TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Thomas Johann Seebecki elektroonikainstituut

Aleksandr Stennikov

MT9P031 baasil kaamera

Bakalaureusetöö

Juhendaja: Argo Kasemaa, doktorikraad, lektor

Tallinn 2015

Autorideklaratsioon

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus ning kinnitan, et esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud. Kinnitan, et antud töö koostamisel on kõikide teiste autorite seisukohtadele, probleemipüstitustele, kogutud arvandmetele jmt viidatud.

Kuupäev:

Autor: Aleksandr Stennikov ……………………….

/allkiri/

Lõputöö ülesanne

Lõputöö teema: MT9P031 baasil kaamera

Teema päritolu: Mind huvitab kaamerate ülesehitus.

Lõputöö eesmärgid: Uurida pildi tegimise protsessi alustades pildisensorist sensorist ja lõpetades pildifailiga arvutis.

Oodatavad tulemused: Valmistada PCB-t mille abil saab teha pilte.

Lahendatavad küsimused: Valguse teisendamine elektrisignaaliks. Kiirete signaalide vastuvõtt, töötlus ja salvestamine. Andmete saatmine arvutisse.

Kuupäev: ……………………

Üliõpilane: ……………………
/allkiri/

Juhendaja: ……………………
/allkiri/

Kinnitaja: ……………………
/allkiri/

Abstract

This work describes process of digital camera development. The goal is to test image acquisition process starting from light capture and finishing by getting image files on computer. MT9P031 sensor was chosen and based on its outputs other components were established. They were STM32F439 microcontroller that received raw data from sensor and processed it and 8 megabyte SDRAM memory where images were stored. Communication with computer was achieved through USB-to-serial converter. PCB was developed and produced. Image acquisition can be started by pressing snapshot button or by sending command from computer. Python script was written to automate this process. During this work author learned the basics of image sensor operation, image processing and data acquisition. This knowledge will allow creating new, more complicated cameras.

The thesis is in English and contains **29** pages of text, **3** chapters, **21** figures and **5** tables.

Resümee

See töö kirjeldab digitaalse kaamera arendusprotsessi. Eesmärgiks on pildi tegimise protsessi testimine alustades pildisensorist ja lõpetades pildifailiga arvutis. MT9P031 sensor oli valitud ja selle väljundite basil ülejäänud kaamera komponendid olid paigaldatud. Nad on STM32F439 mikrokontroller mis võtab vastu info sensorist ja töötleb seda ja 8 megabaitiline SDRAM mälu kus info on salvestatud. Andmevahetus arvutiga käib läbi USB-to-serial adapteri. Pildi tegemine algab kui kasutaja vajutab snapshot nuppu või kui saadatakse käsk arvutist. Oli kirjutatud script python-is mis automatiseeiub pildi tegemise protsessi. PCB oli arendatud ja toodetud. Töö käigus sai autor põhiteadmisi sensorite tööpõhimõttest, pilditöötlusest ja andmesalvestamisest. Omandatud teadmised lubavad arendada keerulisemad kaamerad tulevikus.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti **29** leheküljel, **3** peatükki, **21** joonist ja **5** tabelit.

Lühendite ja mõistete sõnastik

ADC – *Analog-to-Digital Converter*

ARM – *Acorn RISC Machine*, processor architecture

ASCII – *American Standard Code for Information Interchange*

CCD – *Charge-Coupled Device*

DCMI – *Digital Camera Interface*

DMA – *Direct Memory Access*

DOF – *Depth of Field*

DRAM – *Dynamic Random Access Memory*

ESD – *Electrostatic Discharge*

FMC – *Flexible Memory Controller*

FPGA – *Field Programmable Gate Array*

GPIO – *General Purpose Input/Output*

I2C – *Inter-Integrated circuit*

JPEG – *Joint Photographic Experts Group*

LCD – *Liquid-Crystal Display*

LED – *Light-Emitting Diode*

LQFP – *Low-profile Quad Flat Package*

LVTTL –*Low Voltage Transistor-Transistor Logic*

NMOS – *N-channel MOSFET*

PCB – *Printed Circuit Board*

PLL – *Phase-Locked Loop*

PNG – *Portable Network Graphics*

RGB – *Red, Green, Blue*

SDRAM – *Synchronous Dynamic Random Access Memory*

SOT – *Small-Outline Transistor*

SRAM – *Static Random Access Memory*

STM - *STMicroelectronics*

SWD – *Serial Wire Debug*

TIFF – *Tagged Image File Format*

UART – *Universal Asynchronous Receiver/Transmitter*

USB – *Universal Serial Bus*

Jooniste ja tabelite nimekiri

Table of contents

Introduction

Digital imaging has seen an explosive growth in last decades. It all started in early twenties when first cathode ray tubes appeared. They were clumsy and power-hungry and their use was limited. It was not until the late XX century when first semiconductor image sensors appeared. They were first expensive but year after year their possibilities grew, they became cheaper, more compact and the image quality increased dramatically. Now semiconductor sensors are the most popular way of making images.

Imaging has a wide variety of applications. We all have seen cameras in our mobile phones but it is just the tip of the iceberg. Cameras are used in medical applications, security and supervision, remote sensing, process automation, machine vision and art. Every application has different requirements for camera hardware. While artistic photography may need larger photosensitive area for higher DOF machine vision will almost certainly need smaller sensor to make all objects in frame look sharp.

There are many ways to produce an image. Generally you need a data source, which can even be a mathematical function that can provide data that can be depicted as two-dimensional array. In this document word image or frame means two-dimensional array that depicts a visual perception. Camera is an electronic device that uses photo sensor to capture photons and convert them to electrical signal. This signal is then processed and stored in memory as image file.

The goal of this project is to create a device that mimics digital camera operation. User has to press a button and single digital image will be produced. Creating such device requires solving many different problems such as conversion of light into signal, transferring large amounts of data between electronic components fast, image manipulation, compression, processing and also data transfer to computer. Making such device will expand authors understanding of topic and will give a solid foundation for future projects.

Thesis consists of several chapters. “Architecture” gives a brief overview of entire system and processes and describes how main components were chosen and how they interface each other. “PCB Development” describes component selection, schematic capture, PCB layout and electrical testing of manufactured PCB. “Software” describes how software works both on PCB and on computer and how images are created, processed and transferred.

1. Architecture

1.1 Overview

Making an image means transposing three-dimensional scene on a two-dimensional surface. Images are usually done by cameras. Imaging process starts when light reflected from scene enters lens and is focused on a photosensitive area, or sensor. Sensor then converts light into electric signal. Electrical signal can then be displayed on some media such as screen.

Now we will outline essential components that are needed to make camera work. This project focuses on electronics and software mainly, so optics, lightning and camera body will be omitted. Image capture starts with a sensor. Sensor captures light and converts it to signal. Images can be displayed on a computer screen but before that we need to convert signal from image sensor to displayable format. For that, we need some sort of processor. Since image has to be captured fast to allow handheld operation it is expected that sensor will generate large amount of data, much more than communication protocol will handle. That implies having some sort of buffer that stores data from sensor before it is fully transmitted.

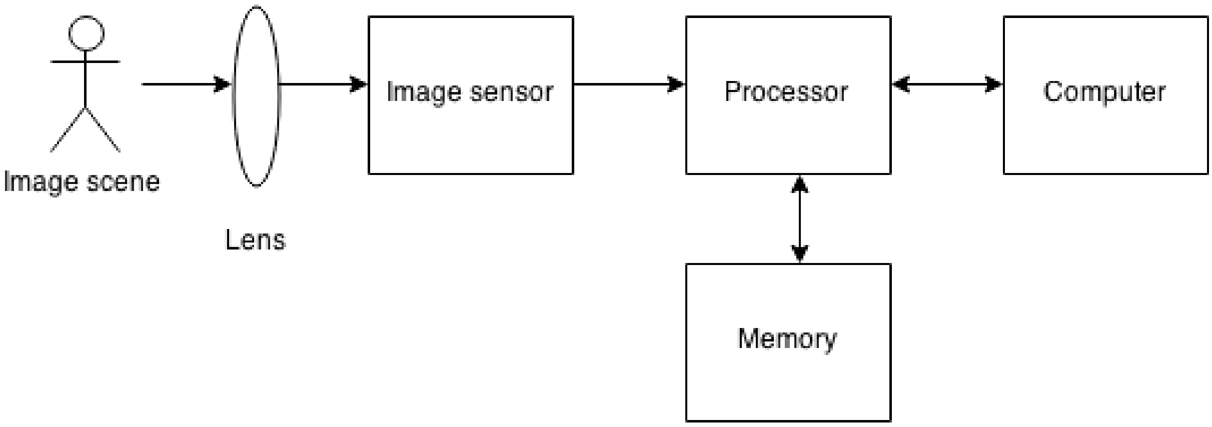


Illustration 1: Main components participating in image creation

The design process will begin by finding a suitable sensor. Its task is to produce images. Sensor interface will be analyzed and appropriate processor will be selected. Processor's task is to get data from sensor, process it into displayable format and send to computer. Memory will be required as a buffer to store rapidly incoming data from sensor. All components require electricity for operation so power consumption will be calculated. Finally an imaging sequence will be described when main components will be established.

1.2 Image sensor

A heart of a camera is its sensor. There are many sensors commercially available. We will now produce a set of requirements that will outline our specific needs and will help in sensor selection.

1. Ability to capture entire frame in a short time. This excludes a wide selection of single pixel and line scan sensors. Pressing a button must produce image instantly without requiring camera body movement to get entire frame.
2. Colour frames and visible spectrum. Single pixels only indicate level of charge of a single photocell and have no colour information. Sensors have colour filters in front of every pixel that allow only one specific frequency to pass.
3. Sensor should have a meaningful area both in pixels and physically. Pixel count influences image resolution and level of detail. One or two megapixels will be enough. Larger physical area will allow more light to get to sensor which in turn will reduce shutter speeds and improve low-light performance.
4. Electronic shutter. Pixel exposure timing must be regulated electronically without complex shutter mechanisms.
5. Communication interface, not very complex and capable of transmitting pictures at reasonable speeds.
6. Availability of electronic components in small quantities and at reasonable price.

Image sensors can be divided into 3 big categories: single pixel, line or area. Single pixel sensor has only one light-sensitive element. Although it is possible to move such sensor around to cover some area over time it is not practical in our case. Line sensor consists of only one line of pixels. Such sensors are often found in barcode scanners, computer mice or satellites. We need to capture two-dimensional image in short amount of time and such sensor does not suit well. A good solution will be area sensor.

Colour sensor means that it will capture images in visible spectrum. Single photocells can provide only luminosity information. To get colour filters must be used. There are two approaches: placing filters in front of lens and then combining several pictures taken with different filters or placing filters directly in front of photocells. Second approach allows making images faster and will be chosen in this project. Filters allow only one frequency to pass. When red, green and blue filters are used we can get intensity of these 3 spectrums. When combined they can produce colour image.

Typical image sensor uses photoelectric effect to convert photons into electric charge. Normally sensor is built from one piece of silicon and charges from several detectors would stick together. This is prevented by charge coupled devices (CCD) which use electric potentials to attract charges and hold them in one place. Charges are then shifted out to output stage which samples potential and converts it to analog signal. Many sensors have integrated circuits which further process this signal to make device easily interfaceable with typical processors.

