# PORTABLE LIGHT FIELD CAMERA UTILIZING AN ANGLE-SENSITIVE PIXEL IMAGER

**A Design Project Report**

**Presented to the School of Electrical and Computer Engineering of Cornell University**

**in Partial Fulfillment of the Requirements for the Degree of**

**Master of Engineering, Electrical and Computer Engineering**

**Submitted by**

**Jason Wright**

**MEng Field Advisor: Alyosha Molnar**

**May 2013**

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

**Project Title**: Portable light field camera utilizing an angle-sensitive pixel imager

**Author**: Jason Wright

## Abstract

A prototype of a portable implementation of a light field camera utilizing an angle-sensitive pixel imager was designed and built. Using a custom hardware design with an ATXmega128A1 microcontroller as its core, data from the rolling-shutter imager is read and processed into a format that can work with a number of unique image processing algorithms, including computational refocus and passive 3D range mapping. The device includes dual 80 MSPS ADCs, 64 MB of SDRAM, an FTDI USB-to-Serial converter and micro-USB connector, an SD card slot, a TFT LCD touchscreen, and an integrated charge controller for a 2000 mAh polymer lithium ion battery.

## Table of Contents

Table of Contents ........ 3
Executive Summary ........ 5
Introduction ........ 6
  Light Field Imaging ........ 6
  Angle-Sensitive Pixels ........ 7
Previous Setup ........ 8
  Hardware ........ 8
  LabVIEW ........ 9
  MATLAB ........ 9
Hardware Design ........ 9
  Requirements ........ 9
  Top Level Design ........ 10
  Imager (Top Board) ........ 12
  Amplifiers & Digital Counter (Bottom Board) ........ 13
  Data Acquisition ........ 15
  Microcontroller ........ 17
  External Bus Interface (EBI) and SDRAM ........ 19
  USB ........ 20
  SD Card ........ 21
  Battery ........ 22
  Touchscreen ........ 23
  Lens Hardware ........ 24
  Manufacturing ........ 25
  Errata ........ 25
Software Design ........ 26
  Requirements ........ 26
  Top Level Design ........ 26
  UART Interface ........ 27
  Imager Timing ........ 28
  Touchscreen Library ........ 28

Troubleshooting ..... 28
Results & Conclusion ..... 29
Other Options ..... 31
Future Work ..... 31
Works Cited ..... 32
Appendix A: PCB Schematics ..... 33
Bottom Board ..... 33
Control Board ..... 34
Clock Board ..... 34
Appendix B: Code Listing ..... 34
AVR Firmware ..... 34
MATLAB Scripts ..... 35
Appendix C: Bill of Materials ..... 35
Bottom Board ..... 35
Control Board ..... 37
Clock Board ..... 39
Appendix D: Hardware Photos ..... 40

## Executive Summary

A light field camera, also called a "plenoptic" camera, is a device capable of capturing information about the direction of an incident ray of light, in addition to its wavelength and intensity. By treating light as a "field" of vectors, this type of device enables many interesting algorithms within the field of computational photography, including computational refocus (changing the effective focal length after an image has been taken) and passive 3D range mapping using a single image.

For this project, an imager that utilizes angle-sensitive pixels as a means of capturing light field information was used as the core of a prototype for a portable light field camera. The imager consists of an array of photodiode charging circuits with paired diffraction gratings above, each with different angle, spacing, and phase. By exploiting the Talbot effect to relate intensity response with incident angle, light field information can be reconstructed in software.

Previous experimental setups used NI DAQmx equipment to record information from the imager. To facilitate experiments in a wider range of test conditions, a portable light field camera was designed and built. Using a custom hardware design with an ATXmega128A1 microcontroller as its core, data from the rolling-shutter imager is read and processed into a format that can work with a number of unique image processing algorithms, including computational refocus and passive 3D range mapping. The device includes dual 80 MSPS ADCs, 64 MB of SDRAM, an FTDI USB-to-Serial converter and micro-USB connector, an SD card slot, a TFT LCD touchscreen, and an integrated charge controller for a 2000 mAh polymer lithium ion battery.

The PCBs containing the custom hardware design meet the proposed specification, and the core of the software to get data from the imager using the camera is complete. However, more work needs to be done to synchronize frame acquisition with the timing of the imager's rolling shutter and to get the TFT LCD touchscreen working properly.

## Introduction

### Light Field Imaging

Normal digital cameras record images by exposing an array of photosensitive pixels to light reflecting from a scene. Individual photodiodes record the amount of light measured at their particular location in the array, called an imager, and those values can be used to construct an image. However, the incident light on the camera lens contains more information than the photodiodes measure. While conventional photosensitive pixels aggregate all incident photons into one intensity value, the photons themselves also have direction, polarization, and phase. (The wavelength, or color, of the light is not directly measured; rather, a typical imager will contain a Bayer filter – a repeated pattern of pixels filtered to be sensitive to red, green, or blue light.)

The purpose of a light field camera is to capture information about a field of light, represented as vectors with intensity (their magnitude) and with direction. This information enables interesting possibilities in computational photography, such as synthetic refocus, wherein an image can be assigned an arbitrary depth of field in software after the image has already been captured.

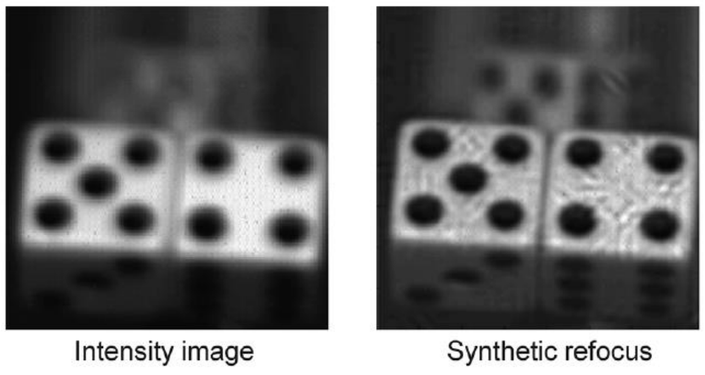


Figure 1. Computational refocus example [1].

In addition, it is possible to use light field images to extract a 3D range map from a single image by comparing the relative angle between neighbor pixels. Objects that are very close to the imager will have steep angle differences between neighboring pixels, while objects that are very far away will have incident rays that are closer to parallel and smaller angle differences.

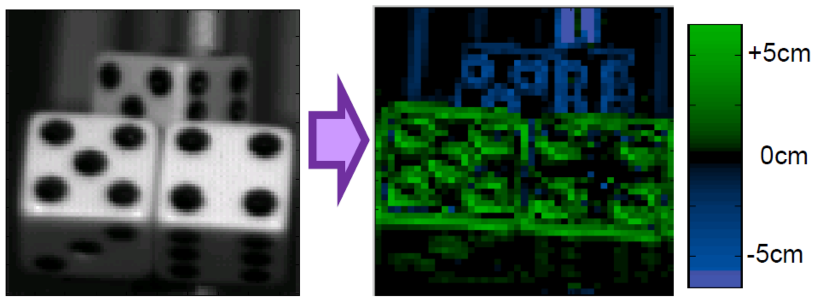


Figure 2. Passive 3D range map example [1].

Other applications of light field cameras include light field tomography and microscopy [2], and generation of pure “4D” images for light field displays [3], which are theoretically capable of generating 3D images with a very wide viewing angle.

## Angle-Sensitive Pixels

One approach to light field imaging is to use an imager that records the angle of incident light in addition to its intensity at each pixel. To accomplish this, the imager uses tiled subunits of conventional photodiode circuits, each with a diffraction grating etched above. Each diffraction grating is at a different angle, spacing, and phase from the others, and each photodiode has a complementary photodiode with a diffraction grating that is π out of phase.

By virtue of the Talbot effect, which relates incident angle with intensity for a 2-layered diffraction grating, a periodic intensity response can be assumed. Because each collection of subunits (a “macropixel”) has diffraction gratings with many unique combinations of angle, spacing, and phase, one incident angle can be reconstructed, providing sufficient information to run light field-related post-processing algorithms.

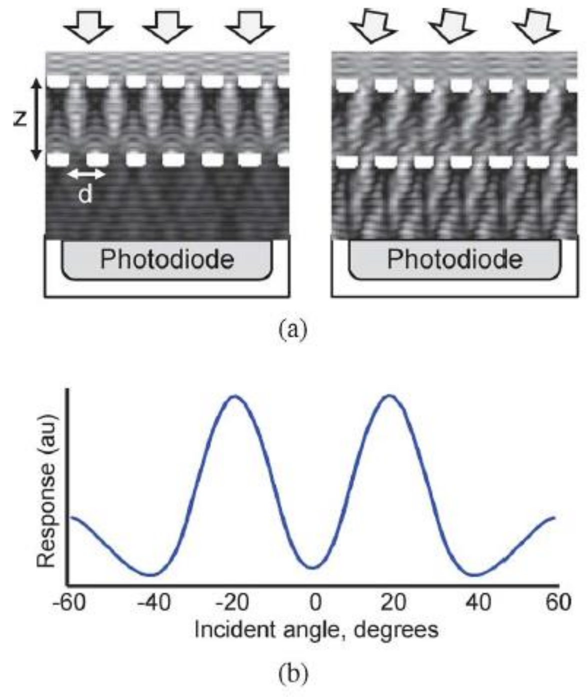


**Figure 3. Side view of each angle-sensitive pixel (a). The Talbot effect, a periodic intensity response to incident angle (b) [1].**

## Previous Setup

### Hardware

The existing setup consisted of two PCBs on an optical table, connected to an NI DAQmx for image capture. The NI DAQmx uses a fiber optic cable for high-speed data transmission, so it needs to be connected to a computer via PCI. This requires BIOS to recognize the device when the computer boots up, so the DAQmx must be powered on before the computer boots.

The DAQmx can be configured and tested using Measurement & Automation Explorer, accessible from within LabVIEW. The particular chassis in lab is a PCIe-6537.

The four external SCB-68 boxes must be connected, with ribbon cables hooking into the 16-pin machine headers on the camera. (For each header, the top pins are data and the bottom pins are all connected to ground.)

The board itself must be powered using an external power supply. It can accept any input voltage from 3.3 V to 5 V, and has on-board LDO voltage regulators to ensure a stable supply. The power pins are at the top left corner of the board. The 2nd pin should be connected to 5 V and the 4th pin should be connected to ground.

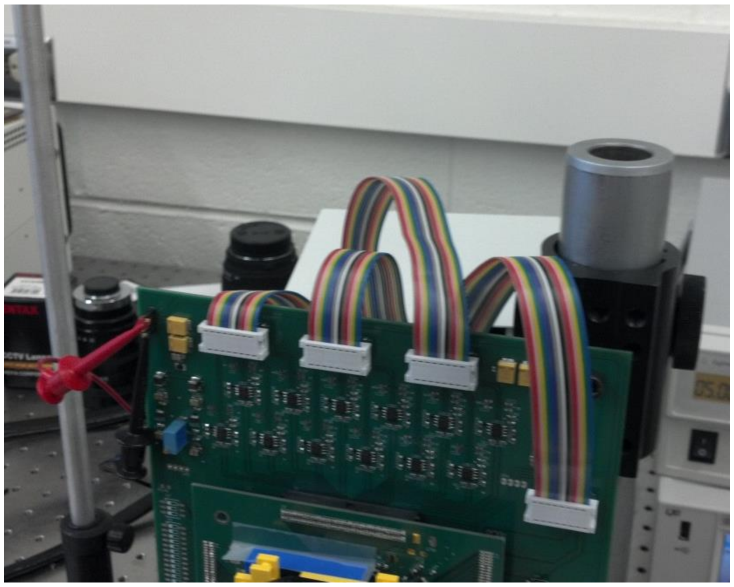


**Figure 4. Image showing the required connections for the original board. The updated board has the machine pin headers lined up at the top, and doesn't require an external power supply.**

The board also requires an external function generator to clock the imager. The imager should receive a 10 MHz square wave with a peak-to-peak voltage of 900 mV, with a DC offset of 450 to 900 mV. This can be accomplished using a tri-coupler and a second DC power supply to bias the function generator. There is an RF coaxial connector on the top board (the one containing the imager) where the input clock signal should connect.

### LabVIEW

I worked with a LabVIEW VI, `dumdumfull.vi`, which was written to work with the NI DAQmx to read analog voltages on 27 channels. The VI is fairly straightforward, and the MATLAB script described below makes working in LabVIEW unnecessary. However, the DAQmx does require a little bit of configuration – in particular, it's important to check connectivity with the PCIe chassis using Measurement and Automation Explorer before running the VI.

