Adapting a CMOS Camera For High Speed Spatial Measurements

A thesis submitted in partial fulfillment of the requirement
For the degree of Bachelor of Science in Chemistry and
Physics from the College of William and Mary in Virginia,

by

Timothy Ray Leftwich

Williamsburg, Virginia
April 17, 2003

Abstract

We have adapted an off-the-shelf CMOS camera to image high speed laser-induced cavitation events. This requires hardware and software synchronization between the camera and the external lasers. In the current configuration this system can acquire 1 frame per second. The size of a frame is 640 by 480 pixels. Each pixel contains 1 byte of information for each of the three separate colors. The maximum frame rate for this camera is 60Hz. This paper discusses possible ways to improve the frame rate.

1. Introduction

Cameras are frequently used to collect two dimensional spatial data. Scientific cameras can be expensive. However, one can convert an inexpensive CMOS camera into a scientific camera. Other advantages of using a CMOS camera are that they are designed to interface with a computer and many can take pictures in color. However, unlike may scientific cameras, CMOS cameras have low frame rates, less than 100 frames per second. There are two major problems encountered in converting the CMOS camera into a scientific camera. The first is synchronizing it with the experimental equipment. In our application the camera's vertical synchronize signal runs the laser, i.e. they are synchronized. The other major problem is controlling the camera. The source of this problem is that there are many layers of software between the camera and the data file of the picture created in the computer. In our application, a 511+ chip is used to control the camera and allow the camera to be connected through a USB connection to a computer. C++ programs tell the 511+ to run the camera in single frame mode, which captures one frame per second, which is a small fraction of the maximum frame rate of the camera. Obviously the frame rate can be improved, but this requires changes in the software. This paper explains how the different parts of the camera work as well as how changes in each part affect the operation of the camera. This paper also details considerations in choosing the appropriate CMOS camera, converting it to a scientific camera and outlines some of the problems with synchronizing the camera.

The first step in designing an experiment is to choose the appropriate equipment. One must examine the experiment to determine what features the camera will need to have. For a camera some of the more important features to consider are the frame rate, the minimum amount of illumination required to collect data, the resolution, and if a color camera is necessary. If the frame rate that you require for your experiment is below 60 frames per second, an inexpensive CMOS camera may be an excellent replacement for a standard scientific camera. Our experiment involves taking several time sequenced pictures of explosions in liquid caused by laser pulses from the second harmonic of a Nd:YAG. The lasers fire at a frequency of 10Hz; although, the laser can run a couple hertz faster or slower if needed. The laser pulse is about 2 ns in length. In order to try to optimize the amount of data collected one would want a camera that can take pictures at a frame rate of 10 frames per second. The camera that I chose has a maximum frame rate of 60 frames per second. The second harmonic of a Nd:YAG laser is also used to pump one of two dye lasers, one producing red light and the other producing yellow light. There is a delay of 5 ns between the firing of the Nd:YAG laser and the red laser, which is followed by another 5 ns delay before the yellow laser fires. The experiment is also conducted in low light, the only source of light being the lasers. So the camera must be able to take pictures in low light. In order to take pictures in low light, the camera must have the ability to turn the auto exposure off. Also because the lasers are fired every 10 seconds the camera must have spatial resolution of approximately 30 μm. The size of each pixel on the image array for the camera I chose is 7.6 μm by 7.6 μm. In searching for an appropriate camera I found several CMOS cameras that fit the criteria each for less than $100. The CMOS cameras are inexpensive because they are normally used for video conferencing, which does not require the camera to be synchronized with any other piece of equipment. Therefore, because they are not designed for this purpose synchronization becomes a nontrivial problem. I decided to try two CMOS cameras, the ov7120, an omnivision 7000 series black and white camera, ($64.95) and the ov7620, an omnivision 7000 series color camera ($68.95) [1].

2. Synchronizing with the Laser

When the camera is used as a video camera it occasionally drops frames and has a maximum frame rate of about 30Hz. Therefore, for our laser application synchronization becomes a key issue. There are two ways to synchronize the camera. The first way is through the hardware; either a signal from the hardware of the camera triggers the laser or a signal from the laser triggers the camera. The second is to synchronize through the software; a command in a program triggers the camera and the laser. With hardware synchronization, the frequency of the pixel clock of the camera determines the frequency that the equipment is triggered. One must be able to operate the clock of the camera at a frequency that is in the range that the equipment can operate, in our case around 10 frames per second.