One known sensor manufacturer is Kodak. Their sensor business was sold some time ago to Truesense Imaging and later to ON Semiconductor. ON Semiconductor has very nice sensor documentation [1], where all aspects of sensor operation are described. KAI−02150 is good example. Let’s check its datasheet [2]. Sensor is divided into 4 quadrants each having its own control signals. Every quadrant has a set of horizontal and vertical clock signals and an analog output. Here is an image illustrating image readout process:

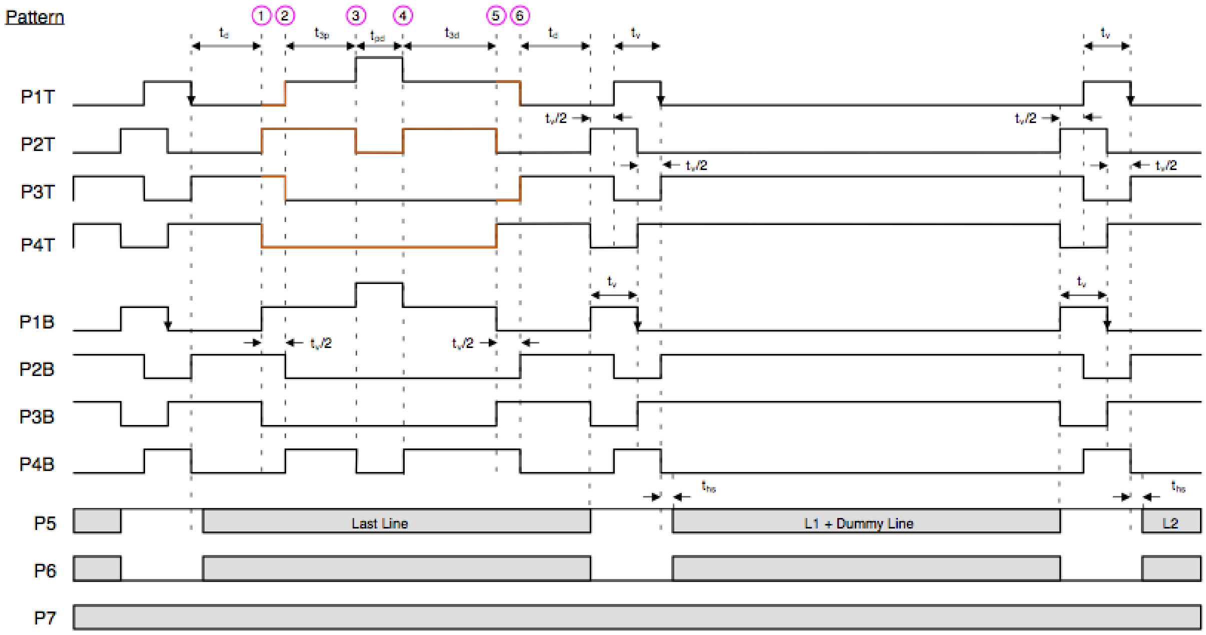


Illustration 2: KAI-02150 image readout signals [2]

External controller has to drive CCD phases to read out accumulated charges. Analog signal then has to be read out by very fast ADC. That implies a very low-level operation. It would require generating lots of clocks signals with very precise timing. This is probably best done by FPGA or some dedicated image processor and the complexity level is higher than this project originally suggests.

Another well known manufacturer is Aptina. They have a wide range of colour and monochrome sensors aimed at surveillance and mobile markets. MT9P031 looks promising. It is 5.70 by 4.28mm 5MP 12 bit colour image sensor. Here is its block diagram:

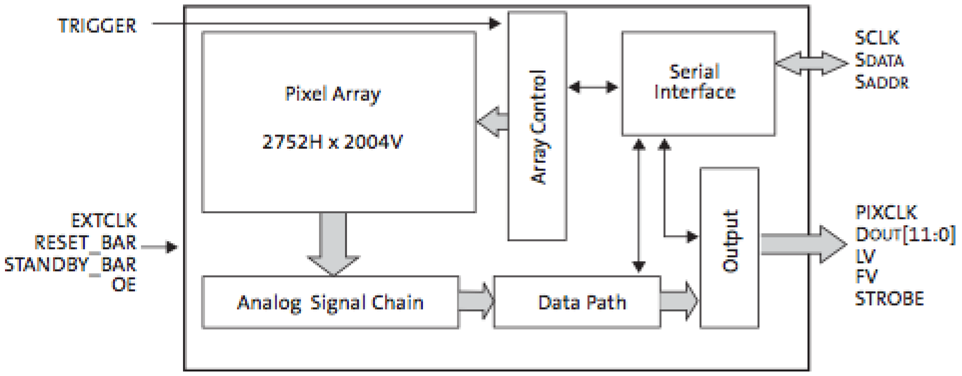


Illustration 3: MT9P031 block diagram [3]

This sensor has CCD control and ADC built-in. Image parameters can be set up via I2C bus. Image capture is initiated by pulling TRIGGER pin low. Image data is transferred via 12 bit CMOS-compatible parallel interface. Every pixel should be read in at falling edge of PIXCLK. Data reception is synchronized by LINE_VALID and FRAME_VALID signals. Sensor's frequency range is between 96 and 6MHz. Sensor requires 2 voltage rails: 2.8V for IO and internal analog circuits and 1.8V for internal core logic. 1 rail would be better of course, but 2 are tolerable. Datasheet specifies that absolute maximum voltage is 3.1V so other components on board should use 2.8V as well to allow direct interfacing.

1.3 Processor

Processor will receive image from sensor, save it to memory, perform basic processing and transmission to PC. For that, we need several interfaces: camera interface, memory interface, UART for transmission and I2C for sensor communication.

Since I have some prior experience with STM32 line one of the STM microcontrollers will be chosen. Let's research their most advanced F4 line. This line features 32-bit ARM Cortex M4 microcontrollers. They are quite expensive and power hungry and have lots of peripherals. Price and power consumption is not of prime concern in this project so we can afford using most advanced chips. STM32F4x9 series has:

- Digital camera interface, exactly the same as in our sensor
- Several parallel memory interfaces
- Several communication peripherals such as UART or I2C
- 180MHz operation which should be enough to handle data stream
- LCD controller, in case we want to add LCD screen some time later
- 1.8V to 3.3V operation. Can use same supply as sensor.
- LQFP package which means easy soldering

To support development STMicroelectronics produces development kits for its every major microcontroller. These feature not only microcontroller but also other devices such as gyroscopes that can be communicated with. STM32F429Discovery kit has external SDRAM memory on it and that will definitely be helpful.

1.4 Memory

Image sensor generates much more data than processor can store. Data also comes in faster than it can be transmitted to computer. Thus we have to select memory capable of storing images as they arrive.

First choice is between volatile and non-volatile memory. Most common solution for non-volatile memory is flash memory. Flash memory is low-cost and offers large memory sizes in relatively compact packages. Flash memory is non-volatile which means data will be saved even after power down. Since our plan is to transmit images to PC right after frame capture, permanent storage is not necessary. Moreover, overwriting data in flash memories is not very comfortable process because entire page has to be erased before new data can be written, arousing need for buffers and so on. Random access memory would be much better for that. It is also important for memory to have a parallel interface. That will allow for much faster data transfer rates. A DRAM is preferred to SRAM because of higher density and lower cost. A great idea would be using a memory chip from existing project. That will simplify development because we could use existing schematics, layout and code for reference. An STM32F429I Discovery development kit features IS42S16400J memory chip. It is 8 megabyte 200MHz DRAM with LVTTL interface supporting 3.3V operation. Having memory on development kit is extremely helpful because we can use existing schematics for reference.

Will 8 megabytes be enough? A five-megapixel colour image encoded with 8 bits per colour will take roughly 15 megabytes. That is too much. However our initial goal is to get image of about 1 to 2 megapixels. Two-megapixel image will be 6 megabytes, which already fits. Another option is to use some sort of compression, in that case we may drive image size much lower.

1.5 Power supply

A good power supply is absolutely essential for proper circuit operation. Previously mentioned components require following voltage rails: 1.8V for sensor core supply, 2.8V for microcontroller and sensor as well as 2.8V analogue for sensor and microcontroller circuits. Memory needs 3.3V.

Let’s calculate power consumption. Table below lists power consumption per every sensor function. According to the table total power consumption is 381mW, which is exactly same as number given at datasheet's first page, however since sensor will be driven at much lower frequency and frame rate real power consumption is expected to be much lower.

	Voltage level, V	Current, mA	Power, mW
Digital operating current	1.8	28	0.05
I/O digital operating current	2.8	38.6	0.108
Analog operating current	2.8	72	0.202
Pixel supply current	2.8	2.4	0.00672
PLL supply current	2.8	5	0.014

Table 1: Estimated sensor power consumption

STM32's operating voltage range is 1.8V-3.3V. [5] Image sensor uses 2.8V and memory 3.3V. Since applying 3.3V to image sensor pins directly will produce too much stress on it, it would be wise to set microcontroller operating voltage at 2.8V. That still allows us to communicate to memory since all STM's digital pins are 5V tolerant and will handle 3.3V coming from memory. Additional benefit is that we can utilize image sensor's analog 2.8V power supply for STM's analog supply pins thus saving on power supplies. Microcontroller power consumption is a function of voltage, frequency and amount of enabled peripherals. Following table lists required peripherals and tries to estimate power consumption at 180MHz frequency and 3.3V power supply (worst case).

Peripheral name	Current, μA	Max frequency, MHz	Total consumption, mA
GPIOA	2.5	180	0.45
GPIOB	2.56	180	0.46
GPIOC	2.44	180	0.44
GPIOD	2.5	180	0.45
GPIOE	2.44	180	0.44
GPIOF	2.44	180	0.44
GPIOG	2.39	180	0.43
GPIOH	2.33	180	0.42
GPIOI	2.39	180	0.43
DCMI	3.72	180	0.67
FMC	21.39	180	3.85
I2C1	4	45	0.18
TIM9	7.22	90	0.65
USART1	4	90	0.36

Table 2: Estimated microcontroller power consumption [6]

According to the table total power consumption by peripherals will be 9.67mA. Of course that does not include many things such as core, clock tree, buses etc. It is interesting to note that almost 40% of power consumed by peripherals will be consumed by FMC, which interfaces external SDRAM. If low current consumption is desired it would be wise to turn FMC off when unused. Timer also consumes surprisingly much power. Absolute maximum current consumption estimated in datasheet when all peripherals are enabled is 77.8mA which makes 220mW at 2.8V. [7] It seems that we will consume much less.

Memory datasheet mentions maximum operating current of 110mA. At 3.3V power dissipation will be 363mW. [8]

PCB will also feature some passive components such as pull-up transistors whose power consumption is low enough to safely ignore it. Perhaps it is worth mentioning that 8 LED-s will be present for debugging purposes. Of them 3 will show software status and 5 will indicate power presence. All LED-s are red with forward voltage of 2V [9]. Current is limited by 1k resistor. Their total current consumption is estimated to be slightly less than 10mA.

This data is enough to select power supply. Here are the requirements for every power rail:

Voltage level, V	Current drawn, mA
1.8	28
2.8	126.8
2.8A	79.4
3.3	110

Table 3: Power requirements

Now it is time to decide between switching or linear power supplies. Since this is not battery powered application energy conversion efficiency is not of concern. Estimated current is also low enough to keep temperatures low. Switching power supplies can also introduce noise to sensitive sensor elements. It would be easier to use linear regulators here. Typical linear regulator in SOT-25 package is usually capable of producing up to 150mA of current. Specific models will be selected later.

Two power supply connectors will be used. One of them is standard 2.54mm header which makes it easy to connect PCB to external power supply. Another is mini USB type B connector, which allows powering camera directly from computer if power supply is not available. Since power wires may be long and possess unwanted parasitic inductance, a simple circuit of resistor and TVS will be used for protection.

1.6 Communication interface

There are many ways of establishing communication between PCB and computer. Of them all UART is perhaps the easiest and fastest to implement. It has relatively slow data connection (up to 3000000 baud in our case) and no advanced noise protection. Since environment is pretty good (office desk), noise should be no problem. Camera will be connected to external USB-to-UART adapter cable via 3-pin connector.

How long will image transfer be with UART? Let's assume UART will be operated at 3000000 baud with 1 start bit, 8 data bits, no parity and 1 stop bit. That makes ten bits per frame of which 8 hold valid data.

$$data\,transfer\,speed = \frac{3000000*8}{10} = 2400000\,bps = 300\,kilobytes\ per\ second$$

An uncompressed 8-bit RGB image of 2592 by 1944 pixels will be 15MB in size. Transferring it will take 50 seconds at 300 kbps, which is ok. Smaller images will be transferred faster. Remember, we are not streaming 4k video here.

1.7 Imaging sequence

Now that main components are selected imaging sequence can be described. Sequence starts when user presses snapshot button. This button is connected to pulled-up SNAPSHOT signal. Pressing the button will drive signal low which will tell microcontroller to start image capture. Alternatively image capture can be triggered by sending message via UART. Microcontroller will then drive TRIGGER signal low, which causes sensor to capture frame. Length of TRIGGER signal determines shutter speed. Shutter speed and image size can be configured via UART before image capture sequence begins.