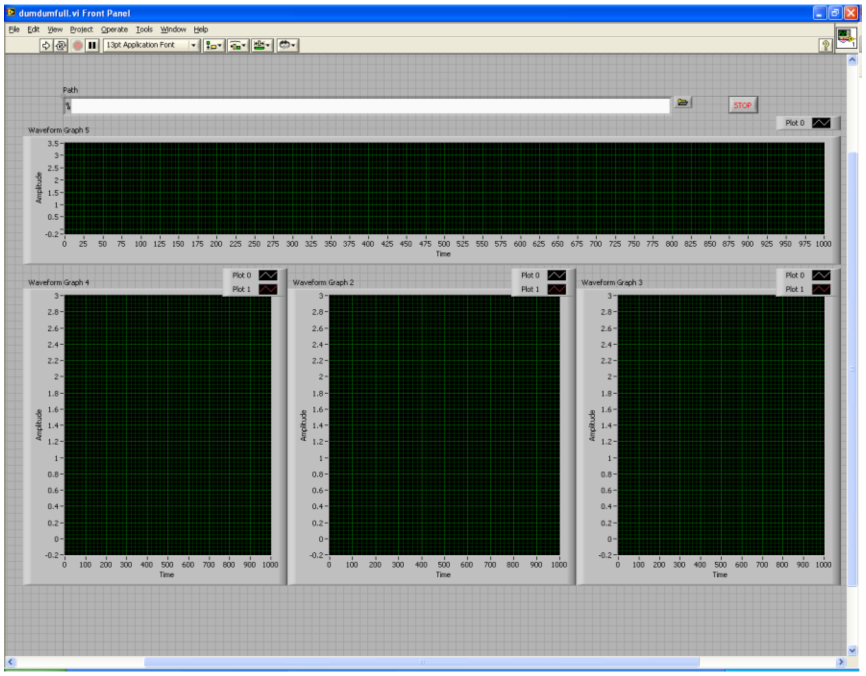


**Figure 5. Front panel of the LabVIEW VI.**

### MATLAB

I worked with existing MATLAB code written by previous PhD students to process the images. The two main scripts used, `unpacker.m` and `reassembler.m`, are used to translate the analog voltage readings from the DAQmx into an image and assemble the image into its ASP components, respectively.

To simplify the process of taking images, I wrote a script `takePicture.m` (included in appendix B) to make acquiring images using the DAQmx a one-step process. The script takes as its argument a filename to save as, and includes an ActiveX component to automate running the LabVIEW VI. This requires LabVIEW to be open to work.

## Hardware Design

### Requirements

The main design requirement for the camera hardware was to create a portable version of an existing experimental setup that relied on large, expensive fiber optic NI DAQmx equipment for data acquisition. To that end, the camera needed the following components:

— Reliable analog-to-digital conversion, to capture data from the imager.
— Temporary storage of an entire frame of image information.
— USB compatibility.
— An integrated, portable power supply.
— A user interface that allows images to be captured and stored easily.

Given these basic requirements, the main design goal was to build a device that could capture images at a frame rate that is as high as possible, while maintaining a reasonable image resolution. To begin selecting components, a MATLAB script was written to calculate the timing requirements for the ADCs and microcontroller (see `adcCalculations.m` in appendix B)

```
>> adcCalculations
Individual ADC Sample Rate: 1.7241 MHz
Total ADC Sample Rate: 41.3793 MHz
8 Bit ADC Slave Sample Rate: 41.3793 MHz
16 Bit ADC Slave Sample Rate: 82.7586 MHz
Number of XMegas/frame required: 1.2931
```

This script assumes that the imager is clocked at 10 MHz, that all 24 data lines must be read, and that the microcontroller is a 32 MHz device. The Individual ADC sample rate number is derived from previous data acquired from the imager using the DAQmx. The data sample rate in `dumdumfull.vi` is 10 MHz, and the `SLICEBIN0` line (the lowest bit of the imager's digital counter, which flips every time the output data lines change which pixel they are connected to) has a low-to-high transition every 12.8 samples. Within that period, the first few samples are thrown away since the photodiode needs a small amount of time to charge (every `SLICEBIN0` transition releases the charged voltage on the imager's photodiode circuitry), and the last few samples are averaged together. To get images as fast as possible with a minimum sample rate, the prototype portable camera is designed to only take one sample within that 1.28 µs period. Given a 32 MHz microcontroller, it is assumed that there will be enough of a delay between the interrupt triggered by `SLICEBIN0` undergoing a low-to-high transition and actual data acquisition to give the photodiode a sufficient amount of time to charge.

The XMegas/frame result of 1.2931 is assuming a 32 MHz device, and was calculated to consider the possibility of having "slave" microcontrollers to acquire data in parallel. However, the design and software development is a lot simpler using one microcontroller, and the cost of acquiring an image using two frames from the rolling shutter is not particularly detrimental. The imager itself is capable of acquiring images at video frame rate or above, but designing a data acquisition system for those speeds was determined to be outside the scope of this project.

The camera also has the physical requirement of being able to mate with a commonly-available lens, and for it to be handheld and portable.

## Top Level Design

The camera is split up into four boards:

— Top board – contains the imager and lens mount. This is unmodified from the original lab setup.

— Bottom board – contains amplifiers, latches, and buffers, and switches for configuring the imager.

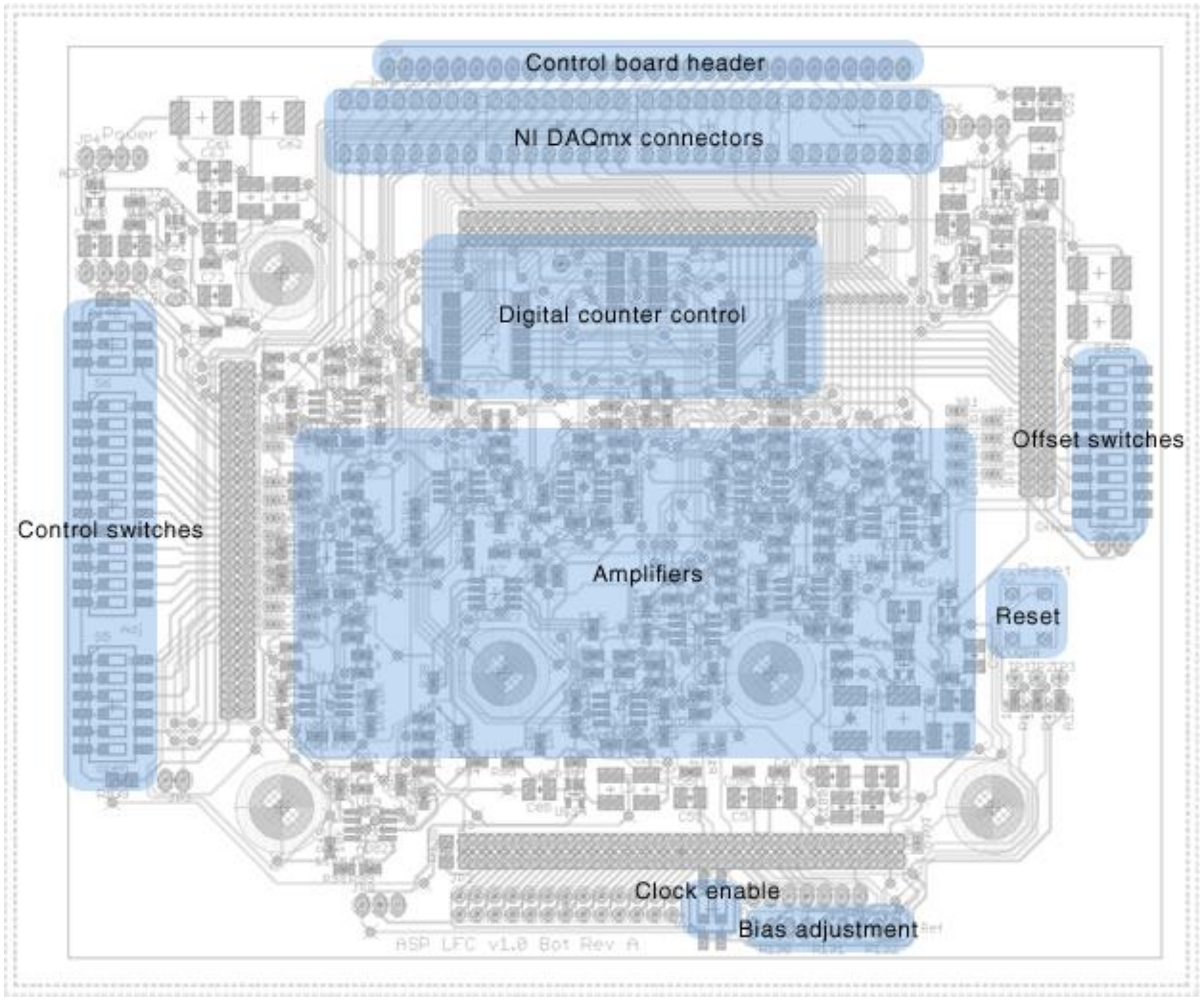


Figure 6. High-level diagram of the bottom board.

— Control board – contains the microcontroller, ADCs, multiplexer, USB and SD card connections, touchscreen, and battery.

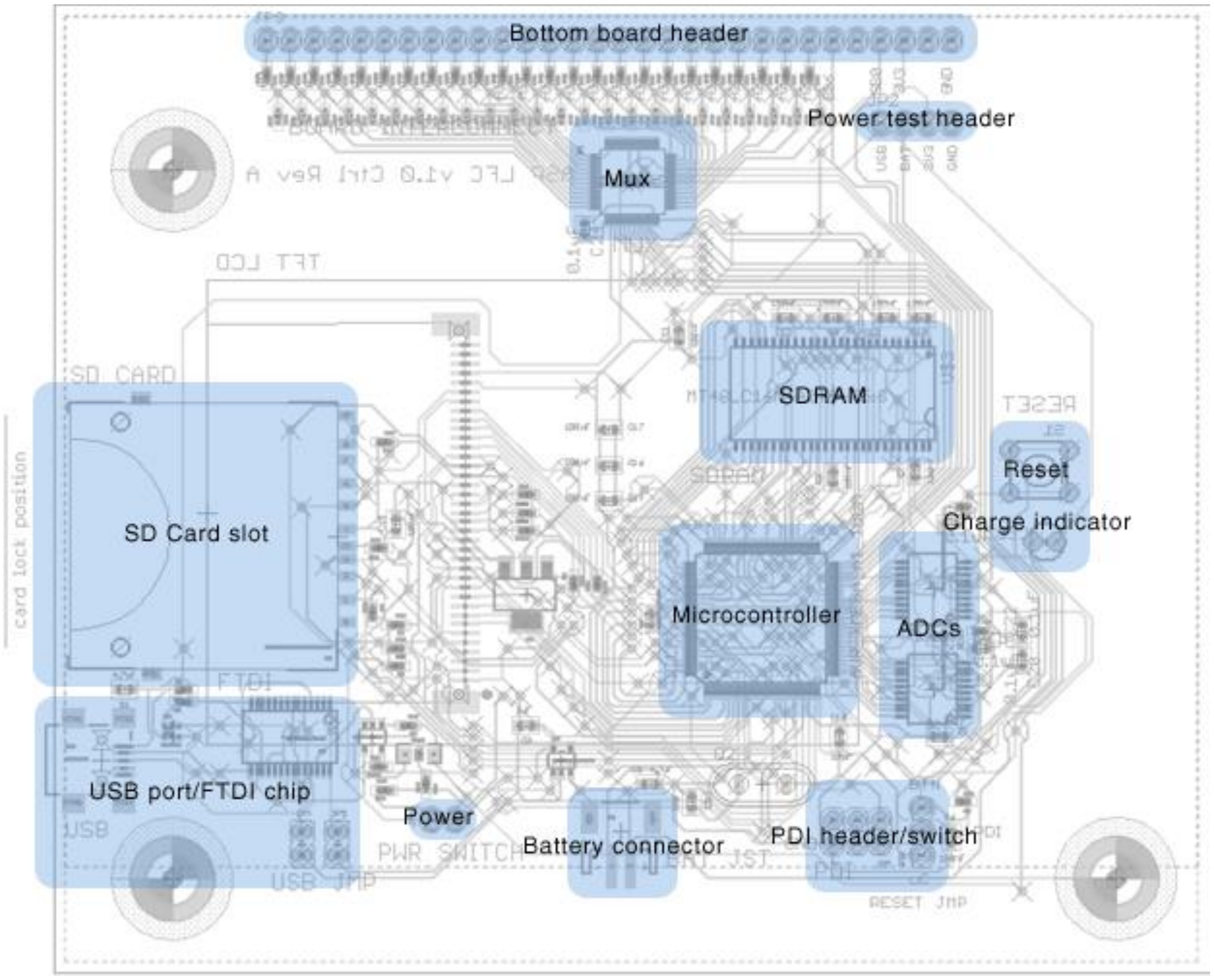


Figure 7. High-level diagram of the control board. Not labeled: Touchscreen, on the back of the board.

— Clock board – contains a 10 MHz oscillator and coaxial cable connection for use with the top board.

## Imager (Top Board)

The imager is composed of 153,600 pixels, arranged as a 48 by 50 tiling of subunits. There are on-chip ADCs built into the imager, but I chose not to use them for two reasons:

1) I wanted to mimic the existing experimental setup as closely as possible, which used the NI DAQmx for analog-to-digital conversion.
2) I wanted precise control over timing, and debugging the on-chip ADCs would have been very difficult without a more in-depth understanding of how the imager chip worked.

The imager has a number of configurable parameters, which are broken out to switches on the PCB. Most can be left in the off position, but it is useful to use the offset switches (the ones on the right side of the PCB) for hardware debugging. They allow changing the starting offset of the digital counters used to roll through the pixels in the imager.

