item, then send

$MAS0,0,1#

If you want to add index 1,“MRS” and 1000 steps macro item, then send

$MAS1,1,1000#

## Set Macro Item Count

$MACxx#

Used this command to set macro item count.

After all macro item are set, still need to sent this command to tell machine the macro item count.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

If you want to set macro item count as 5, then send

$MAC5#

## Execute Macro

$EXE#

Used this command to execute macro.

During macro execute period, after each macro execute device will output macro result

$MAR,xx,yy#

|  |  |  |
| --- | --- | --- |
| xx | integer | Macro step |
| yy | integer | Macro result(0=successful, 1=error) |

After all macro are done

Upon Successful Response returns:

“$OK#”

else

“$ERR#”

## Get result of CAL

$GETxx#

Used this command to get CAL result, xx is the index of calculate result.

Upon receiving this command, N frames are captured (N is set via $AFCxx# command) as per the ROI and then all frames are averaged. Then a vertical average of all the lines in the ROI is calculated and sent as response.

Upon Success Response returns: ^ssss#dddddddddddd…….ddddd^EOF#

Where sssss is the size in bytes of averaged line data

dddd….is the averaged line data

else

“$ERR#”

Example:

$CAL# will return

^120#yy ¡‹ŸŠ¤Œ¬|§‹‰¨‚£–Ÿ€•£y¨™ˆ£Š’£ƒŸŠž€¦†¦‚•‰£‚¡”¨ˆ˜‹ Š›”§©^EOF#

## Start JSON Macro

$JON#

Used this command to issue macro execute from macro.json.

During macro execute period, after each macro execute device will output macro result

$MAR,xx,yy#

|  |  |  |
| --- | --- | --- |
| xx | integer | Macro step |
| yy | integer | Macro result(0=successful, 1=error) |

After all macro are done

Upon Successful Response returns:

“$OK#”

else

“$ERR#”

Example:

If you want to set macro item count as 5, then send

$MAC5#

## Set motor rotate direction

$DIRxx#

Used this command to set motor direction, xx is the motor direction, it accept 0 or 1.

Upon Successful Response returns: “$OK#”

else

“$ERR#”

Example:

$DIR1#

## Get position sensor status

$QPSxx#

Used this command to get status of position, xx is the identity of position sensor.

Upon received this command, machine will polling status of position by xx, and return the status to host.

Output “$ON#” or “$OFF#”.

Example:

“$QPS1” will return

$OFF#

## Get value of ELC command

$ELC?#

Used this command to get value of command ELC.

Upon Successful Response returns: “$ELCxx#”

Example:

$ELC?# will return

$ELC5000#

## Get value of GNV command

$GNV?#

Used this command to get value of command GNV.

Upon Successful Response returns: “$GNVxx#”

Example:

$GNV?# will return

$GNV32#

## Get value of AGN command

$AGN?#

Used this command to get value of command AGN.

Upon Successful Response returns: “$AGNxx#”

Example:

$AGN?# will return

$AGN1X#

## Get value of ROI command

$ROI?#

Used this command to get value of command ROI.

Upon Successful Response returns: “$ROIho,hc,vo,lc#”

Example:

$ROI?# will return

$ROI0,1280,480,20#

## Get value of AFC command

$AFC?#

Used this command to get value of command AFC.

Upon Successful Response returns: “$AFCxx#”

Example:

$AFC?# will return

$AFC1#

## Get value of DIR command

$DIR?#

Used this command to get value of command DIR.

Upon Successful Response returns: “$DIRxx#”

Else

“$ERR#”

Example:

$SAV# will return

$OK#

## Get firmware version

$VER#

Used this command to get firmware version of device.

Upon Successful Response returns: “$VERxx.yy#”

Example:

$VER# will return

$VER1.18#

## Save current setting into config.json

$SAV#

Used this command to save current setting into config.json file in SPI flash.

Upon Successful Response returns: “$OK#”

Example:

$AGN?# will return

$AGN1X#