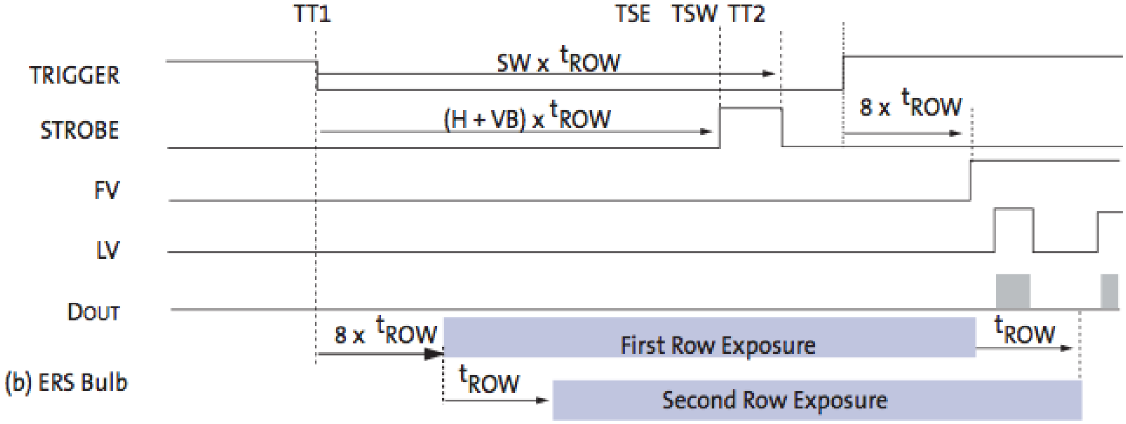


Illustration 4: Line readout sequence showing delay between two consecutive line readouts [13]

Once first pixel row has finished exposure it is transmitted to microcontroller. Principle is following: FRAME VALID signal is driven high during entire frame transmission. Rising edge lets microcontroller know that image will be transmitted soon and gives it time to prepare. Pixels are not exposed simultaneously but single rows start exposure with small delay after each previous row as shown in illustration 4. That ensures that data is read out and transferred shortly after exposure and does not wait somewhere in buffer. Once row has finished exposure LINE VALID signal is driven high and parallel data output is initiated. Data is read in by microcontroller on every falling edge of PIXCLK. Rising edge of LINE VALID signal per every line allows microcontroller to synchronize data in case something gets lost, thus increasing chance of correct frame reception. When entire line is transmitted LINE VALID signal goes low and sensor waits for next line to finish exposure before initiating next line transfer. When all lines are transferred FRAME VALID signal goes low which tells microcontroller that frame is transmitted.

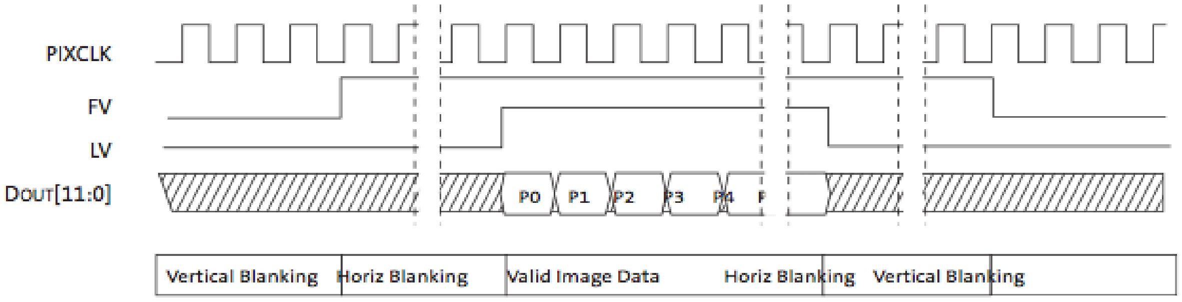


Illustration 5: Signal timing. Data is sampled on PIXCLK falling edge when both FRAME VALID and LINE VALID are acive [14]

It is important to note that line by line scanning may introduce artifacts when pictures of fast moving objects are taken. In that case first line is exposed with object in one location and by the time second line begins exposure object has moved a little causing distortion. This can be avoided by using global reset release mode which exposes all pixels simultaneously. That however requires mechanical shutter to cover sensor while pixels are read out to prevent overexposure. Mechanical shutter is a very complex and precise mechanism and its implementation is well out of scope of this project. Artifacts can be reduced by using faster clock, that way lines will be read out faster and time between line exposures will be reduced.

Photosensitive cells can only capture light intensity and not colour. To make colour images colour filters are used in front of every photocell. They make a so-called bayer pattern. It uses an array of red, green and blue filters that allow only certain light wavelengths to pass. It eliminates a lot of luminosity and is generally avoided in poor lightning conditions. Red, green and blue colours are chosen because their combination can produce every colour in visible spectrum. If we apply 0% of every colour we will get black and if every colour will be 100% we will get white. This colour representation model is called additive because we add components together to get colour information. Since one pixel only represents one colour some data is lost. If we have red pixel for example then green and blue data for that pixel is missing. To solve this we can add 4 neighbour pixels together, that will make picture smaller and only little colour information in green channel will be lost. Another way is to extrapolate data. If we have only red value for a pixel we can get its green component by averaging neighbour green pixels. This way is usually preferred since it allows making bigger pictures and no data from green channel is lost. Many other colour encoding algorithms exist.

Image data arriving to microcontroller is stored in memory. To reduce its size an algorithm combining 4 pixels into 1 is used. That means that red and blue pixels remain unchanged and 2 green pixels are averaged into 1. Another measure of preserving space is converting 12-bit per channel to 8, which is a standard value for many image formats. Below is a drawing which shows how 4 neighbour pixels are located.

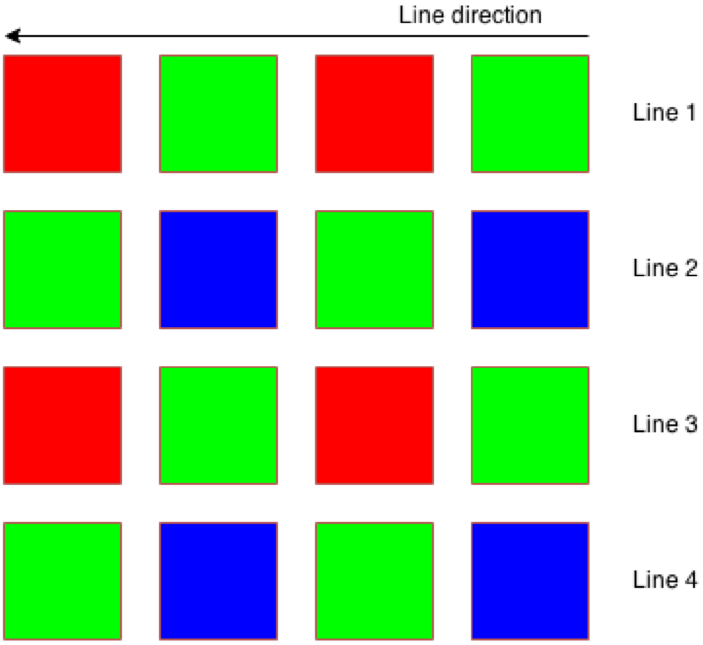


Illustration 6: Pixel location on lines

As was mentioned earlier, image data arrives line by line. 4 pixels: red, green, green and blue are composed together to make a single three-colour pixel. Each pixel has 12 bit resolution and it is reduced to 8 to preserve space. The equations are given below:

$$R8=\frac{R12}{16}$$

$$G8=\frac{G12+G12}{32}$$

$$B8=\frac{B12}{16}$$

Red and blue 12-bit pixels are divided by 16 to get 8-bit value. Since there are 2 green pixels (because more light is available in green spectrum) they are averaged before getting converted to 8-bit value.

But the problem now is that two pixels are located on one line and another two on the next line. We can not calculate green value until both lines arrive. This is solved by storing first line in microcontroller memory buffer array. Then when second line arrives green value is calculated and pixel values are written in external memory. Buffer array can then be used again.

2 PCB Development

2.1 Schematic

This chapter describes component selection and connections between them.

STMicroelectronics has STM32CubeMX software which is intended to simplify development. It allows user to select specific microcontroller, assign its pins and generate code that sets everything up. We will use that program for microcontroller selection.

The smallest STM32F4 series microcontroller with both digital camera interface and SDRAM memory interface is in LQFP144 package. At 22x22mm it is relatively big chip. Of course we can use smaller package such as BGA but then it will be extremely hard to solder by hand, which is not really good for our project. LQFP144 then. Embedded flash memory size does not matter much, so we may choose any vale. 1 and 2 MB are available and we will choose 1 to make board cheaper. Embedded RAM is more important because it can be used for buffers. 256KB is only option available and we will use it.

Next page shows screenshot of pin assignments. LQFP144 package has more pins than we need and some pins are left unused. This is good, it will leave free space for signal traces Assignments and functionality are following:

- 15 pins are used by digital camera interface. That includes 12 bit parallel bus, LINE_VALID, FRAME_VALID and PIXCLK signals. This interface transfers captured images to microcontroller. It is a fast interface, with frequencies up to 100MHz. Higher frequencies are required for higher frame rate, but since we intend to make only single frames we can use much lower frequency. That will greatly simplify design because we can then treat wires as wires and not as transmission lines.
- 2 pins are used by I2C bus. It is used to communicate to sensor and set image parameters and operation mode. Sensor is the only device we communicate to via I2C. It is usually recommended to use 4.7k resistors for pulling I2C clock and data line up. We use 1k instead since it will not affect I2C performance but will simplify bill of materials, because it will eliminate need for 4.7k resistors.
- 2 pins are dedicated for image capture commands. Process is initiated by pressing snapshot button. That drives snapshot signal low which is read in by microcontroller. Microcontroller then drives TRIGGER signal low. Length of TRIGGER determines exposure time.
- 1 pin is used by TIM9 peripheral. It works in output compare mode and generates 6MHz clock for image sensor.
- 2 pins are used by USART1 peripheral. It is used for debugging and data transfer to computer.

- 2 pins are for SWD programming and debug.
- 3 pins are used to control LED-s. LED-s can provide a lot of information about program execution.
- 2 pins are used by external crystal oscillator. The oscillator circuit is a typical 2-pin oscillator with 2 capacitors.

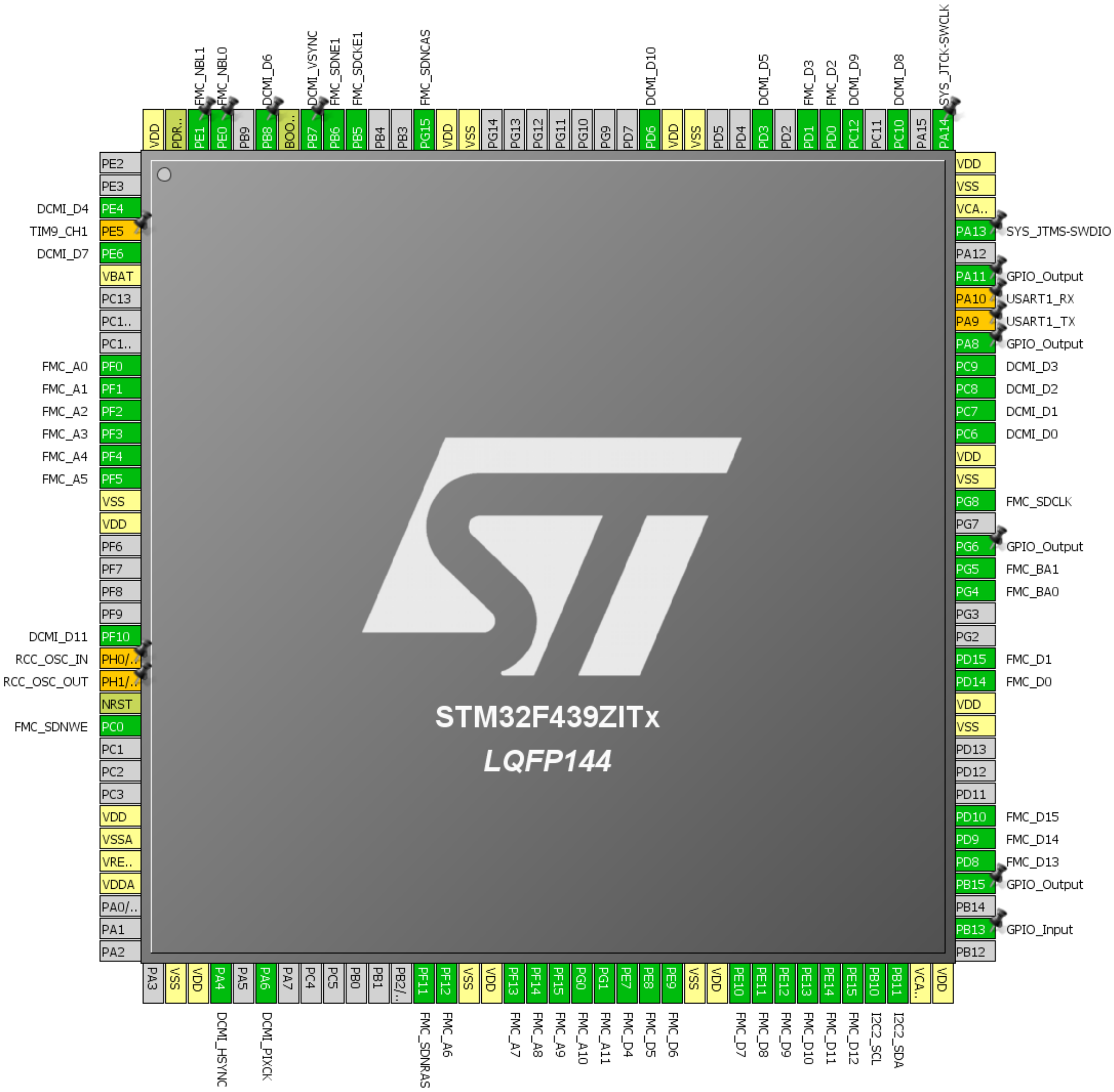


Illustration 7: STM32F4 pin assignment

Crystal oscillator capacitors are equal in capacitance and are selected by equation

$$C1\,, C2 = 2*(Cload - Cstray) = 2*(18\mathrm{p} - 5\mathrm{p}) = 26\mathrm{p}$$

Where load capacitance is 18p according to datasheet [10] and stray capacitance is estimated to be 5pF. Capacitor values C1 and C2 are chosen to be 27pF because large number of 27pF capacitors is available and it is close to calculated value.