## Amplifiers & Digital Counter (Bottom Board)

The bottom board primarily contains amplifiers for each analog signal line. Each amplifier has the following form:

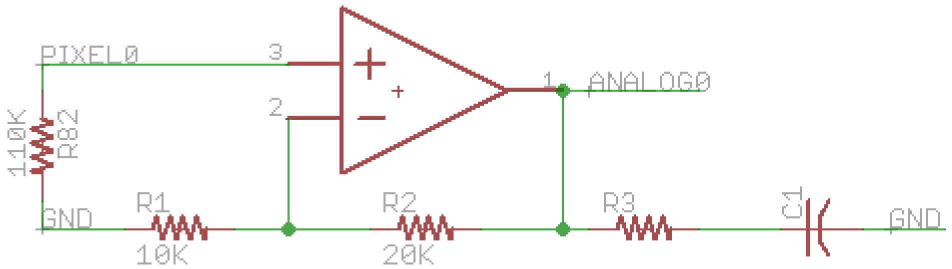


Figure 8. Non-inverting amplifier.

This is a normal non-inverting amplifier with $G = 1 + \frac{R2}{R1} = 3$. The other components are used for impedance matching and frequency compensation.

The board also has two D-type flip-flops and two inverting octal buffers connected to the digital counter signals, which are driven by the imager. The imager uses a 7-bit addressing scheme, so the highest bit of READBIN6 (the row address) is used as the clock of the first flip-flop, and BREAKOUTBAR6 is the (inverted) output. The non-inverted output is used as the clock of the second flip-flop, and the (inverted) output is connected to the input, which divides the frequency in half to create an output signal FRAME (connected to a test point on the board and can be used to verify the frame rate of the shutter). For data acquisition purposes, measuring low-to-high transitions of BREAKOUTBAR6 is sufficient to mark the start of each frame. The rest of the octal buffers are used to create the other BREAKOUT signals, which can then be used to read the column address of the shutter at any time. They are left unconnected on the control board because they are not needed for data acquisition.

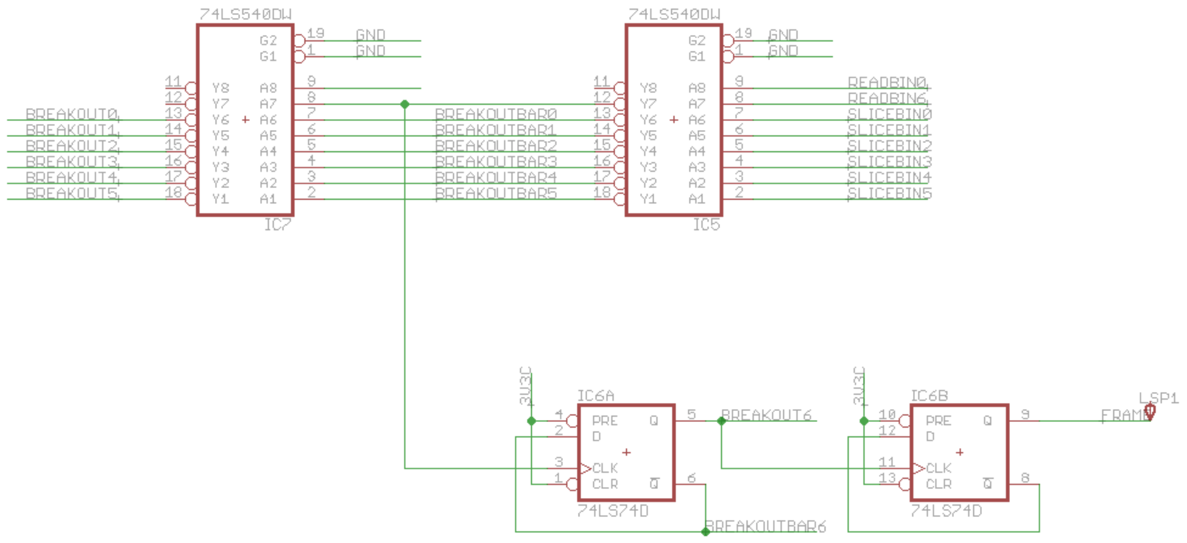


**Figure 9. Digital counter breakout circuit.**

The board also contains several switches for configuring the imager and potentiometers for fine-tuning the imager's current biases.

The following sets of switches are on the board:

- S7 (left side, top set of 3): Used to enable or disable internal sum/difference computation between complementary pairs of ASPs. For normal operation, these should all be left in the OFF position.
- S6 (left side, middle set of 12): Configuration switches. These are used for setting the gain of the imager's internal amplifiers. For normal operation, these should all be left in the OFF position.
- S5 (left side, bottom set of 5): Address switches. If the top switch (ADDREN) is ON, the other switches can be used to drive the imager's address. For normal operation, these should all be left in the OFF position.
- S4 (right side, set of 9): Offset switches. These can be used to change the initial value of the imager's address when in rolling shutter mode. For normal operation, these should all be left in the ON position, except the lowest switch (LINK_CORE) which should be left in the OFF position.
- S1 (bottom, set of 2): Clock enable/Amplifier enable switches. For normal operation, these should both be left in the ON position.

The board also contains 3.3 V and 1.8 V LDO voltage regulators. The imager uses a 1.8 V reference, and the 3.3 V is used to drive the CMOS logic chips. The board also contains Tyco-style connectors to mate with the top board containing the imager and lens mount.

## Data Acquisition

Given the requirement of an ADC capable of 41.3793 MHz data acquisition, I chose to use external ADCs rather than use the ADCs typically integrated into common microcontroller packages. After some searching, the AD9283 (8-Bit, 50 MSPS/80 MSPS/100 MSPS 3 V A/D Converter) was determined to be a good, low-cost option. The 80 MSPS package easily meets the specified requirement, and I chose to include two ADCs so that conversions could be done in parallel. Each ADC uses 1 port of an 8-bit microcontroller for data acquisition, and 2 pins to control the conversion.

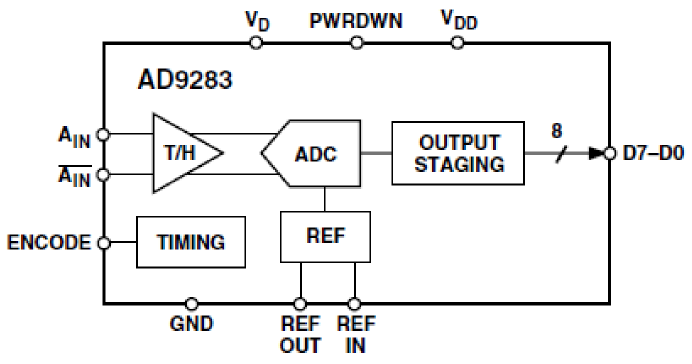


Figure 10. Functional block diagram of the AD9283 [4].

The AD9283 uses a track-and-hold SAR design [4]. The input voltage is first used to quickly charge a capacitor, which then feeds through a comparator with the output of a successive approximation register (SAR). The SAR steps through all quantization levels of the output using a bit-per-stage pipeline type converter utilizing switch capacitor techniques to find the closest match. There is an input voltage reference pin, although the ADC generates a nominally 1.25 V reference on its own.

The consequence of the AD9283's fast conversion time is its low input voltage range, which is specified by the datasheet as 1 V pp. To accommodate this, the input analog voltages are stepped down using a resistive divider from a maximum of 3.3 V (measured experimentally).

Since there are 24 analog data lines and only two ADCs available, an analog multiplexer is needed to switch the signals. The ADG726 is capable of 16-to-1 analog multiplexing with a switching time of 30 ns, ensuring it won't be the limiting determinant of the effective frame rate. The ADG726 also comes in a dual package, working perfectly with the dual ADC setup.

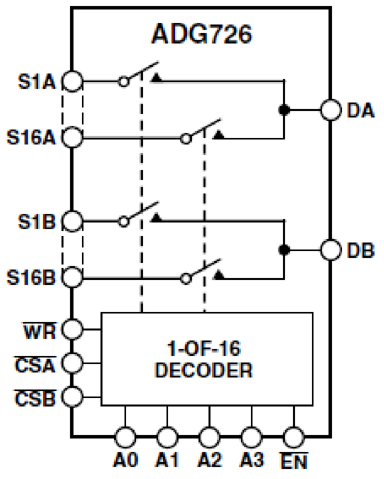


Figure 11. Functional block diagram of the ADG726 [5].

The combined circuit for the dual ADCs and the analog multiplexer is as follows:

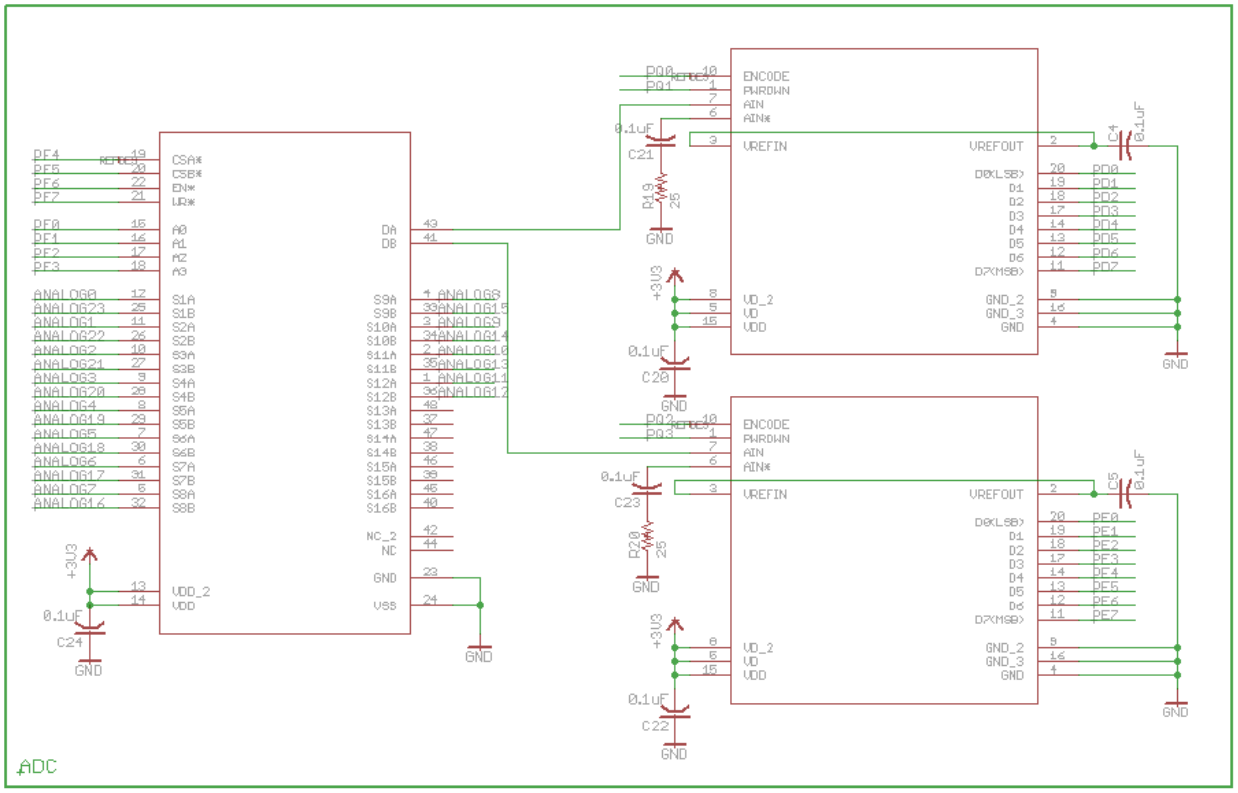


Figure 12. ADC and analog multiplexing circuit.

The analog multiplexer is designed to connect directly to the stepped-down analog signals, and to PORTF of the microcontroller. The outputs connect to the two ADCs, which are controlled by PORTQ and output the results of the conversion on PORTD and PORTE. The choice of decoupling capacitor and resistor values are based on the recommended circuits in the corresponding datasheets for each device.

## Microcontroller

After evaluating a few different options, the ATXMEGA128A1 microcontroller by Atmel was selected for the following reasons:

- Internal 32 MHz clock.
- 100-pin package (almost all were needed).
- Built-in external bus interface (EBI) to add additional memory.
- Prior experience programming AVR microcontrollers.