100nF ceramic decoupling capacitors are placed as close to every power supply pin as possible to provide path to high current peaks and reduce noise.

User flash memory is selected as boot space. For that, BOOT0 pin is pulled down to GND. The process and possible values are described in STMicroelectronics's AN2606. [11]

PDR_ON pin is driven high to enable power supply supervisor, just as on other packages where this pin is not present. Development kit also has this pin held at high level.

Reset pin is pulled up internally and no external resistor is required. User can reset microcontroller by pressing “reset” button.

Image sensor datasheet provides typical configuration schematic. [12] Although it seems sane several changes are made.

- Serial address is selected as 0xBA by pulling Saddr pin high. Image sensor is only slave device on I2C bus and any of given addresses can be selected.
- STANDBY_BAR pin is pulled high via 0-ohm resistor. Pulling it low forces sensor to enter low power standby mode. Since power saving is not of concern this functionality is not necessary and we can save PCB space by not connecting this signal to microcontroller. If it at some point it will be required, resistor can be desoldered and wire placed instead of it.
- STROBE pin signals when all pixels are exposed simultaneously. Its intended use is external strobe lightning – a strobe must be produced only when all pixels are exposed, otherwise frame will only be lit partially. External impulse lightning sources are not used in this project. However 1mm by 1mm test point is added to allow observing this signal behaviour with external tools such as oscilloscope.
- NRST pin is connected to same reset button that resets microcontroller. That allows resetting them both simultaneously.
- NOE pin when driven high sets sensor outputs to high-Z state. It may be useful when several devices are used on the same bus which is not our case. Therefore this functionality is not required and this pin will be pulled low by 1k resistor.

Memory schematic is quite straightforward. All signals are connected to their corresponding pins on microcontroller. Since PCB is small traces will be much shorter than wavelength and no terminating resistors will be required. Precise calculation will be made during layout process.

Power consumption was estimated in chapter 1.5. It was calculated that no power rail will consume more than 130mA. This is good because we can select power supplies in compact packages. A simple Farnell search reveals that many linear regulators in 5-pin SOT-23 packages can provide this current. Power dissipation is a bit more complicated. Because package is small, very efficient dissipation is not possible and component will heat up. We can estimate that real average power consumption will be smaller due to lower frequency operation and SOT-23 will be able to handle that. Anyway, here are the selected components:

- 1.8V: Torex XC6219B182MR. Its output current is stated to be 240mA. Our consumption is estimated to be 28mA, which will give (0.028*(5-1.8)) = 90mW power dissipation. That is suitable.
- 2.8V: two identical components will be used for both analog and digital supply. Analog supply current is estimated to be no higher than 127mA. This number was taken as absolutely worst case with all peripherals enabled, however my previous experience with STM microcontrollers shows that usually consumption is up to 50mA. Texas Instruments TPS79328DBVRG4 provides 200mA which is fine. At 127mA and 5V input it should produce 280mW. Package should handle it.
- 3.3V: Estimated current consumption is 110mA. That will result in 190mW power dissipation, which should be perfectly handled by Microchip's MCP1801T-3302I/OT linear regulator.

Manufacturer	Name	Output voltage, V	Output current, mA	Power dissipation, mW
Torex	XC6219B182MR	1.8	28	90
TI	TPS79328DBVRG4	2.8A	80	180
TI	TPS79328DBVRG4	2.8	127	280
Microchip	MCP1801T-3302I/OT	3.3	110	190

Table 4: Selected linear regulators and their estimated power dissipation at 5V input

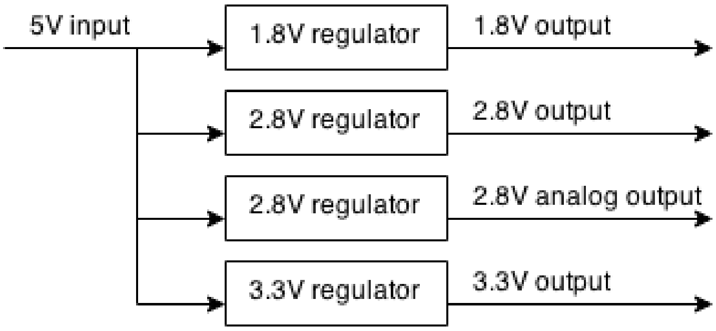


Illustration 8: Power scheme overview

Connecting power wires may produce voltage spike which may damage circuits. A very simple solution is proposed. Transient voltage suppressor (TVS) acts as a clamp and starts conducting when voltage gets too high. Large 10uF capacitor stabilizes voltage level. 1 Ohm resistor is placed in series and its purpose is to keep current from growing uncontrollably especially when capacitor is charged. Resistor ensures that wire parasitic inductance will not store large amount of energy. Since resistance is low voltage drop will be insignificant and can be ignored.

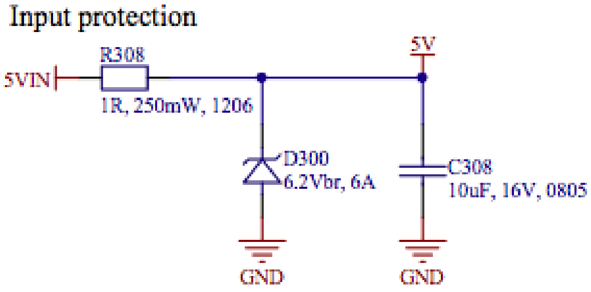


Illustration 9: Input protection

Picture below shows typical linear regulator circuit. All linear regulator circuits in this project are similar. The common components are 100nF capacitor which stabilizes input voltage, 10uF capacitor that reduces ripple voltage and 2.54mm connector for measuring power levels. Only exception is analog 2.8V supply which also features digital and analog ground separation. Separation goal is to prevent high frequency noise from interfering analog circuits. In this project a 0-ohm resistor is used, however it can also be substituted by ferrite bead if necessary.

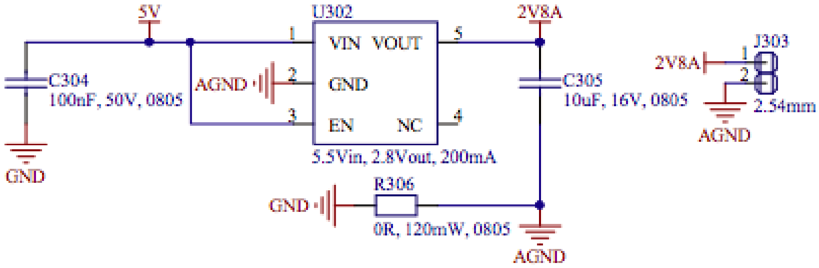


Illustration 10: 2.8V analog linear regulator circuit

5 LED-s are used to show if power is present, one for each voltage level. If at some point PCB does not work user can quickly check these LED-s to determine whether power supply fault is to blame. Red LED-s are used. Their voltage drop is 2V and simple series circuit consisting of 1k resistor and LED can be used for all supplies except 1.8V. 1.8V is lower than diode's voltage drop and that turns diode useless. Problem is fixed by adding NMOS transistor. When base voltage is higher than 0.5V (transistor threshold voltage) transistor opens and allows current flow through LED.

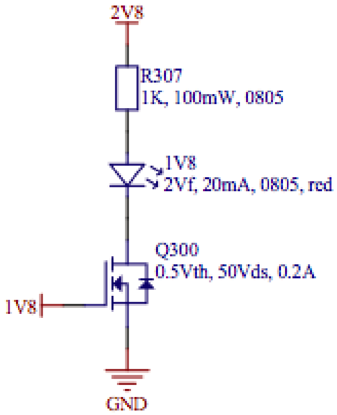


Illustration 11: LED that shines when 1.8V is present

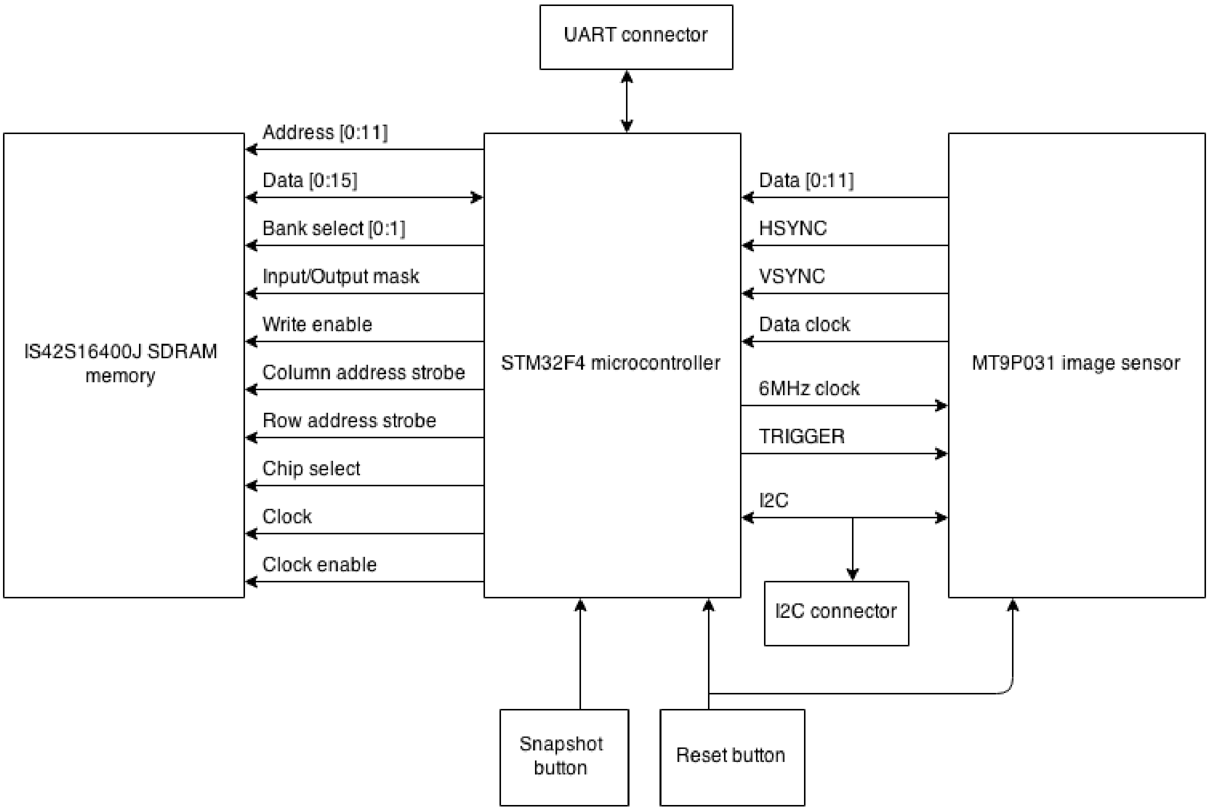


Illustration 12: Main components and connections between them

2.2 Layout

PCB layout starts with outlining main features that will influence component and trace placement. First, PCB should be small. That reduces cost and also keeps tracks short. Another thing to pay attention for is that no huge components should be located hear sensor. Nothing should obstruct view and also there must be enough room for casing. Third idea is to keep all connectors, LED-s and texts on one side. This simplifies testing because it eliminates need for flipping PCB to measure that pesky hidden signal.

We will start by placing 3 most important components - sensor, microcontroller and memory. Both sensor and memory have large parallel high-speed buses coming out of them and going to microcontroller. We have to figure out a way to accommodate them without interfering each other. After that smaller components will be placed.