The circuit for the microcontroller is as follows:

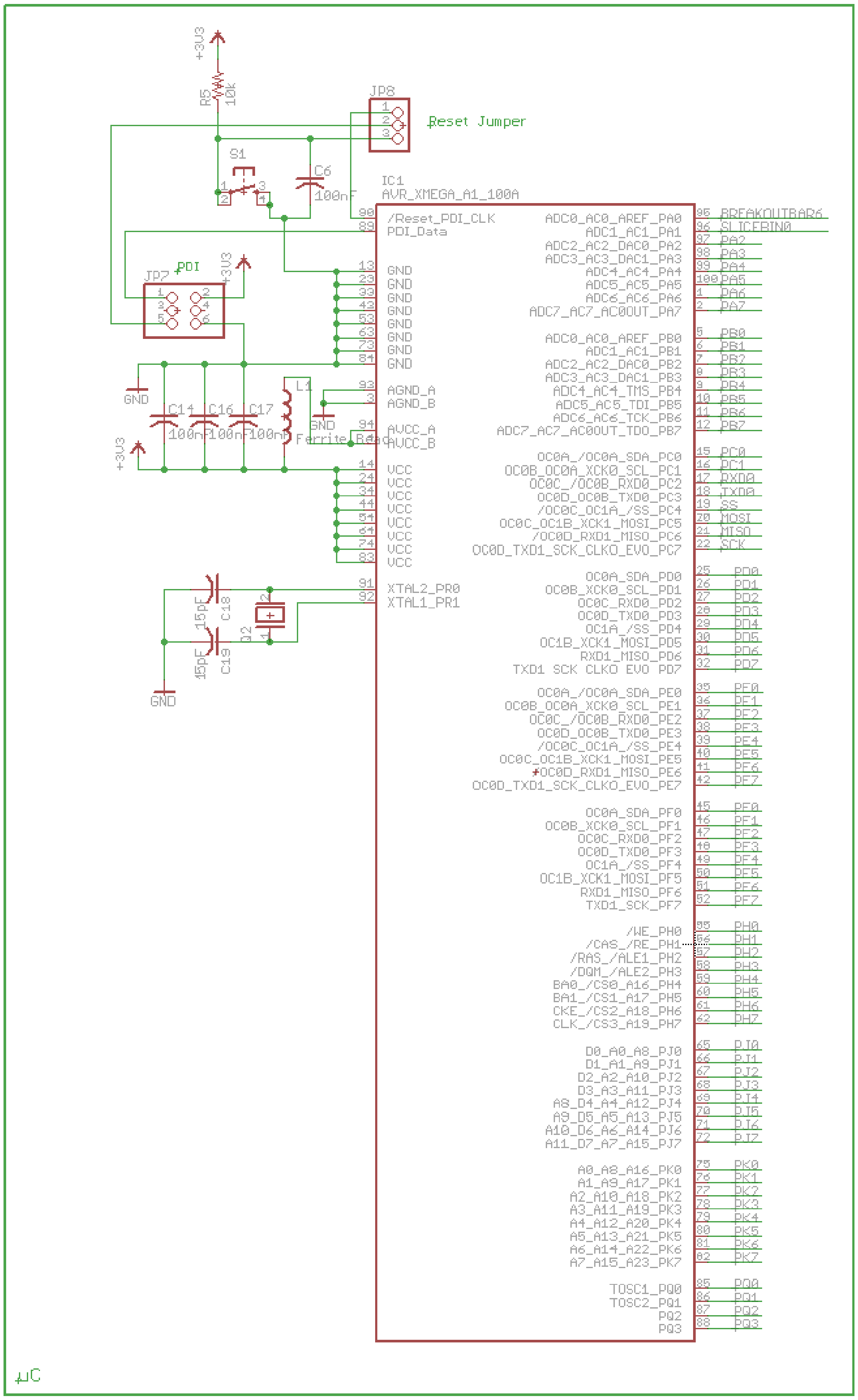


Figure 13. Microcontroller circuit.

The microcontroller circuit includes a 6-pin PDI header to program the device using an AVR ISP mkII. A reset button and 10K pull up resistor is included, with a jumper to switch the PDI reset line between the button and the PDI programming header. There are decoupling capacitors for each power line, and a 50 ohm ferrite bead to isolate the analog voltage reference (helpful, though not necessary for the final design which primarily uses analog comparators and not the ATXMEGA's on-chip ADCs).

The PCB includes a pad for an external crystal oscillator and corresponding 15 pF capacitors, although the microcontroller worked fine using the built-in 32 MHz clock. This layout was designed with reference to the XMEGA schematic checklist [6].

## External Bus Interface (EBI) and SDRAM

The imager has a resolution of 400x384, with 48 individual pixels comprising each macro-pixel of the output image. This means the frame size for an image with 8 bits of precision per pixel is

$$400 \times 384 \times 48 \times 8\ bits = 7.4\ MB$$

Since the XMEGA line of microcontrollers only has up to 32 KB of internal SRAM, it's necessary to expand the capabilities of the XMEGA by using the External Bus Interface (EBI) to add more memory.

I relied heavily on Atmel's designs for their XPlain evaluation module for the ATXMEGA128A1 [7], which interfaces the XMEGA with 8 MB of SDRAM using the MT48LC16M4A2 chip manufactured by Micron. Because that chip is now obsolete, I used the updated MT48LC4M16A2 chip, which is virtually identical but with a 64 MB capacity. The SDRAM and EBI circuit is as follows:

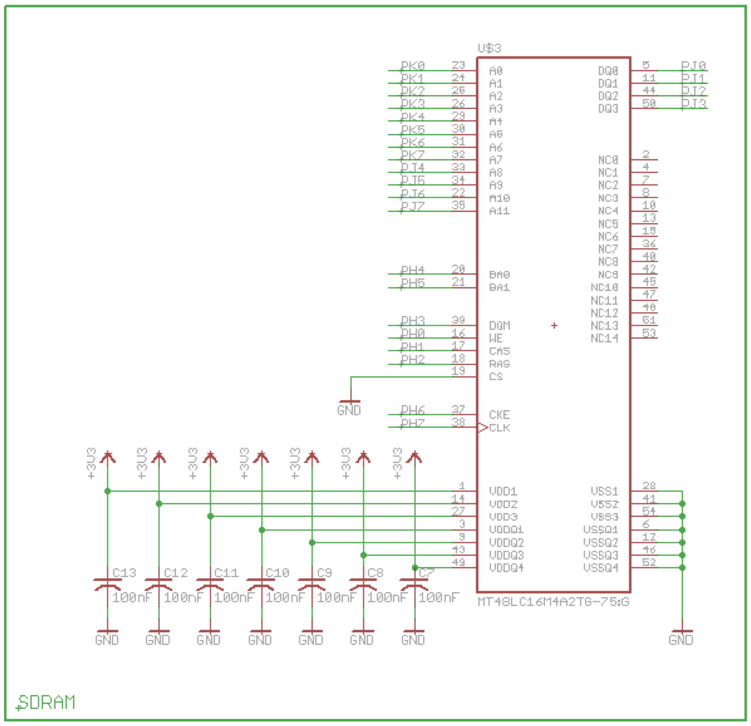


Figure 14. SDRAM and EBI circuit.

The SDRAM module is connected to PORTH, PORTJ, and PORTK, which are specifically designated as an interface for external memory. The choice of the x4 SDRAM module was driven by both the XPlain schematic and many complaints online about problems with the XMEGA's EBI, including documentation problems like the "phantom PORTL" and limitations on the type of devices that work properly in 3-port mode [8].

Unfortunately, problems accessing SDRAM occurred that caused certain addresses to be inaccessible. I started with drivers provided by the Atmel Software Foundation [9], but found that the test routines consistently failed for certain addresses in a regular pattern. The source of this problem was not found after extensive debugging, but it's likely related to either some difference between the MT48LC4M16A2 and the XPlain hardware that the ASF code was designed for, or a problem with the XMEGA's EBI. The short-term workaround possibilities were to develop a lookup table of usable addresses and use that for the main image capture loop, or to write the data more frequently to the SD card (resulting in a slower frame rate). At this time, the only working routine uses rapid SD card access. To recreate the driver issue, just change the bounds on the addressing loop in `SDRAM.c` (as explained in the comments).

If the driver issue cannot be resolved (or if a hardware-related issue is found to be the source of the problem), the easiest solution would be to use an SDRAM module that supports an SPI interface at a clock rate that is fast enough to write one byte (while also performing the conversion and multiplexer timing) within the 1.28 µs window for each pixel. It may also help to use a more powerful microcontroller, although there are none commercially available that have over 7.4 MB of internal RAM. The ARM Cortex M3 processor is closest, and there are some breakout boards available that have an additional SDRAM package built in (e.g. the G120 by GHI Electronics[1]).

## USB

To configure and test the camera from a computer, an FT232RL chip is included to perform USB-to-Serial conversion and interface with the USART on the ATXMEGA128A1. The circuit is as follows:

[1] http://www.ghielectronics.com/catalog/product/373

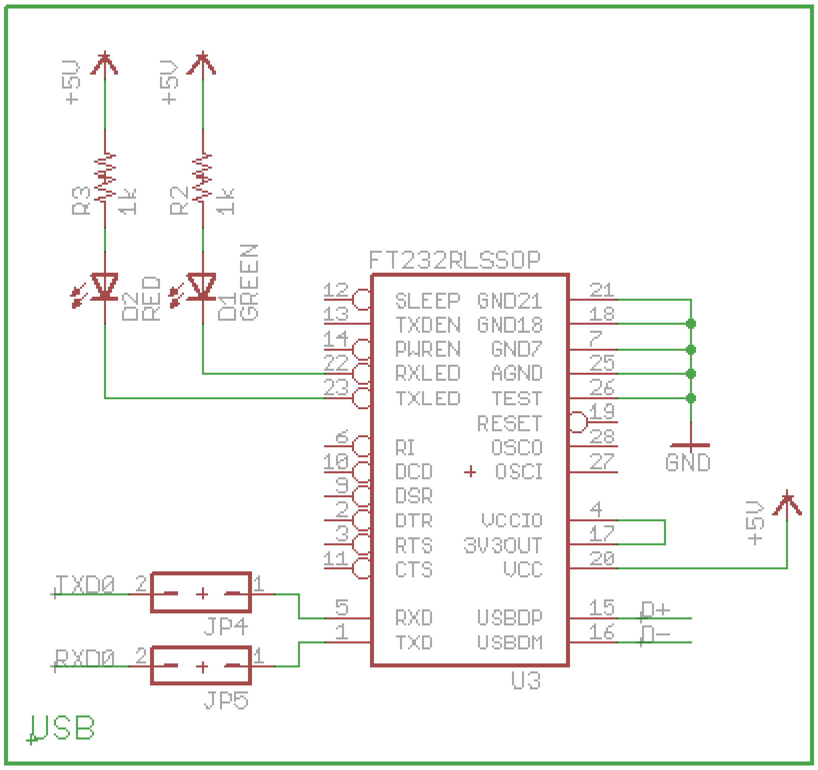


Figure 15. USB to Serial circuit.

The circuit is based on the recommendations of the FTDI datasheet [10]. Two indicator LEDs are connected to the RX and TX lines to show when data transfer is occurring. The TX and RX lines are connected to the microcontroller through two jumpers to optionally disconnect the FTDI chip.

The FT232RL enumerates as a USB device, and drivers are available for almost all operating systems. The control interface can then occur using any serial communication program, e.g. PuTTY, xterm, etc. This is described in more detail in the software section of this report.

## SD Card

A socket for an SD card is included to store images. The microcontroller uses SPI to communicate with the SD card, which operates at CMOS logic level. The circuit is as follows:

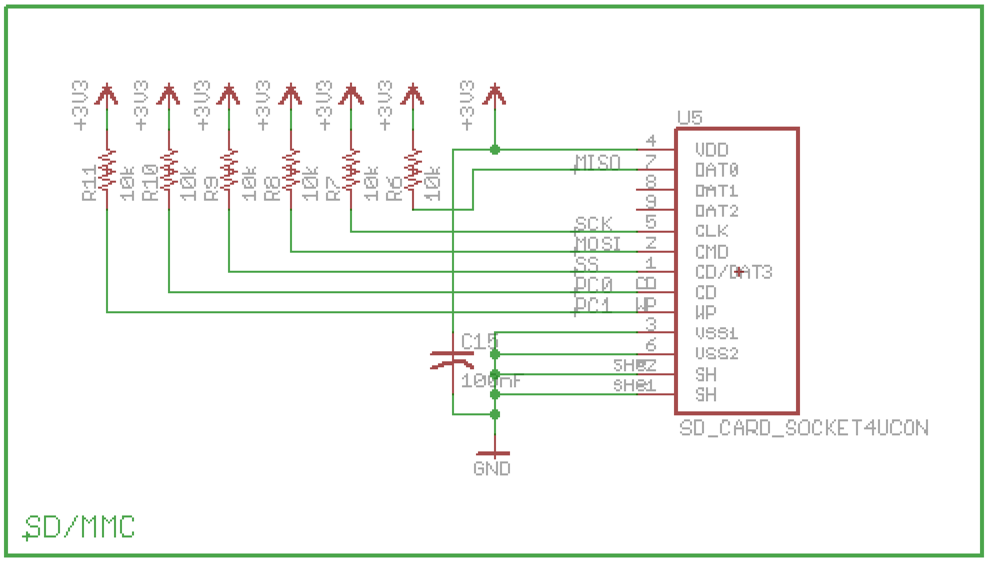


Figure 16. SD Card circuit.

The pinout and choice of 10K pull up resistors are based on a tutorial by Frank Zhao [11].

## Battery

The camera is designed to be powered solely by a 2000 mAh polymer lithium ion battery. A charge controller is included to charge this battery over USB, utilizing the MCP73831 which measures the battery voltage and sources a nominal 4.2 V to charge the battery up to its recommended limit of 3.7 V, at which point it cuts off the supply to prevent the battery from overheating or exploding. In addition, an LDO voltage regulator is included to provide the 3.3 V necessary to operate the microcontroller and the other ICs on the board. The circuit is as follows:

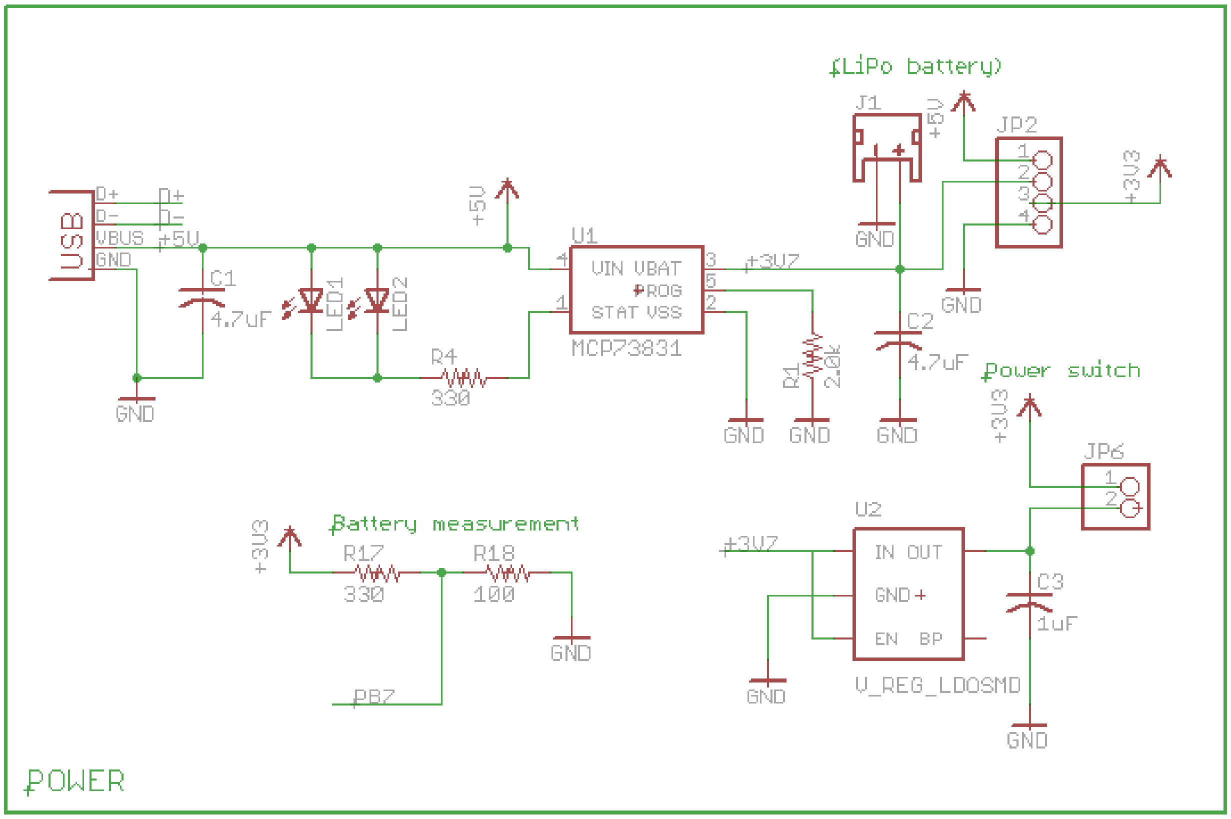


Figure 17. Power management circuit.

Pin B7 is also connected to the battery voltage through a resistive voltage divider. This is used to measure the battery voltage so that the microcontroller can shut itself off if the battery voltage drops below a usable amount, and to alert the user that this is happening.

The charge controlling circuit is adapted from a design by Sparkfun [12].

## Touchscreen

To facilitate a simple user interface, a touchscreen-enabled TFT LCD is included. This could act as a viewfinder for the camera, and to allow more advanced features like a self-timer, previewing computational refocus, changing settings, etc. The circuit is as follows:

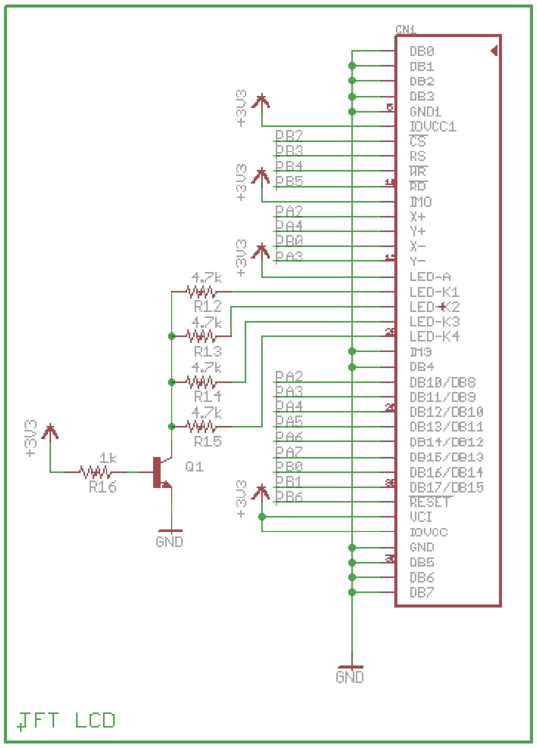


Figure 18. TFT LCD circuit.

The touchscreen is designed to work with an ILI9325 controller. The input data is sent using an 8-bit asynchronous interface. The resistive touchscreen can be accessed using the X± and Y± lines. The backlight LEDs are connected to 3.3 V through a transistor and current-limiting resistors.

## Lens Hardware

A design for a lens mount was created using Sketchup, a simple 3D modeling program. The part is designed to accept a C-mount lens (nominally 25.4 mm diameter) which is commonly used for small CCTV/security camera lenses.

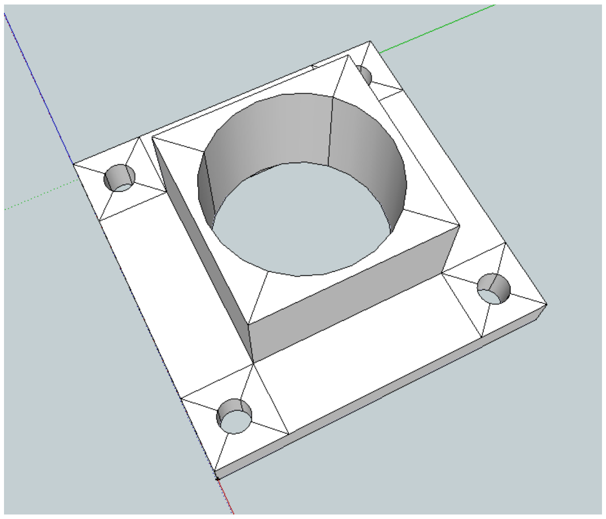


Figure 19. CAD drawing of the PCB lens mount.

The part fits around the imager IC and can be screwed into the inner set of mounting holes on the top board. The current design just holds the lens in place and is not actually threaded; an improved design would have a threaded mounting hole. Future improvements could also include an adjustable spacing mechanism and threaded screw holes.

The part was fabricated using a Makerbot Replicator 2 with ABS plastic.

## Manufacturing

The PCBs were designed in Eagle and manufactured by Advanced Circuits. The components were ordered through Digikey, Mouser, and Sparkfun, and were all soldered by hand.

The camera is designed to be assembled by attaching the bottom board to the control board using bolts and spacers. The bolts need to have a diameter of less than 5 mm to fit the mounting holes on the board. The spacers just need to be long enough to allow sufficient room to connect the data line headers, which need to be connected using either jumper wires or a ribbon cable.

## Errata

The following issues were noticed after the PCBs were manufactured and assembled, and should be corrected in future iterations:

- The top Tyco connector in the bottom board is too close to one of the ICs. The board can still be assembled by using a Dremel to remove some of the plastic casing of the connector, but more space should be given.
- The potentiometer pads are too close together, making it impossible to solder in all three at the same time.
- The machine pin headers for the DAQmx are too close together, making it impossible to connect all four ribbons at the same time.

- The JST connector should be flipped vertically so that the battery cable runs away from the board.
- Additional ground pins should be broken out. The current setup has the ground pin broken out right next to the power pin, which is dangerous when probing because it creates a high risk of accidentally shorting the two pins.
- There is no power switch to disconnect the battery entirely without removing it from the JST connector. The JST connector is not designed for lots of plugging/unplugging.
- There is an occasional problem where the battery won't charge over USB from a computer if the FTDI chip doesn't enumerate properly (which will always happen if the battery is under the minimum voltage of the FTDI chip). You can get around this by having the AVR ISP mkII connected to the PDI header when charging, which will supply enough power to get the FTDI chip to enumerate properly. To solve this, the FTDI chip should be isolated from the battery charging circuitry.
- Optionally, a switching power supply could be implemented so that you can power the camera directly over USB instead of waiting for the battery to charge.
- The mounting holes should be increased in diameter to match the larger holes of the top board to fit standard optical table mounting screws. Use large vias instead of the mounting hole components in the default Eagle library.
- Consider using a standard ribbon cable header for the analog data lines instead of a line of SIL header pins.
- The lens mount should have more clearance to avoid touching the top of the imager IC package.
- In the clock board layout, the RF coaxial connection should be moved closer to the side of the board. (Eagle gives a DRC error when you do this, which should be ignored.)
- Consider merging the clock board with one of the other boards.

## Software Design

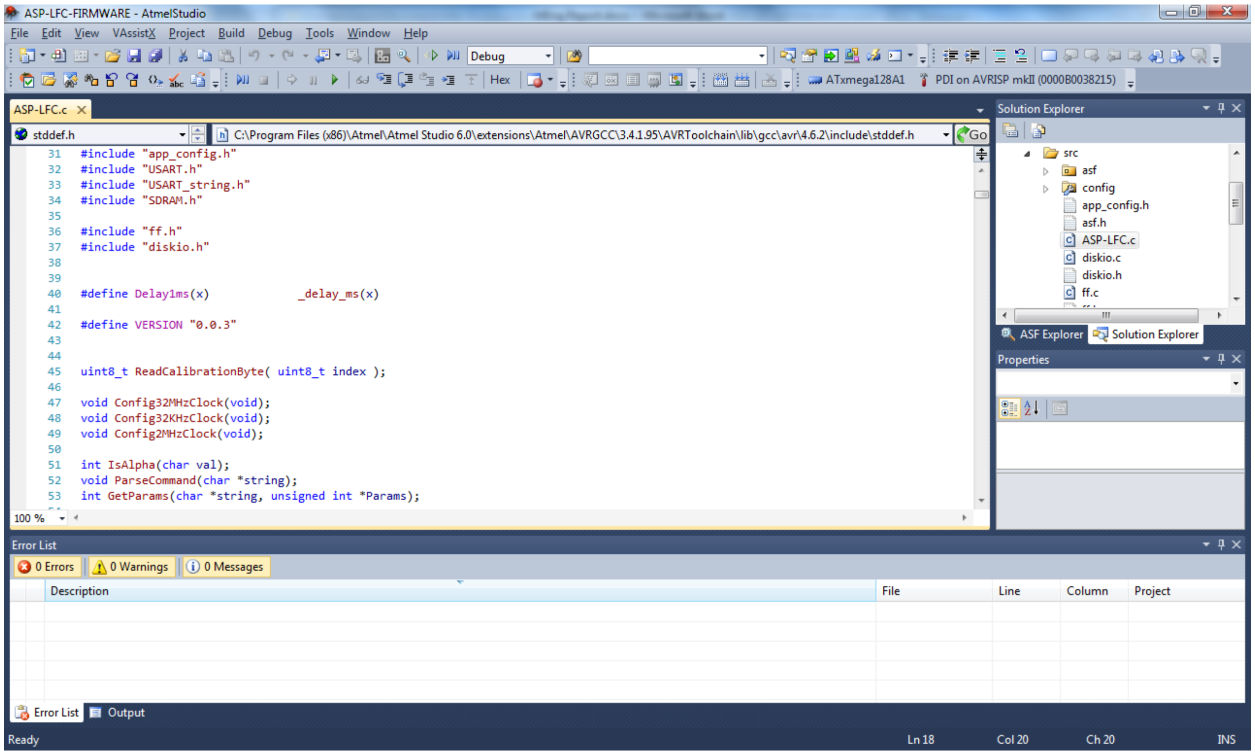
### Requirements

The main software requirement was to have a system capable of the following tasks:

- Rapidly saving the output of the ADCs to memory.
- Using the two digital counter lines (frame and pixel) to store the information sequentially.
- Managing a timing scheme to ensure information is read only at a time when the imager response is valid.
- Interfacing with the hardware peripherals on the board properly.

### Top Level Design

The software is written entirely in C and is designed to be compiled with avr-gcc. Specifically, Atmel Studio 6 and WinAVR were used for all software development and debugging.



```
#include "app_config.h"
#include "USART.h"
#include "USART_string.h"
#include "SDRAM.h"

#include "ff.h"
#include "diskio.h"


#define Delay1ms(x)             _delay_ms(x)

#define VERSION "0.0.3"


uint8_t ReadCalibrationByte( uint8_t index );

void Config32MHzClock(void);
void Config32KHzClock(void);
void Config2MHzClock(void);

int IsAlpha(char val);
void ParseCommand(char *string);
int GetParams(char *string, unsigned int *Params);
```

**Figure 20. Atmel Studio development environment.**

To work with the software, open the project file (`ASP-LFC-FIRMWARE.atsln`) in Atmel Studio 6 and compile (which should automatically program the microcontroller with the compiled image as well). Make sure Atmel Studio is set up for at ATxmega128A1, in PDI mode, with the programmer set to the correct ISP (depends on the type of device you are using).

## UART Interface

The firmware is designed to support a serial UART interface for control & testing. The interface can be accessed using PuTTY or a similar serial communication terminal. The terminal should be set to communicate at 9600 baud, 8 data bits, 1 stop bit, no parity, and XON/XOFF flow control.

Upon reset, the camera will run through a number of test programs, including testing the SDRAM addresses and writing to the SD card (create a file called `asplfc.txt` in the root directory of the SD card, and the program will overwrite its contents).

Once the boot tests are complete, the user can enter single-letter commands to access various features of the camera. Here are the most useful commands:

— `a`: view result of immediate ADC conversion
— `b`: read status of signal lines
— `y`: start frame capture
— `z`: end frame capture

Other commands can also be used, see the `ParseCommand` function for more details.