The sensor operates at 6MHz and its frequency is significantly lower than memory interface frequency, which operates at almost 100MHz. This is why we should be specifically careful with laying memory traces out.

First of all, let's have a look at stackup. This PCB will be 4-layer. That should give us enough room for laying out tracks and placing components. Brandner default stackup is taken as example however other PCB manufacturer I have used – OSHPark – has very similar proportions. The first thing we see is how fat inner isolation layer is compared to other layers – an entire 1.08mm. That is quite a distance. Every time you place via it makes signal path 1mm longer. This is why we should avoid using vias in high frequency signals. Another thing to note is that layers 1&2 and 3&4 are located close to each other. That means that high frequency signal on layer 2 will be greatly affected by copper on layer 1 but not that much by copper from layer 3.

Layer	Name	Height, mm
1	Copper	0.053
	Prepreg	0.18
2	Foil	0.035
	Innerlayer	1.08
3	Foil	0.035
	Prepreg	0.18
4	Copper	0.053

Table 5: Brandner default stackup [15]

That leads to first big decision. Memory and microcontroller will both be located on one layer. Interface between memory and microcontroller will be located on this same layer and layer next to it. Other signals on these 2 layers will be avoided. To reduce parasitic capacitances these 2 layers will be free from polygons. However polygons are great for power supply and grounding and avoiding them completely is impossible. Since 2.8V is the most used voltage level on the board one layer will be dedicated to 2.8V and another will be ground.

Here is what 3 main components location can look like: microcontroller and memory are both located on same bottom (blue) layer while sensor sits on top (red). Sensor is located directly above microcontroller with some room between their pins. That will give us some room for trace and component placement. Sensor is supplied in iLCC-48 package. All pins are located under the body and given the fact that sensor surface is very fragile soldering will be a bit complicated. To simplify soldering and inspection, pads are deliberately made longer so that they stick out. That of course makes footprint larger.

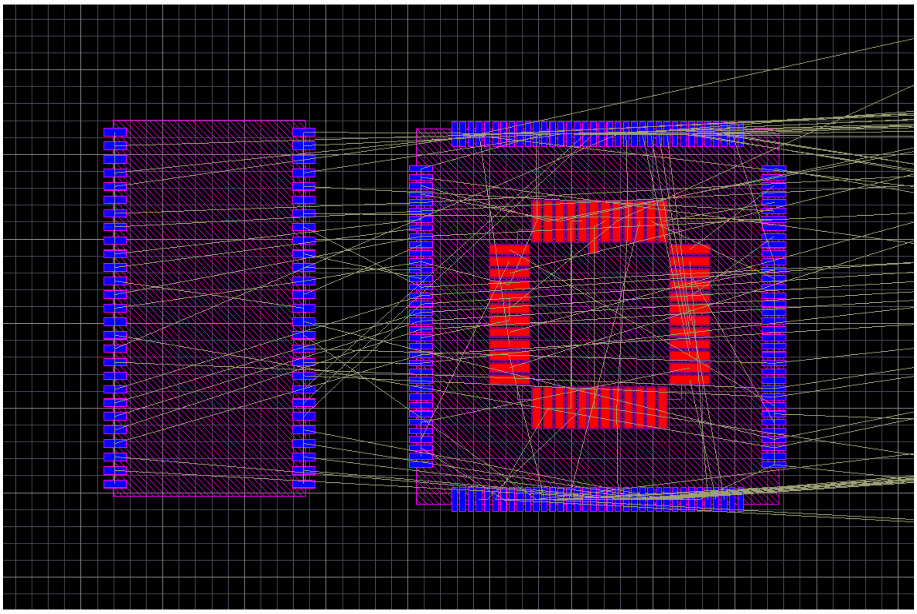


Illustration 13: Main components and their location

Next are decoupling capacitors. They should be located as close to power supply pins as possible. They may be irritating since they take up precious signal trace room but they are absolutely necessary.

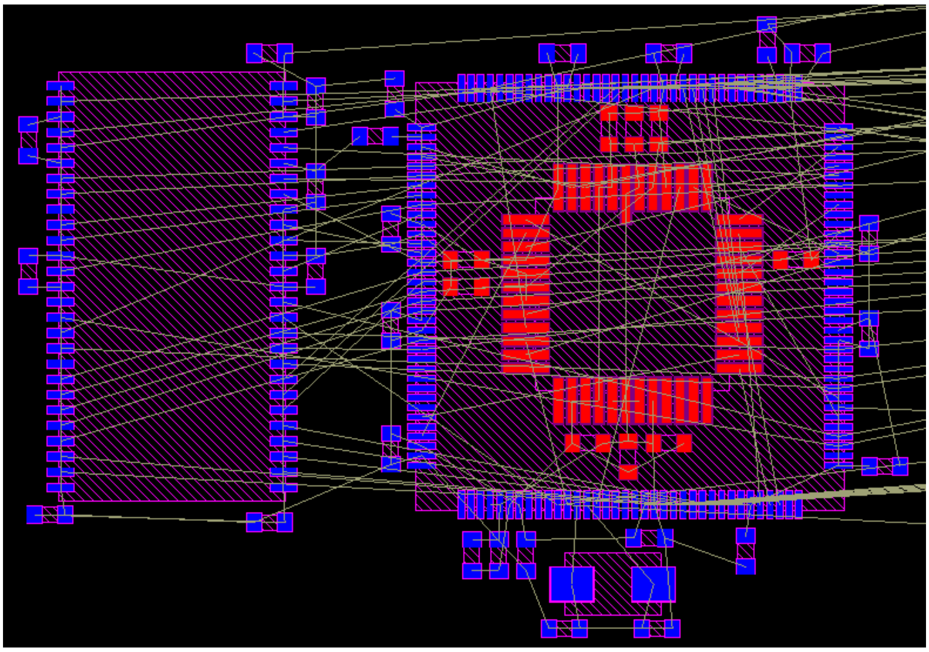


Illustration 14: Decoupling capacitors added

Next picture shows what fully routed PCB looks like. As intended, two bottom layers are used by memory and microcontroller. Free space above memory hosts buttons and several power supplies. Through hole components are located near PCB edges, which helps to preserve some space.

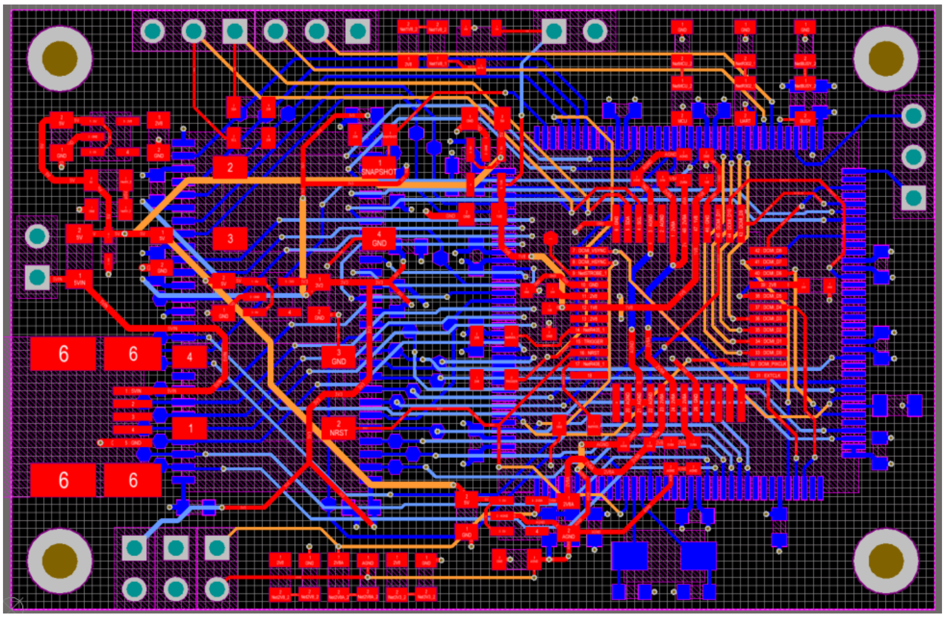


Illustration 15: Fully routed PCB (planes are hidden to make layout visible)

Memory works at 100MHz. Wavelength at this frequency is

$$wavelength\,\lambda = \frac{c}{f} = \frac{300000000}{100000000} = 3\,m$$

where c is speed of light in vacuum and f is signal frequency. PCB is 57 by 37mm. The longest trace is no longer than 50mm. I can assume that wavelength is significantly bigger than PCB trace length and signal reflection effects will be minimal. Terminating resistors will not be used.

PCB has 4 mounting holes in each corner. These holes can fit M2 bolts. Making some sort of case is not a goal of this project, but it is completely possible. Holes have some free space around and no big component should interfere.

Via diameter is 0.2mm, which is relatively small and at the same time allows enough current to pass. No blind or buried holes are present.

On next page layers 3 and 4 are shown. They host memory interface. Distance between signals is kept as high as possible and overlapping is reduced to minimum.

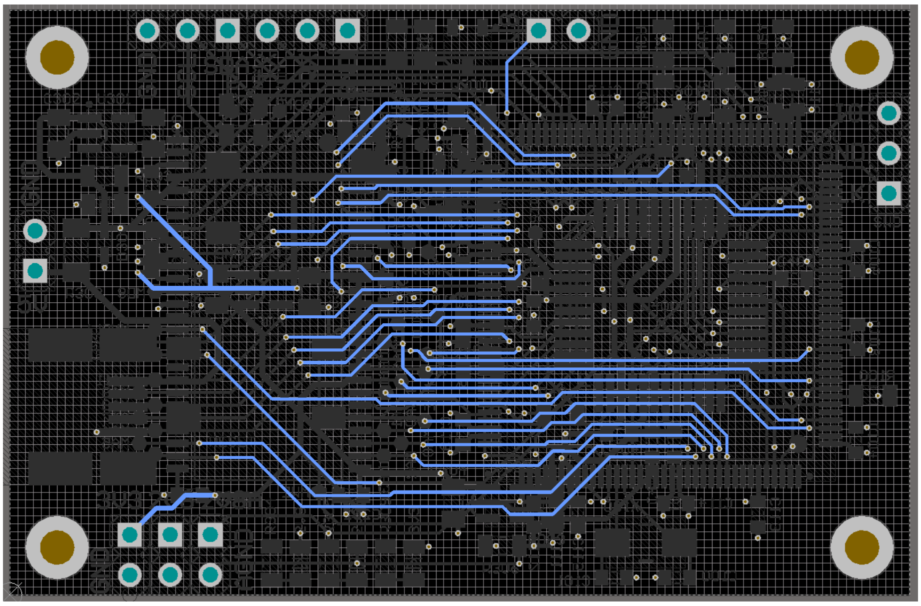


Illustration 16: Layer 3

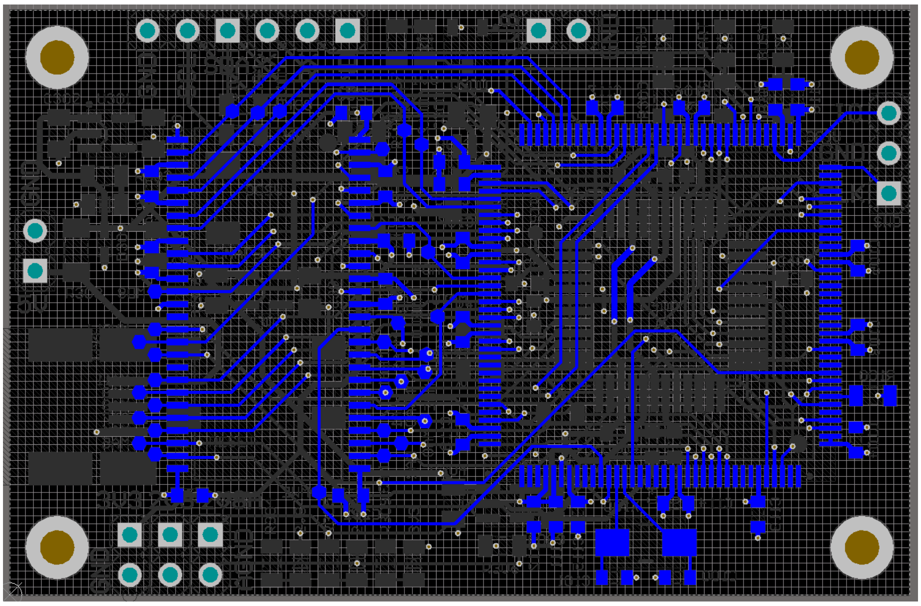


Illustration 17: Layer 4

On next page 3d view is shown. Not only does it allow visual representation of future product but 3d model is also helpful for case design.