## Imager Timing

Both digital counter signals, BREAKOUTBAR6 and SLICEBIN0, are connected to pins on port A. The firmware is written to trigger interrupts for high-to-low transitions on either signal. The interrupt service routines are of the following form, where `INT0` is the BB6 interrupt and `INT1` is the SB0 interrupt:

```
ISR(PORTA_INT0_vect)
{
    if (frame < 3) {
        frame++;
    } else {
        stop_interrupts();
    }
}

ISR(PORTA_INT1_vect)
{
    if (frame) {
        MUXincrement();
        ADconvert();
    }
}
```

The `ADconvert()` function only performs the steps necessary to trigger a conversion on the external ADCs. Once the memory issue is resolved, this should be expanded to save the converted data to an image file.

## Touchscreen Library

Gravitech offers a library for using the ILI9325 with AVR devices [13]. This was modified slightly to deal with the fact that my hardware setup splits DB10-17 across different ports, so an updated version can be found in the code repository for this project.

Unfortunately, there's a problem working with this touchscreen driver. The device responds to some instructions, indicating that the hardware setup seems to be valid, but doesn't identify properly as an ILI9325. This can be verified by running `TSLCDOutIns(TS_INS_START_OSC)`, which should then return 0x9325 after running `TSLCDInDat()` according to the ILI9325 datasheet.

## Troubleshooting

If the device won't program, here are some common problems to check for:

- The programmer should be connected to the computer and working.
- The power switch should be set to on. (There is a header on the PCB to control this, which is clearly labeled as PWR SWITCH. This should be connected to a jumper or a switch.)

— The battery voltage should be above 3.3 V. There are several points on the PCB where you can check this[2]. If it's not, leave the ISP connected and connect the camera's micro USB port to the computer to let the battery charge. The red LED should turn on to indicate charging, and will turn off when charging is complete.
— The PDI header should be connected properly (Pin 6, which is ground, is labeled on the PCB).
— The reset pin of the microcontroller should be connected to the PDI header and not the reset button (there is a labeled jumper on the PCB to control this).

If the device won't communicate over USB, here are some common problems to check for:

— The TX and RX jumpers should be in place.
— The microcontroller should be on and running the correct firmware.
— The serial communication program should be configured properly (see the UART Interface section above).

## Results & Conclusion

A prototype for a light field camera utilizing an angle-sensitive pixel imager was designed and built. The device is capable of successfully reading data from the imager, storing data to an SD card, interfacing with a computer over USB, and controlling the charge rate of an integrated polymer lithium ion battery.

While the hardware works successfully, there is additional software work to be done to capture a full frame in a reasonable amount of time, and to get the TFT LCD touchscreen to work properly as a user interface device.

The following images were taken using the prototype camera hardware. Because of the SDRAM issue, the NI DAQmx was used to take these images.

[2] To check the battery voltage directly, use the exposed right terminal of the JST connector or the pin labeled BAT on the power header (JP2, top right corner of the board). To check the board's regulated 3.3V source, use Pin 2 of the PDI header (bottom right), the left pin of the PWR SWITCH header, or the pin labeled 3V3 on the clock board.

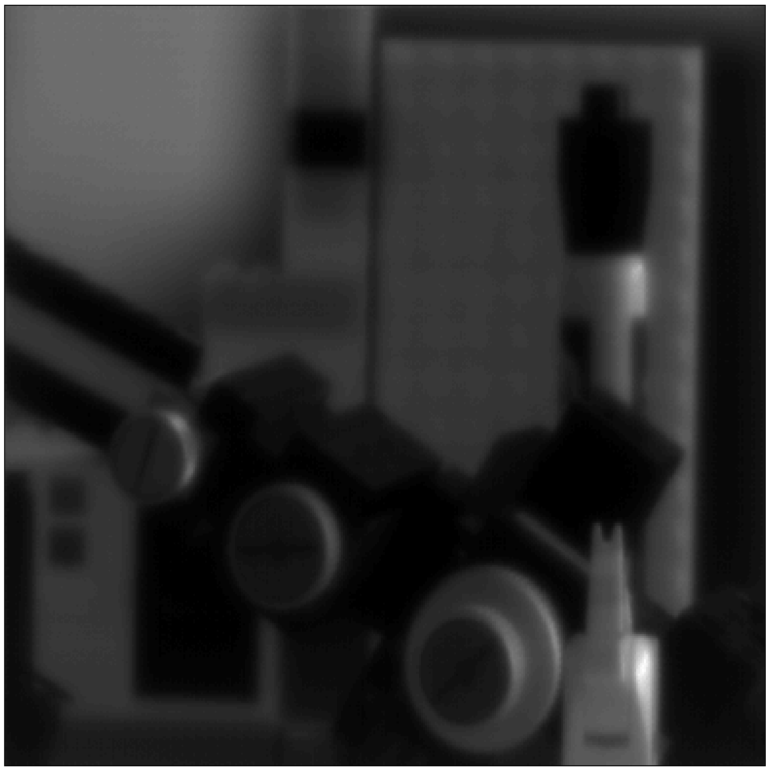


**Figure 21. Test image taken with the camera.**

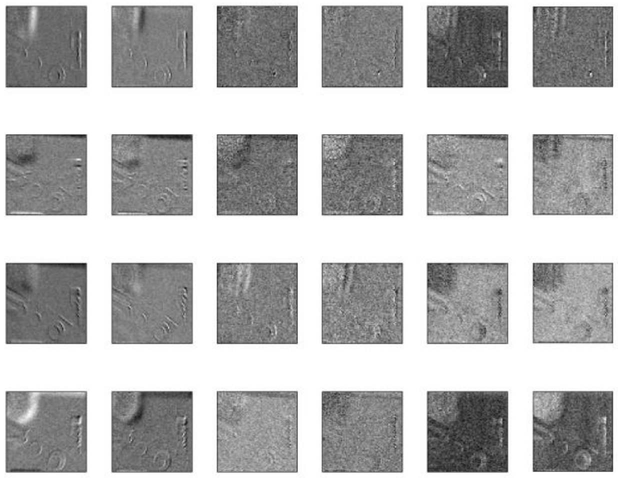


**Figure 22. Result of unpacking the test image into its various ASP components.**

Images of the completed hardware setup are available in appendix D.

## Other Options

It would be possible to build a similar device using an FPGA or a more powerful microcontroller, like a 32-bit ARM device. These devices may be less portable, but it may be worthwhile to build a device if the driver support can be made a lot simpler.

## Future Work

More work should be done in the following areas:

- Configuring the EBI to work with the onboard SDRAM module properly. It’s likely that the problem is that the ASF driver won’t work properly with the 64 MB package, and will require some modification.
- Getting the touchscreen to work properly.
- Fixing the hardware issues (described in “Errata” earlier).

## Works Cited

[1] A. Wang and A. Molnar, "A Light-Field Image Sensor in 180 nm CMOS," *IEEE J. Solid-State Circuits,* vol. 47, no. 1, January 2012.

[2] M. Levoy, R. Ng, A. Adams, M. Footer and M. Horowitz, "Light Field Microscopy," in *SIGGRAPH ACM Transactions on Graphics*, 2006.

[3] D. Lanman, G. Wetzstein, M. Hirsch, W. Heidrich and R. Raskar, "Polarization Fields: Dynamic Light Field Display using Multi-Layer LCDs," in *SIGGRAPH Asia*, 2011.

[4] Analog Devices, "AD9283 Datasheet," 2001.

[5] Analog Devices, "ADG726/ADG732 datasheet," 2002.

[6] Atmel Corporation, "AVR1012: XMEGA A Schematic Checklist," 2010.

[7] Atmel Corporation, "AVR1907: Xplain evaluation board, Xmega CPU Module," 2009.

[8] AVR Freaks, "AVR forum - XMega [expletive], sodding EBI, pftt, grrr.," 2008. [Online]. Available: http://www.avrfreaks.net/index.php?name=PNphpBB2&file=printview&t=68907&start=0. [Accessed 14 5 2013].

[9] Atmel Software Foundation, "Quick Start Guide for the XMEGA EBI Driver," 2013. [Online]. Available: http://asf.atmel.no/docs/latest/xmegaa/html/xmega_ebi_quickstart.html.

[10] Future Technology Devices, "FT232R USB UART IC datasheet," 2010.

[11] F. Zhao, "MMC/SD Card and FAT Tutorial," [Online]. Available: http://www.frank-zhao.com/cache/fat_sd.php. [Accessed 14 5 2013].

[12] Sparkfun, "USB LiPoly Charger - Single Cell," 2011. [Online]. Available: https://www.sparkfun.com/products/10161. [Accessed 14 5 2013].

[13] Gravitech, "2.8" TFT Color LCD 240 x 320 pixel," [Online]. Available: http://www.gravitech.us/2tftcolcd240.html. [Accessed 13 5 2013].

## Appendix A: PCB Schematics

### Bottom Board

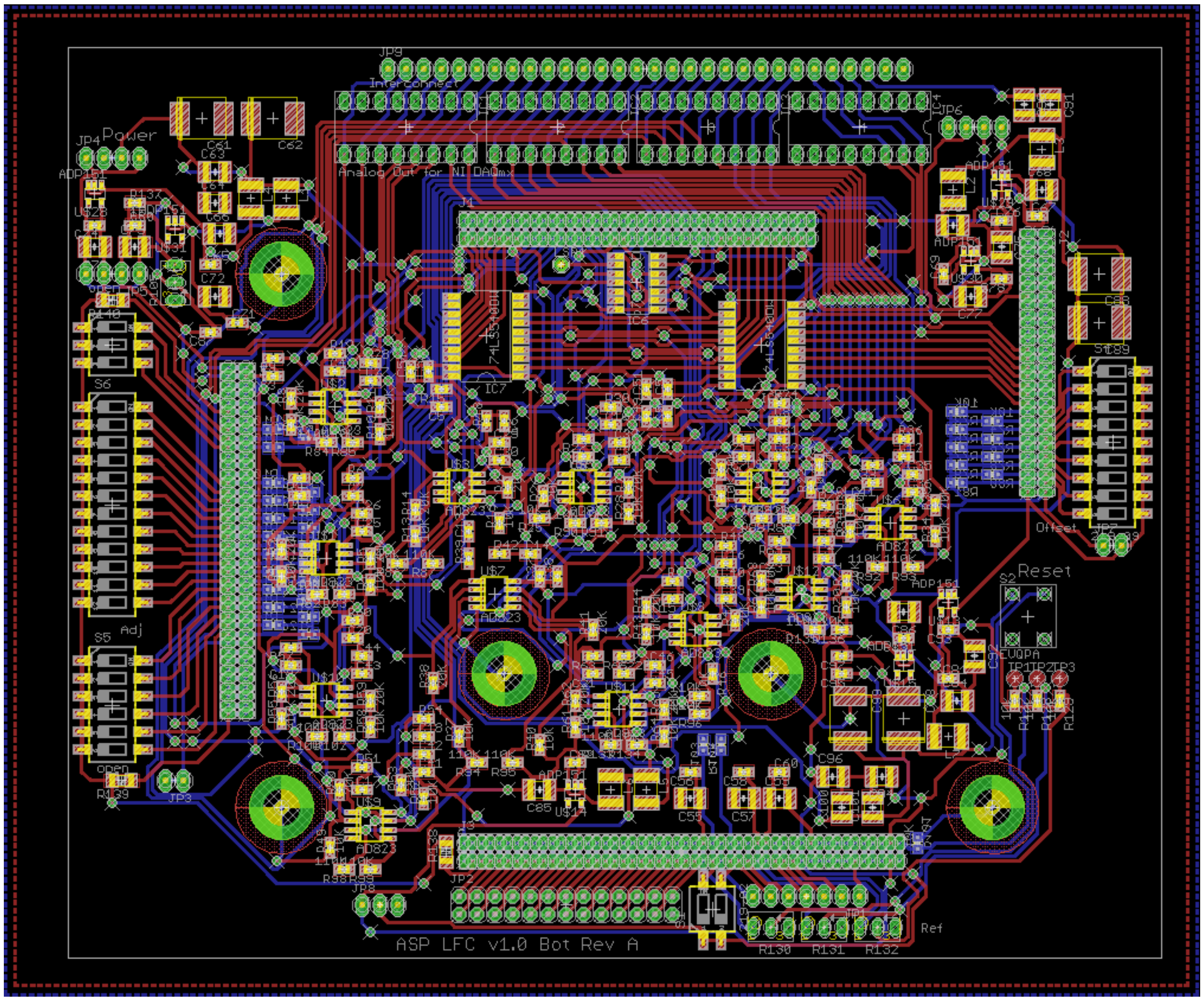


Figure 23. Layout of the bottom board.

## Control Board

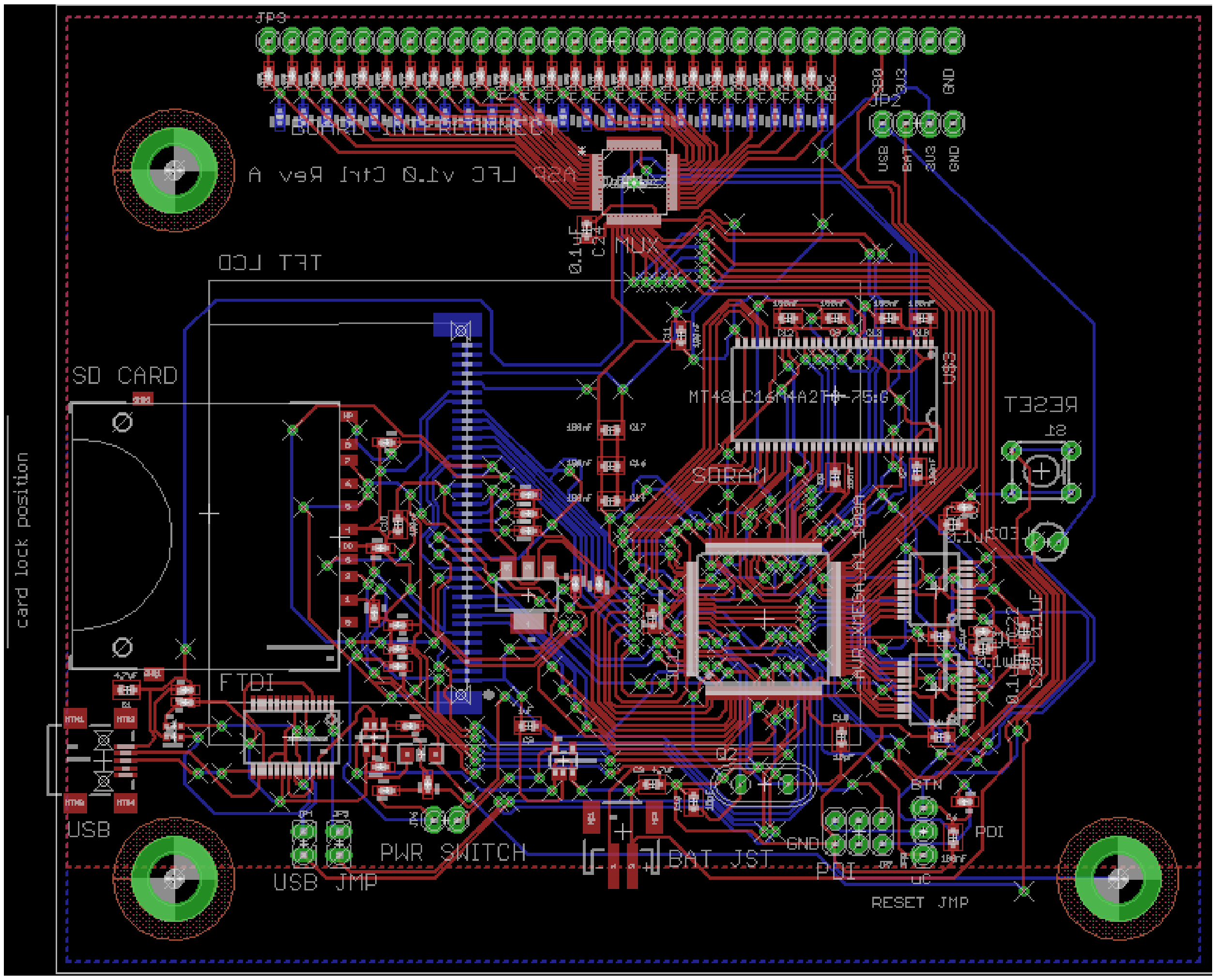


Figure 24. Layout of the control board.

## Clock Board

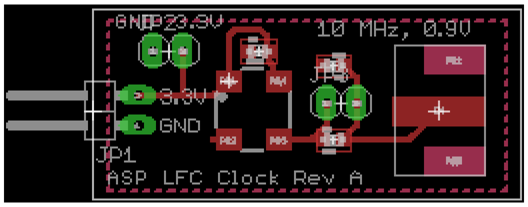


Figure 25. Layout of the clock board.

## Appendix B: Code Listing

### AVR Firmware

All source code is available at http://github.com/jpwright/asp-lfc

## MATLAB Scripts

### *adcCalculations.m*

```
% For Each ADC: (Hz)
adcSampleRate = 1/((12.8-7)/10000000);
disp(['Individual ADC Sample Rate: ' num2str(adcSampleRate/1000000) ' MHz']);

% Total ADC sample rate: (Hz)
totalAdcSampleRate = 24*adcSampleRate;
disp(['Total ADC Sample Rate: ' num2str(totalAdcSampleRate/1000000) ' MHz']);

% Assuming 8 bit external adcs:
eightbitExternalAdcMinCPU = totalAdcSampleRate;
disp(['8 Bit ADC Slave Sample Rate: '
num2str(eightbitExternalAdcMinCPU/1000000) ' MHz']);

% Assuming 9-16 bit external adcs:
sixteenbitExternalAdcMinCPU = totalAdcSampleRate*2;
disp(['16 Bit ADC Slave Sample Rate: '
num2str(sixteenbitExternalAdcMinCPU/1000000) ' MHz']);

% xmega onboard: number of CPUs in 1 frame
xmegaOneFrame = 24/(32000000/adcSampleRate);
disp(['Number of XMegas/frame required: ' num2str(xmegaOneFrame)]);
```

### *takePicture.m*

```
function takePicture(f)
    e=actxserver('LabVIEW.Application');
    vipath = 'F:\pictures\dumdumfull.vi';
    vi = invoke(e, 'GetVIReference', vipath);
    vi.SetControlValue('Path', ['F:\' f '.bin']);
    vi.Run(1);
    vi.SetControlValue('stop', 1);
    pause(2);
    file = f;
    unpacker
    reassembler
end
```

## Appendix C: Bill of Materials

### Bottom Board

| Device | Value | Package | Parts | Description | Qty |
|---|---|---|---|---|---|
| **C-USC0603** | | C0603 | C1-C97 | CAPACITOR, American symbol | 72 |
| **C-USC1210** | | C1210 | C55-C101 | CAPACITOR, American symbol | 23 |
| **CPOL-** | | E/7260-38R | C61-C99 | POLARIZED | 6 |

| | | | | | |
|---|---|---|---|---|---|
| **USE/7260-38R** | | | | CAPACITOR, American symbol | |
| **DIL16S** | | SOCKET-16 | IC1, IC2, IC3, IC4 | Dual In Line / Socket | 4 |
| **L-USL1812** | | L1812 | L1, L2, L3, L4, L5, L6, L7 | INDUCTOR, American symbol | 7 |
| **PINHD-1X2** | | 1X02 | JP3, JP7 | PIN HEADER | 2 |
| **PINHD-1X3** | | 1X03 | JP8 | PIN HEADER | 1 |
| **PINHD-1X30** | | 1X30 | JP9 | PIN HEADER | 1 |
| **PINHD-1X4** | | 1X04 | JP4, JP5, JP6 | PIN HEADER | 3 |
| **PINHD-1X7** | | 1X07 | JP1 | PIN HEADER | 1 |
| **PINHD-2X13** | | 2X13 | JP2 | PIN HEADER | 1 |
| **R-US_R0603** | | R0603 | R3-R126 | RESISTOR, American symbol | 53 |
| **TRIM_US-B64Y** | | B64Y | R105, R130, R131, R132 | POTENTIOMETE R | 4 |
| **R-US_R0603** | 10K | R0603 | R1-R103 | RESISTOR, American symbol | 28 |
| **R-US_R0603** | 10k | R0603 | R127, R128, R129 | RESISTOR, American symbol | 3 |
| **R-US_R0603** | 110K | R0603 | R82-R136 | RESISTOR, American symbol | 24 |
| **R-US_R0603** | 1k | R0603 | R137 | RESISTOR, American symbol | 1 |
| **R-US_R0603** | 20K | R0603 | R2-R71 | RESISTOR, American symbol | 24 |
| **219-02** | 219-02 | CTS-219-02 | S1 | Surface Mount DIP Switch Series 219 SMT | 1 |
| **219-03** | 219-03 | CTS-219-03 | S7 | Surface Mount DIP Switch Series 219 SMT | 1 |
| **219-06** | 219-06 | CTS-219-06 | S5 | Surface Mount DIP Switch Series 219 SMT | 1 |
| **219-09** | 219-09 | CTS-219-09 | S4 | Surface Mount DIP Switch Series 219 SMT | 1 |
| **219-12** | 219-12 | CTS-219-12 | S6 | Surface Mount DIP Switch Series 219 SMT | 1 |
| **74LS540DW** | 74LS540DW | SO20W | IC5, IC7 | Octal BUFFER and LINE DRIVER, 3-state | 2 |
| **74LS74D** | 74LS74D | SO14 | IC6 | Dual D type positive edge triggered FLIP | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | FLOP, preset and clear | |
| **AD823** | AD823 | SOIC-8 | U$1, U$2, U$3, U$4, U$5, U$6, U $7, U$8, U$9, U$10, U$11, U$12 | | |
| **ADP151** | ADP151 | TSOT23-5 | U$13, U$14, U$15, U$28, U$29, U $30, U$31 | | |
| **AMPMODU50 -100** | AMPMODU50 -100 | AMPMODU50 -100 | J3 | | |
| **AMPMODU50 -60** | AMPMODU50 -60 | AMPMODU50 -60 | J2 | | |
| **AMPMODU50 -80** | AMPMODU50 -80 | AMPMODU50 -80 | J1, J4 | | |
| **EVQPA** | EVQPA | EVQPA | S2 | Panasonic EVQPA tactile switch, 6.0mm x 6.0mm | 1 |
| **MOUNT-PAD-ROUND5.5** | MOUNT-PAD-ROUND5.5 | 5,5-PAD | H1, H2, H3, H4, H5 | MOUNTING PAD, round | 5 |
| **TPSPAD1-13** | TPSPAD1-13 | P1-13 | LSP1 | TEST PIN | 1 |
| **TPTP20R** | TPTP20R | TP20R | TP1, TP2, TP3 | Test pad | 3 |
| **R-US_R1206** | open | R1206 | R138, R139, R140 | RESISTOR, American symbol | 3 |

## Control Board

| Device | Value | Package | Parts | Description | Q ty |
|---|---|---|---|---|---|
| **ILI9325_LCD1.0MM** | | ILI9325_28I NCH_TS | CN1 | 2.8" ILI9325-based TFT LCD w/Integrated Touch Screen | 1 |
| **JP1Q** | | JP1 | JP4, JP5 | JUMPER | 2 |
| **JST_2MM_MALE** | | JST-2-SMD | J1 | Mates to single-cell LiPo batteries. CONN-08352 | 1 |
| **LED1206** | | LED-1206 | LED1 | LEDs | 1 |
| **LED3MM** | | LED3MM | LED2 | LEDs | 1 |
| **PINHD-1X2** | | 1X02 | JP6 | PIN HEADER | 1 |
| **PINHD-1X3** | | 1X03 | JP8 | PIN HEADER | 1 |
| **PINHD-1X30** | | 1X30 | JP3 | PIN HEADER | 1 |
| **PINHD-1X4** | | 1X04 | JP2 | PIN HEADER | 1 |
| **PINHD-2X3** | | 2X03 | JP7 | PIN HEADER | 1 |
| **TAC_SWITCHPTH** | | TACTILE-PTH | S1 | Momentary Switch | 1 |
| **TRANSISTOR_NPN"** | | SOT223 | Q1 | Generic NPN BJT | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | BT2222 | |
| **XTAL/S** | | QS | Q2 | CRYSTAL | 1 |
| **C-USC0603** | 0.1uF | C0603 | C4, C5, C20, C21, C22, C23, C24 | CAPACITOR, American | 7 |
| **330OHM1/10W 1%(0603)** | 100 | 0402-RES | R18, R45, R46, R47, R48, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59, R60, R61, R62, R63, R64, R65, R66, R67, R68 | RES-00818 | 25 |
| **C-USC0603** | 100nF | C0603 | C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17 | CAPACITOR, American | 12 |
| **2.0KOHM1/10 W5%(0603)** | 10k | 0402-RES | R5, R6, R7, R8, R9, R10, R11 | RES-08296 | 7 |
| **C-USC0603** | 15pF | C0603 | C18, C19 | CAPACITOR, American | 2 |
| **1K-1%** | 1k | 0402-RES | R2, R3 | 1k-ohm SMT | 2 |
| **2.0KOHM1/10 W5%(0603)** | 1k | 0402-RES | R16 | RES-08296 | 1 |
| **C-USC0603** | 1uF | C0603 | C3 | CAPACITOR, American | 1 |
| **2.0KOHM1/10 W5%(0603)** | 2.0k | 0402-RES | R1 | RES-08296 | 1 |
| **330OHM1/10W 1%(0603)** | 25 | 0402-RES | R19, R20 | RES-00818 | 2 |
| **330OHM1/10W 1%(0603)** | 330 | 0402-RES | R4, R17 | RES-00818 | 2 |
| **2.0KOHM1/10 W5%(0603)** | 4.7k | 0402-RES | R12, R13, R14, R15 | RES-08296 | 4 |
| **C-USC0603** | 4.7uF | C0603 | C1, C2 | CAPACITOR, American | 2 |
| **330OHM1/10W 1%(0603)** | 560 | 0402-RES | R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44 | RES-00818 | 24 |
| **AD9283BRSZ-80REAL** | AD9283BRSZ-80REAL | SSOP20 | U6, U7 | | 2 |
| **ADG726BSUZ** | ADG726BSUZ | SU_48 | U4 | | 1 |
| **AVR_XMEGA_A1_100A** | AVR_XMEGA_A1_100A | TQFP100 | IC1 | Device for Atmel AVR XMEGA A1 Series | 1 |
| **FT232RLSSOP** | FT232RLSSOP | SSOP28DB | U3 | USB UART | 1 |
| **INDUCTOR0603** | Ferrite Bead | 603 | L1 | Inductors | 1 |
| **LED-GREEN0603** | GREEN | LED-0603 | D1 | Various green LEDs | 1 |

| | | | | | |
|---|---|---|---|---|---|
| **MCP73831** | MCP73831 | SOT23-5 | U1 | Miniature single cell, fully integrated Li-Ion, Li-polymer charge management controller | 1 |
| **MOUNT-PAD-ROUND5.5** | MOUNT-PAD-ROUND5.5 | 5,5-PAD | H1, H2, H3 | MOUNTING PAD, round | 3 |
| **MT48LC16M4A2TG-75:G** | MT48LC16M4A2TG-75:G | TSOP54-400 | U$3 | IC SDRAM 64MBIT 133MHz | 1 |
| **LED-RED0603** | RED | LED-0603 | D2 | Assorted Red LEDs | 1 |
| **SD_CARD_SOCKET4UCON** | SD_CARD_SOCKET4UCON | SDCARD_4UCONN | U5 | SD Memory Card Connector | 1 |
| **USBSMD** | USBSMD | USB-MINIB | JP1 | USB Connectors | 1 |
| **V_REG_LDOSMD** | V_REG_LDOSMD | SOT23-5 | U2 | Voltage Regulator LDO | 1 |