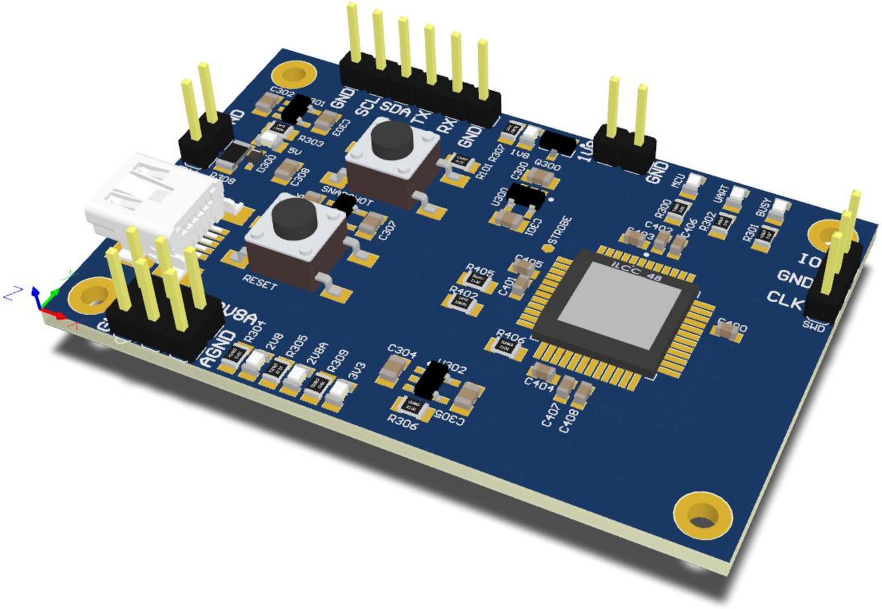


Illustration 18: 3d view

2.3 Testing

First thing to do after PCB is manufactured is to test it. Simple multimeter test indicates no short circuits between power and ground. After PCB is connected to power supply LED-s turn on indicating that all power supplies are good.

Further tests were conducted on PCB with software installed.

How do power supplies behave during power consumption spike? Normally a voltage drop will happen if power supply feedback circuit is slow enough (which is true when dealing with high frequency integrated circuits) or consumed current is higher than power supply maximum current. Decoupling capacitors dramatically reduce this voltage drop, acting as power source at high frequency. Below is oscilloscope screenshot taken during image capture process:

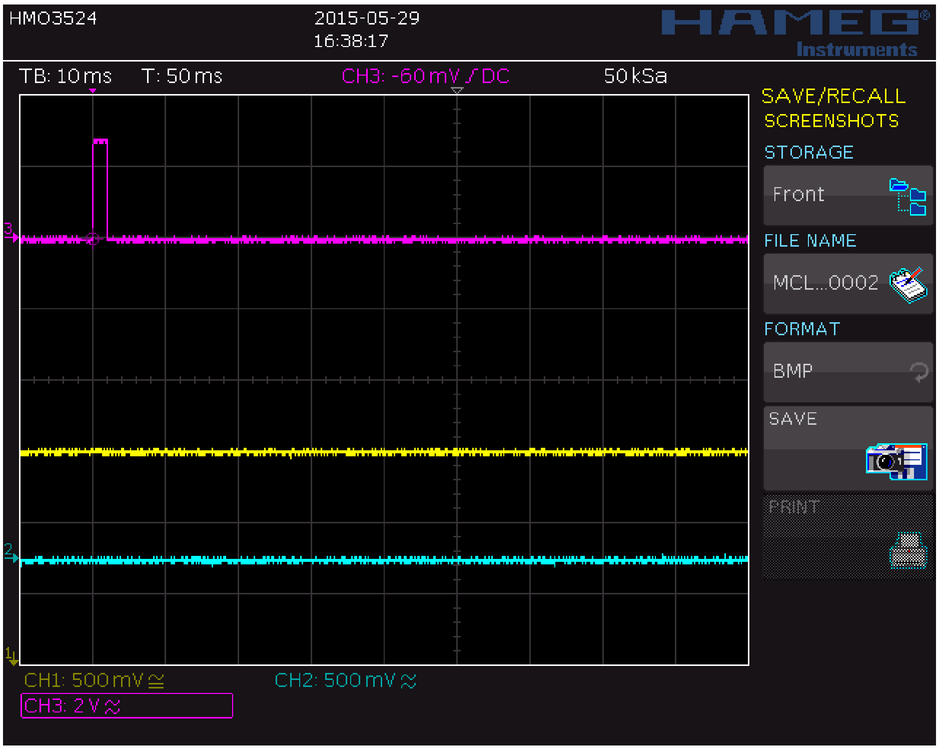


Illustration 19: Power supply behaviour during image capture process

Purple signal is FRAME_VALID. When this signal is high, image is captured and transferred from sensor to microcontroller. Yellow signal shows 2.8V supply, which is used by both microcontroller and sensor. Blue signal is analog 2.8V, which is mainly consumed by sensor analog circuitry. If these power supplies are unstable image quality may degrade or capture process fail. Several tests were done and none showed any significant deviation from normal value.

Another test I was eager to perform was to see UART waveform at 3Mbaud. This is the first time I used such high frequency and I was not sure if it works well with 1m long cable. Image below shows UART TX signal at PCB connector. Some minor overshoots and oscillations are visible, but they are no higher than 500mV. Microcontroller has ESD protection diodes built-in which should handle that. However a better solution would be standalone fast-acting protection diodes on PCB along with small current-limiting resistor.

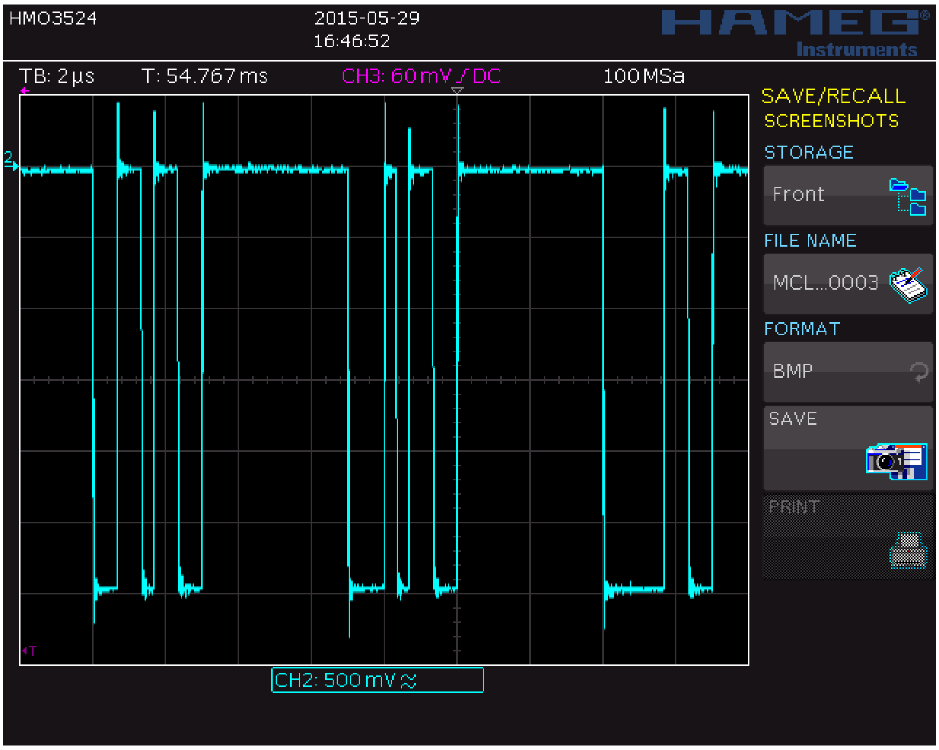


Illustration 20: UART at 3 Mbaud

Image sensor default operating mode is continuous. Right after power up it starts transmitting images at 14fps and full resolution. This behaviour can be disabled by pulling sensor's STANDBY_BAR pin low. However, it was left unconnected to microcontroller during design stage and the only way to stop frames coming in is by sending command via I2C. That causes slight power consumption spike after turning on.

During normal operation current consumption is around 130mA (at 5V input). Normal operation is when microcontroller operates a maximum frequency (180MHz), sensor is ready for capturing image and memory is clocked. During image acquisition process consumed current can rise to 180mA.

3 Software

3.1 Embedded

This chapter describes software which is used on STM32 microcontroller. At the moment of writing this text the software is not entirely complete. Memory interface is missing, which makes receiving large images impossible. STM32 is operating well however and communicates to sensor. Sensor makes small pictures and they can be sent to computer.

Software is written in Atollic TrueStudio IDE. C language is used. STMicroelectronics provides software libraries that cover mostly low-level operation which involves register operations. Our software uses them extensively. No operating system is used.

Compiled software is flashed on PCB with ST-LINK tool. ST-LINK is included in STM32F429Discovery kit and is available as 6-way connector. Of these 6 pins only ground, SWCLK and SDWIO are used in our PCB.

Heap and stack are both located in microcontroller embedded memory. External memory is mapped to FMC block 5 and possible addresses lie in range 0xA0000000 – 0xA07A1200 and can be interfaced via pointer.

Below is software in pseudocode:

```
main {
    set up buffers and variables
    initialize communication interfaces
    initialize sensor and set its parameters
    initialize memory interface
    print welcome message
    loop {
        if picture has to be taken {
            drive trigger signal low
            read image in, process it and store it
            send entire image to computer
        }

        if message arrives from computer {
            process it
        }
    }
}
```

Camera is connected to computer via UART. User can send commands to configure image parameters such as shutter speed and size. Image capture can also be initiated via UART. When started, camera helps user by printing possible command list and explanations how to use them. Typical command looks like that:

```
cmd –width 999 –height 999 –speed 100 –pic –getdata
```

"cmd" indicates start of command frame. Camera then searches for "-" symbol which indicates start of specific command. Command is then followed by values if applicable. In the example above camera is ordered to take 1000 by 1000 pixel picture with exposure length of 100ms and send it to computer. Here is a list of possible commands:

- row – sets first image row to specified coordinate. Even values in range [0:2004] are accepted.
- column – sets first image column to specified coordinate. Even values in range [0:2750] are accepted.
- width – sets image width. Odd values in range [1:2005] are accepted.
- height – sets image height. Odd values in range [1:2751] are accepted.
- speed – sets shutter speed. Values starting from 1ms are accepted.
- pic – initiates frame capture. After receiving this command camera will take picture with previously specified parameters and store it in memory.
- getdata – initiates image transfer process. After receiving this command camera will transfer image as 8-bit RGB comma separated values. 8-bit is chosen to make transfer faster, however in future command will be added for configuring output parameters. Image data is preceded by header indicating image parameters such as width and height.
- barpattern – forces sensor to output predefined vertical monochrome bar pattern instead of capturing real image. Sensor has several patterns which purpose is to test image reception. They were very helpful during development process as many errors came during image reception software testing, especially when pointing to specific place in two-dimensional array.
- redpattern – produces red image. Red component is 100% for all pixels, green and blue are 0%.
- greenpattern – produces green image. Green component is 100% for all pixels, red and blue are 0%.
- bluepattern – produces blue image. Blue component is 100% for all pixels, green and red are 0%.
- nopattern – sets sensor back to normal operating mode

After command is executed successfully "OK" response is sent back. If error occurred, "ERROR" response is sent with possible reason. Much more commands will be added later to provide more control over sensor.

UART heavily uses interrupts for data transmission. Usual method of writing data to register and waiting for it to become empty is not very good because it uses lots of processor time. To save time, messages are first saved into queue data structure. When previous transmission ends, interrupt is generated and if there is data in queue it is transferred to register for transmission. This system allows much smoother transmission of messages because main program does not have to wait while hardware becomes available. Incoming messages are also stored in memory and are read out when main program is free.

Image reception is done by polling if register has received new data. This enables processing data on the fly. If we were using DMA then data would be automatically stored in memory without processing. Interrupt-based reception is also not practical because data arrives fast and processor switching from main context to interrupt takes time.

Image sensor datasheet specifies that minimum operating clock frequency is 6MHz [16]. During software testing it was observed that it works with 1MHz frequency as well. Frequencies below that were not tested.

Further code development will include memory interface for storing large images, better data encoding for transmission and many more minor performance and feature improvements.

3.2 On computer

All data sent from microcontroller to computer is formatted as ASCII string terminated by carriage return and newline characters. Therefore we can simply access it with any serial console program such as PuTTY. Manual operation is however slow and annoying and because of that a script was written that handles communication.