## Clock Board

| **Device** | **Value** | **Package** | **Parts** | **Description** | **Qty** |
|---|---|---|---|---|---|
| **PINHD-1X2** | | 1X02 | JP2, JP3 | PIN HEADER | 2 |
| **PINHD-1X2/90** | | 1X02/90 | JP1 | PIN HEADER | 1 |
| **CAP0603-CAP** | 0.1u | 0603-CAP | C2 | Capacitor | 1 |
| **RESISTOR0603-RES** | 100 | 0603-RES | R2 | Resistor | 1 |
| **RESISTOR0603-RES** | 270 | 0603-RES | R1 | Resistor | 1 |
| **J658** | J658 | SMA_J658 | U$1 | | 1 |
| **OSCILLATORSMD** | 10MHz | CRYSTAL-SMD-7X5 | U1 | Generic 5x3 and 7x5 oscillators | 1 |

## Appendix D: Hardware Photos

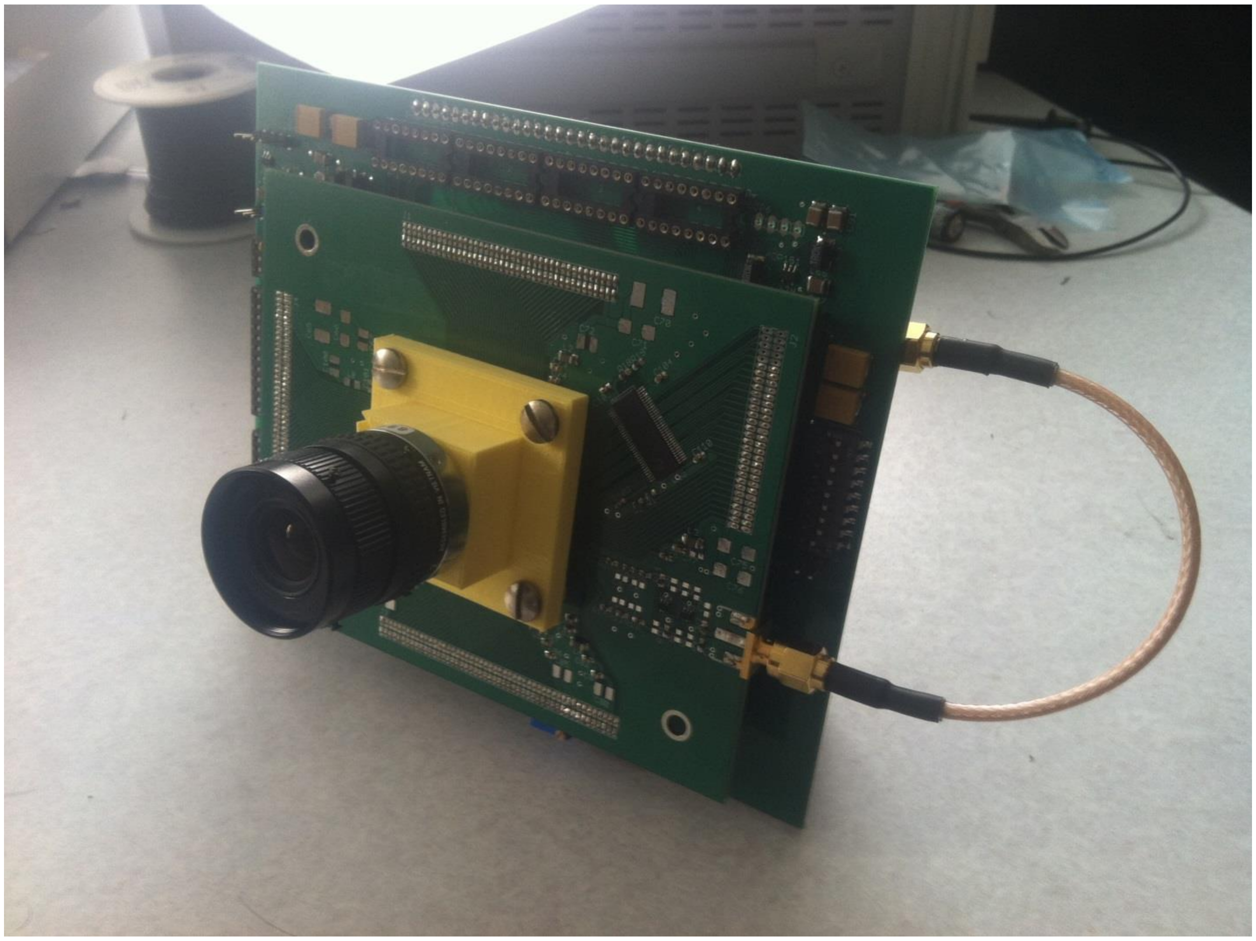


Figure 26. Front of the completed hardware.

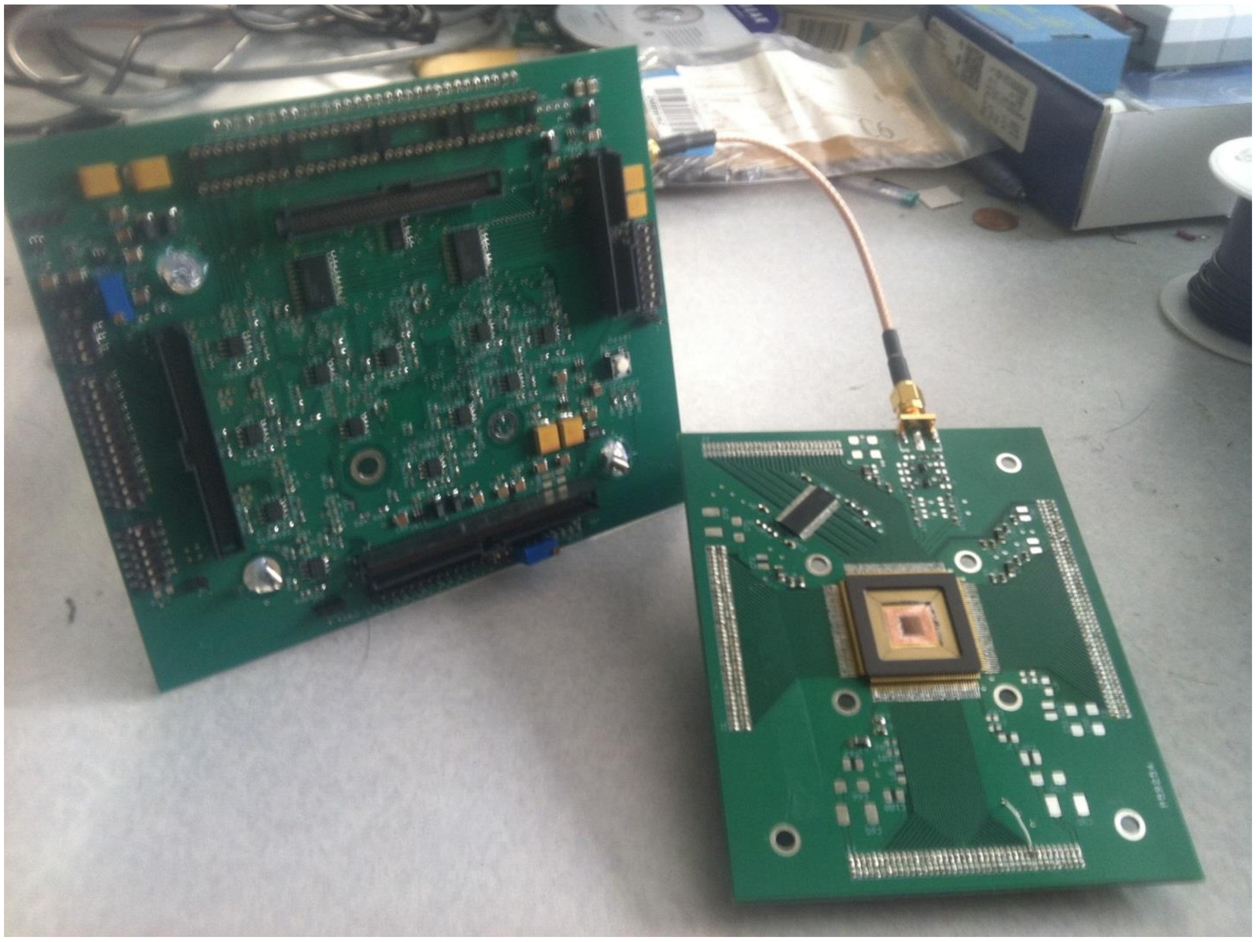


**Figure 27. Camera with imager board and lens removed and bottom board exposed. The clock board is attached to the underside of the bottom board.**

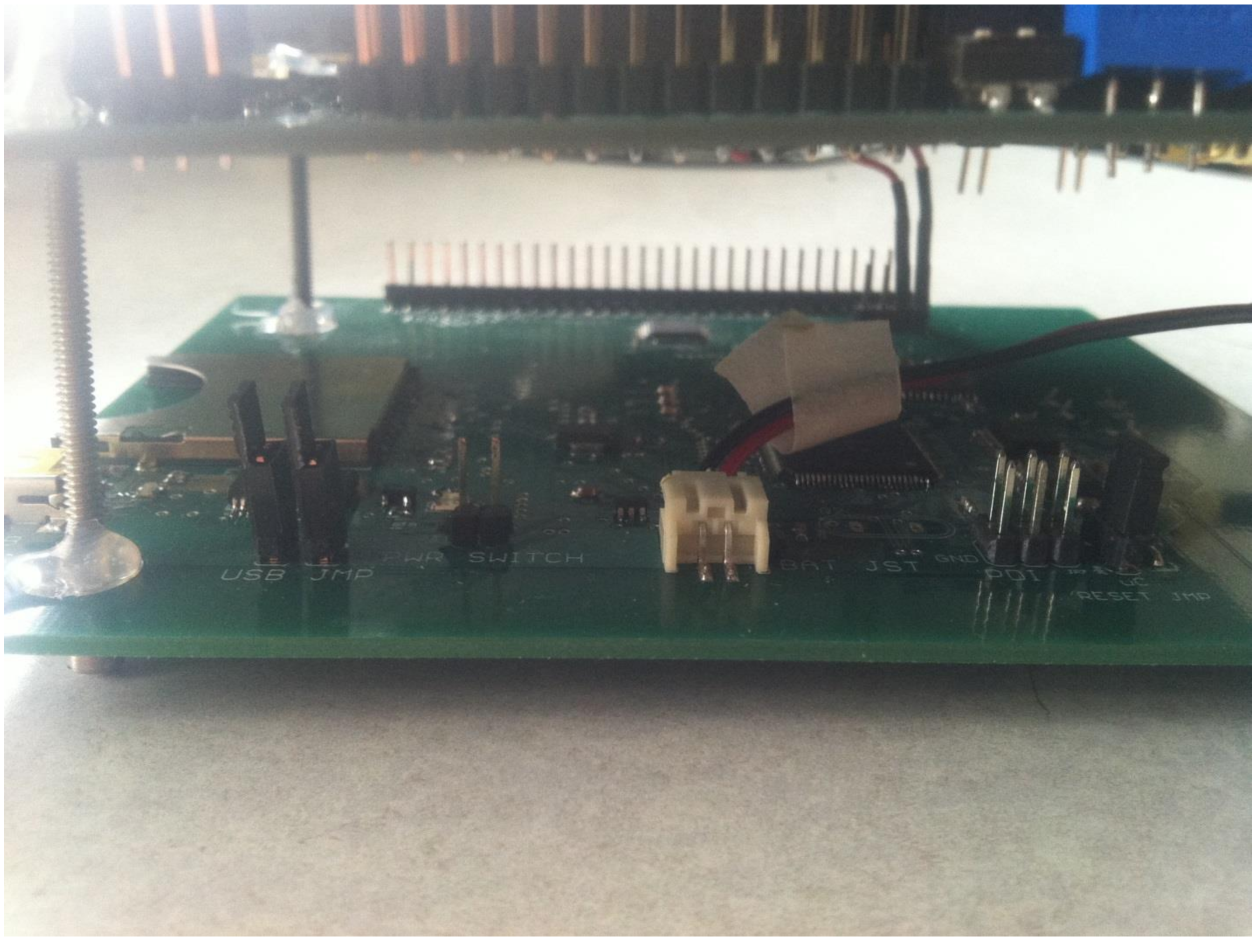


Figure 28. Close up of the control board from the side.