The goals of the script are:

- configure camera to required operating mode
- set up image parameters
- order camera to take picture
- receive picture
- save picture as a file

Python was chosen as script language because it is simple and powerful enough to handle that task. Python is also very high-level language which means one can create complex applications relatively fast.

Some preparations are needed before executing script. First of all, camera must be connected to computer via USB-to-UART cable. Next, camera has to be powered up and completely initialized. Camera is ready for communication when red “MCU” led is blinking.

Script is executed by typing

```
python camera1.py
```

in console. That will start script which then sets up camera to predetermined parameters, orders it to take picture and stores picture on computer as PNG file.

Script pseudocode is following:

```
connect to serial port
find available filename
set camera parameters
tell camera to do picture with given parameters
get response from camera
set up three-dimensional matrix to store image data
read data in
save matrix as PNG image
```

Image manipulation heavily relies on OpenCV. OpenCV is an open source computer vision library that includes many image processing functions. We use it for saving arrays as file and splitting RGB images into separate channels.

PNG file format is chosen to store images. The reason is that PNG offers lossless compression. PNG photos will take less memory space than non-compressed formats such as TIFF. JPEG usually performs better for storing images than PNG but it adds compression losses which we cannot afford here.

It was previously mentioned that sensor can transmit artificially generated patterns. This is exceptionally useful for testing image processing pipeline because we know what result to expect. Below 125 by 125 monochrome vertical bar pattern sent by sensor:

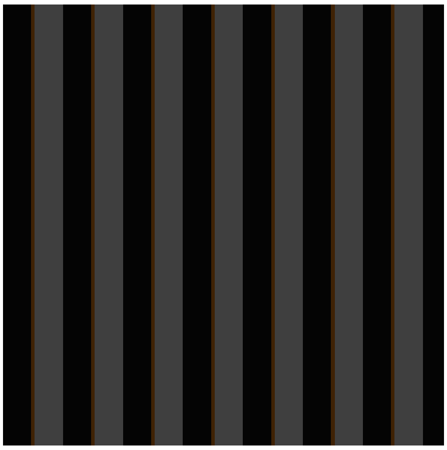


Illustration 21: Monochrome vertical bar pattern

Brown artifacts can be observed on edges. This happened because bar width was odd number and two pixels – green and blue – were located on lighter bar while another two pixels – red and green – were located on dark bar.

Computer-side software has many possibilities of improvement. Not only more camera commands can be implemented but also some image processing such as demosaicing can be performed on computer. It will simplify testing of image processing algorithms at expense of transfer speed.

Conclusion

In this project a digital camera was developed. It was based on MT9P031 image sensor. This sensor is easy to control and can provide relatively good image quality. Based on image sensor outputs entire camera system was developed which included STM32F4 series microcontroller, 8 megabyte SDRAM memory buffer and proper power supply. PCB was then developed and produced. Embedded software was developed which handled image sensor operation and image transfer to computer. On computer script was executed to properly receive and save image. Proposed design was proved to be working.

Development process showed that creating digital camera requires knowledge from many different fields such as optics, mechanics electronics and software. Many issues arise during product design. Sensor has to work efficiently, data transmission must not be a bottleneck and software must be effective and simple to use. Author received a good overview of entire imaging pipeline, starting from sensor operation and finishing with creating image file on computer. Several image sensors were evaluated and their working principles studied. High-speed interface between sensor and microcontroller was implemented. An introduction to image storage and processing was made. Software was written that interfaced camera to computer. The interface allowed for both data and command transfer. This was author’s first experience with interfacing PCB with computer.

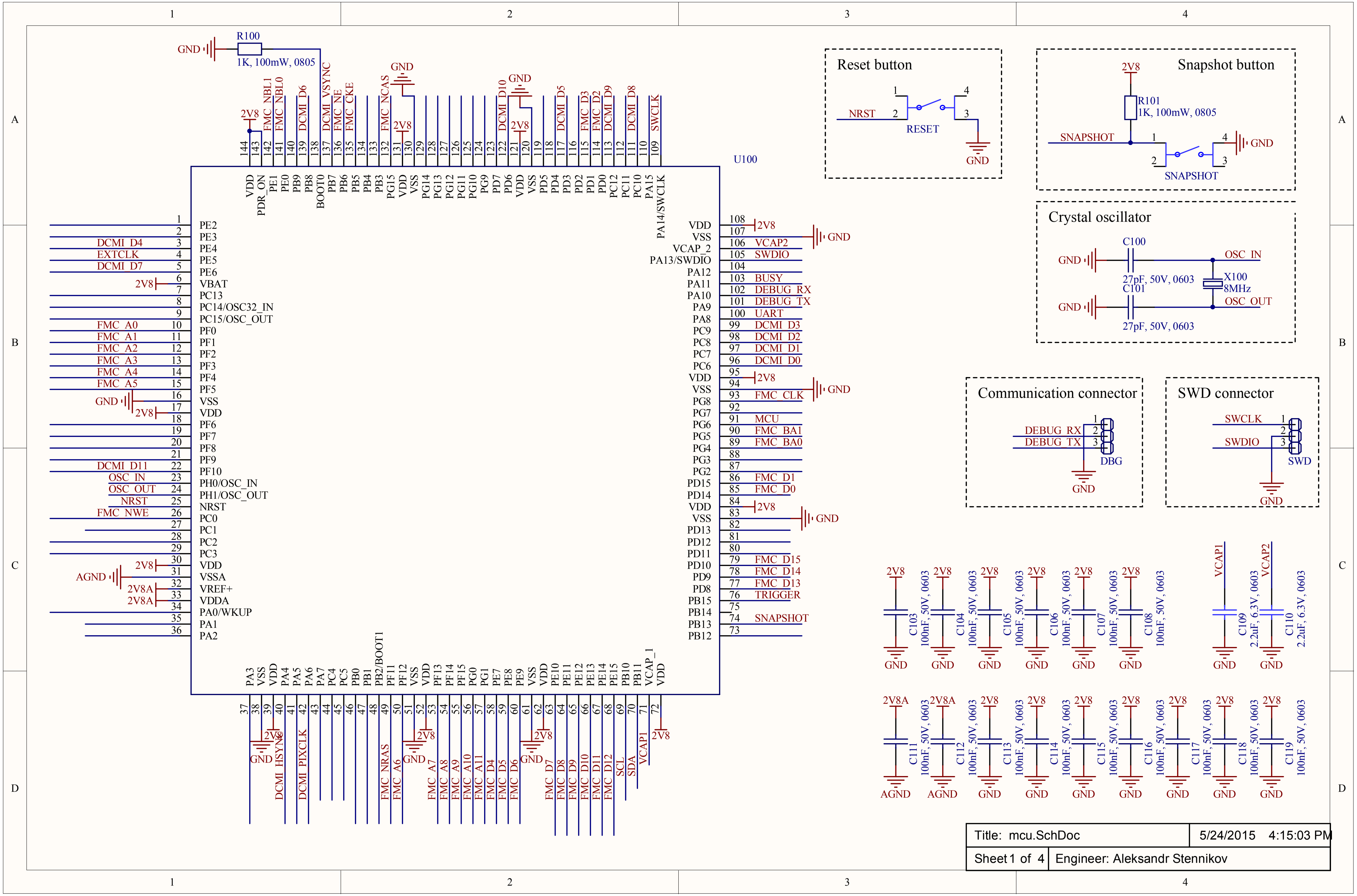
I enjoyed working on this camera. It greatly expanded my knowledge of the subject and gave me a lot of valuable information which I will definitely use in later projects. The main goal now is to perfect the software to make use of full sensor's potential. First of all memory interface needs to be implemented to buffer larger images. Then control commands will be updated to allow more advanced camera control. Data transfer will be made faster. Later this project may be expanded by using bigger sensor and faster processor capable of running operating system such as Linux. It will greatly simplify software creation and will open lots of new possibilities.

Kasutatud kirjandus

1. ON Semiconductor application notes [WWW] http://www.onsemi.com/PowerSolutions/supportDoc.do?type=AppNotes&rpn=KAI-02150 (25.05.2015)
2. KAI–02150 datasheet [WWW] http://www.onsemi.com/pub_link/Collateral/KAI-02150-D.PDF (25.05.2015)
3. MT9P031 datasheet p 5 [WWW] https://www.aptina.com/assets/downloadDocument.do?id=865 (25.05.2015)
4. MT9P031 datasheet p 43 table 21 [WWW] https://www.aptina.com/assets/downloadDocument.do?id=865 (25.05.2015)
5. STM32F43x datasheet p 1 [WWW] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00077036.pdf (25.05.2015)
6. STM32F43x datasheet p 112-114 [WWW] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00077036.pdf (25.05.2015)
7. STM32F43x datasheet p 107 table 31 [WWW] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00077036.pdf (25.05.2015)
8. IS42S16400J datasheet p 15 [WWW] http://www.issi.com/WW/pdf/42-45S16400J.pdf (25.05.2015)
9. KPT-2012SURCK datasheet p 2 [WWW] http://www.kingbright.com/attachments/file/psearch/000/00/00/KPT-2012SURCK%28Ver.17A%29.pdf (25.05.2015)
10. Crystal oscillator datasheet p1 [WWW] http://www.farnell.com/datasheets/1754353.pdf (25.05.2015)
11. STMicroelectronics AN2606 [WWW] http://www.st.com/web/en/resource/technical/document/application_note/CD00167594.pdf (25.05.2015)
12. MT9P031 datasheet p 6 figure 2 [WWW] https://www.aptina.com/assets/downloadDocument.do?id=865 (25.05.2015)
13. MT9P031 datasheet p 31 figure 21 [WWW] https://www.aptina.com/assets/downloadDocument.do?id=865 (25.05.2015)
14. MT9P031 datasheet p 12 figure 8 [WWW] https://www.aptina.com/assets/downloadDocument.do?id=865 (25.05.2015)

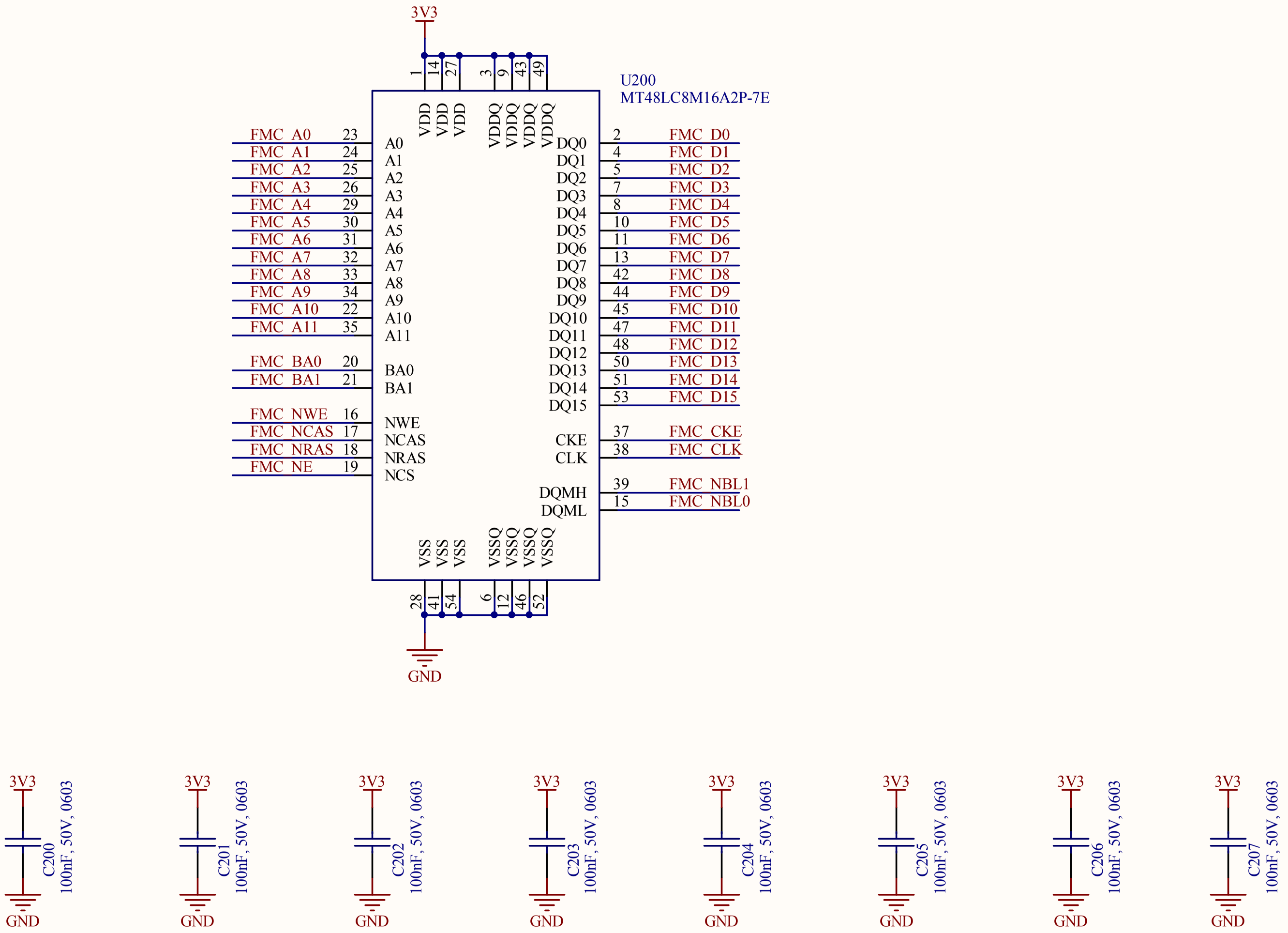
15. Brandner stackup page with default 1.6mm stackup [WWW]
https://www.brandner.ee/eng/74/93 (25.05.2015)
16. MT9P031 datasheet p 5 [WWW]
https://www.aptina.com/assets/downloadDocument.do?id=865 (31.05.2015)

Lisad

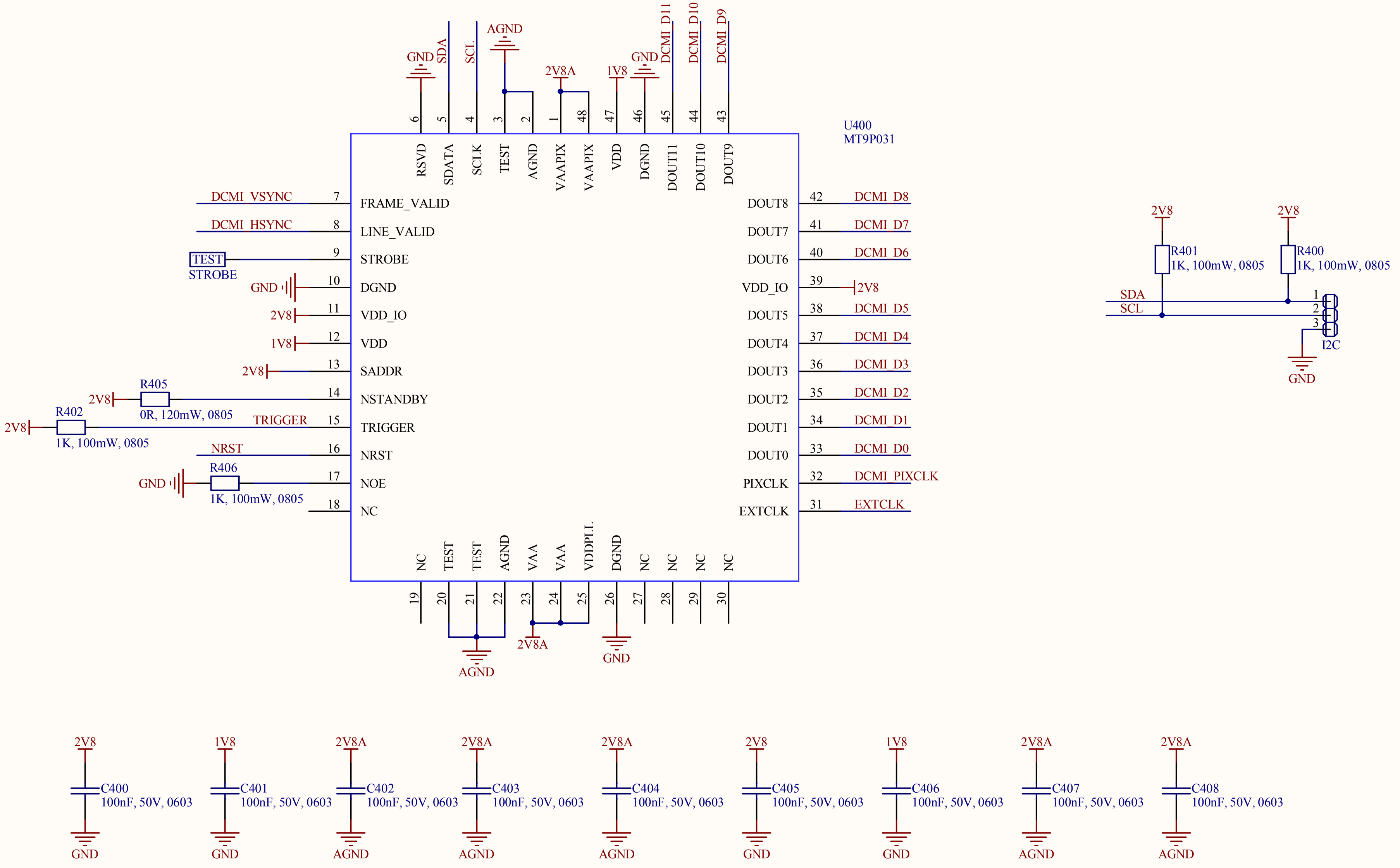


Title: mcu.SchDoc 5/24/2015 4:15:03 PM

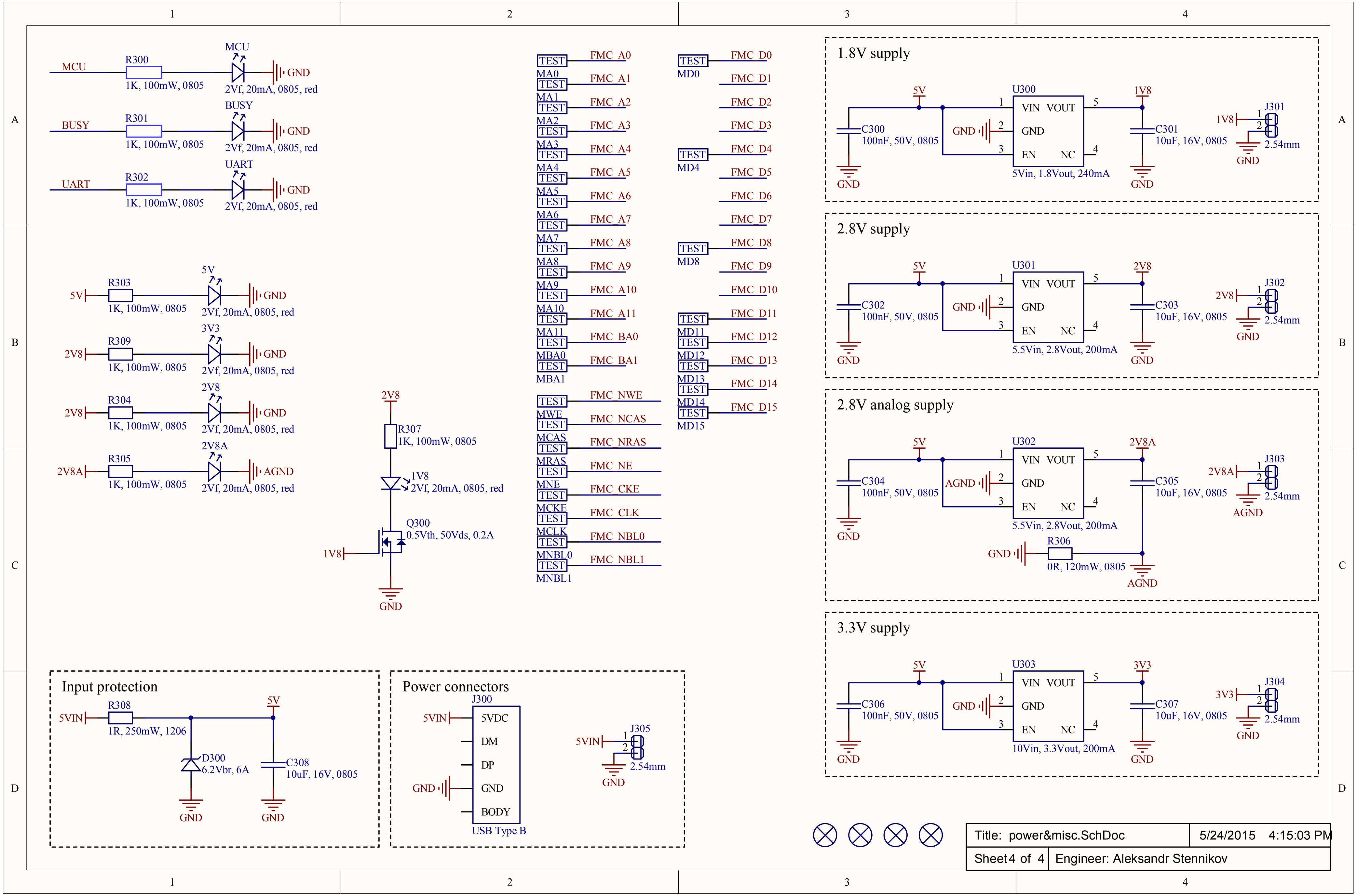
Sheet 1 of 4 Engineer: Aleksandr Stennikov



Title: memory.SchDoc	5/24/2015 4:15:03 PM
Sheet 2 of 4	Engineer: Aleksandr Stennikov



Title: sensor.SchDoc		5/24/2015 4:15:03 PM
Sheet 3 of 4	Engineer: Aleksandr Stennikov	



Title: power&misc.SchDoc	5/24/2015 4:15:03 PM
Sheet4 of 4	Engineer: Aleksandr Stennikov

Comment	Designator	Footprint	Product Name	Supplier	Supplier Number	Quantity
2Vf, 20mA, 0805, red	1V8, 2V8, 2V8A, 3V3, 5V, BUSY, MCU, UART	LED2012X06N	2012SURCK	Farnell	2099241	8
27pF, 50V, 0603	C100, C101	CAPC1608X06N	MC0603N270J500CT	Farnell	1759058	2
100nF, 50V, 0603	C103, C104, C105, C106, C107, C108, C111, C112, C113, C114, C115, C116, C117, C118, C119, C200, C201, C202, C203, C204, C205, C206, C207, C400, C401, C402, C403, C404, C405, C406, C407, C408	CAPC1608X06N	MC0603F104Z500CT	Farnell	1759123	32
2.2uF, 6.3V, 0603	C109, C110	CAPC1608X06N	06036C225KAT2A	Farnell	1657930	2
100nF, 50V, 0805	C300, C302, C304, C306	CAPC2012X06N	MCCA00296	Farnell	1759167	4
10uF, 16V, 0805	C301, C303, C305, C307, C308	CAPC2012X06N	C0805C106K4PACTU	Farnell	1288204	5
6.2Vbr, 6A	D300	SOD-523	PESD5Z5.0	Farnell	1829230	1
Pin Header 3x1	DBG, I2C, SWD	Pin Header 3x1	826629-3	Farnell	3418297	3
USB Type B	J300	Mini USB	54819-0572	Farnell	9786473	1
2.54mm	J301, J302, J303, J304, J305	Pin Header 2x1	826629-2	Farnell	3418285	5
0.5Vth, 50Vds, 0.2A	Q300	SOT95P245X110-3N	BSS138LT3G	Farnell	2101819	1
1K, 100mW, 0805	R100, R101, R300, R301, R302, R303, R304, R305, R307, R309, R400, R401, R402, R406	RESC2012X06N	MC01W080551K	Farnell	9333711	14
0R, 120mW, 0805	R306, R405	RESC2012X06N	MCMR08X000	Farnell	2073603	2
1R, 250mW, 1206	R308	1206[3216] resistor	CRCW12061R00FKEA	Farnell	1653087	1
Tactile Switch	RESET, SNAPSHOT	SMD Tactile Switch	FSM4JSMA	Farnell	3801305	2
STM32F439 144-Pin	U100	TSQFP50P2220X2220X160-144N	STM32F439ZGT6	Farnell	2393672	1
IS42S16400J-7TLI	U200	TSOP80P1200X120-54N	IS42S16400J-7TLI	Farnell	2253831	1
5Vin, 1.8Vout, 240mA	U300	SOT95P300X145-5N	XC6219B182MR	Farnell	8797021	1
5.5Vin, 2.8Vout, 200mA	U301, U302	SOT95P300X145-5N	TPS79328DBVRG4	Farnell	1287687	2
10Vin, 3.3Vout, 200mA	U303	SOT95P300X145-5N	MCP1801T-3302I/OT	Farnell	1578370	1

MT9P031	U400	LCC70P1000X1000X125-48N	MT9P031I12STC	Farnell	1695390	1
8MHz	X100	5.0x3.2 SMD Crystal	ABM3-8.000MHZ-D2Y-T	Farnell	2101329	1