Synchronizing through the hardware usually requires custom designing electrical circuitry. Depending on the experiment, this may require a complicated circuit connected to several pieces of equipment or a simple circuit connected to the camera and one other piece of equipment. In our experiment, the camera can be connected to the laser with a simple circuit. Synchronizing through the software does not require that one build any additional circuitry, but it requires modifying software that may be complicated. The advantage to this is that the camera can be triggered independently, regardless of the clock frequency of the camera. The disadvantage is that when the software asks the camera to take a picture it can be anywhere in its period. This period consist of capturing the picture and transferring the information to video processor and then to the computer. The image capture is a fast process; however, the transfer of this image is not. It is possible, for example, that the camera can be in the part of its period where it has only transferred half of the data from the image array to the analog processing block of the camera. So, the camera will just give the last picture taken, which was before the laser fired. The experiment is currently hardware synchronized with the camera triggering the laser. This choice was made because the layers of software are very complicated.

3. Software Layout

The layers of software can make it difficult to synchronize the camera with other equipment. The firmware, the software inside the camera, is well documented but very complicated. Despite this complexity it may be possible to program an EZUSB (a piece of hardware that can be used to connect the camera to the computer through a USB connection) to control the camera. The EZUSB could possibly be programmed to set the registers of the camera, receive the output information from the camera, and process this information into a format that can be displayed on a computer. The details in the camera section will outline the information that one must know to do this. This may allow the camera to reach it's maximum frame rate. There is very little information about the software inside the 511+ chip (another piece of hardware that can used to connect the camera to the computer through a USB connection). Because there is limited information about the 511+, it is difficult to use it to control the camera. There are several layers of software that the picture must go through before it can be displayed on a computer screen, which can be seen in figure 1.

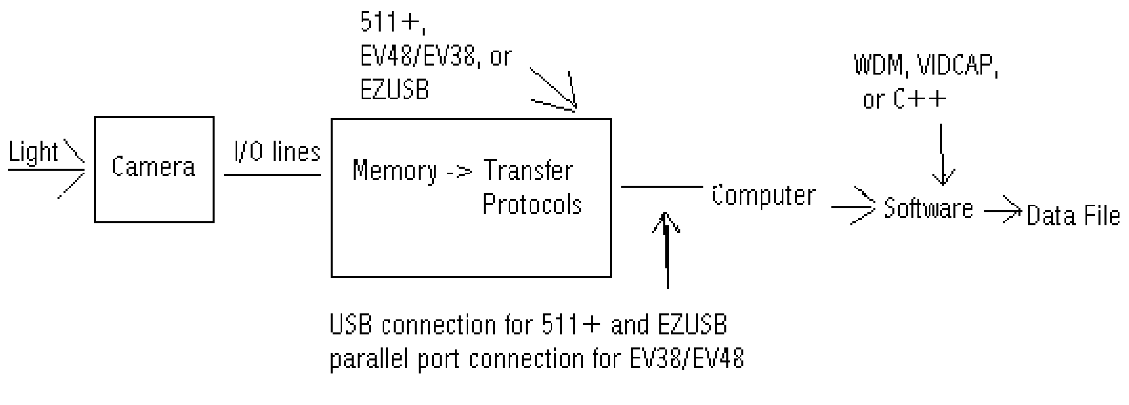


Figure 1: Above is a block diagram of the different levels of hardware.

First, the picture must be captured and processed by the camera. A detailed description of the software in the camera can be found in the camera section.

Second the output from the camera must pass to and commands must flow from the hardware interface to the computer. In our case this hardware is the 511+ chip, which comes on the same board as the camera. The camera is controlled and the information received through the I/O lines that are connected to a 511+ chip. The 38EV/48EV and EZUSB, other hardware that could be used instead of the 511+ chip, were considered but ultimately were not used. The software in the 511+ is undocumented. There has been some code written for Linux to control the chip; however, because of the complexity of the registers in the 511+ the code is incomplete.

Finally the information is transferred from the external hardware to the computer where programs save and display the data. The computer programs also give commands to the external hardware. The 511+ is connected to a computer through a USB line and information is transferred through this line as well as commands from the computer program to the 511+. Once the information is transferred to the computer, it is sent to a computer program where it is written as a data file. At first, the computer programs that came with the 511+, WDM and VIDCAP, were used. It was soon discovered that when using one of these programs if a frames was dropped during the transmission to the PC, the software padded the file with a duplicate of the previous frame to keep the video in real time. The frame drop rate was erratic and depended on the history of what other device drivers had been loaded into the computer prior to taking data. So, we wrote a custom C++ program. This program runs the camera in single frame mode, capturing an image roughly once a second. The program does this by using a function call to video for windows (an old program found in most computers that is used to control the drivers of a camera) that captures a single frame. The pictures are saved under a file name that includes the date and frame number. One can know when the frame was taken (date in the file name) as well as what frame it is (the frame number in the file name).

We use another custom C++ program to analysis the data. It can filter for red, blue or green light. This makes it possible to see the light emitted from the red and yellow lasers that illuminate the explosion. I later modified the program to save the filtered images. Brian and I also modified this program to display on a graph the intensities along a column or a row of pixels of each of the three filtered colors individually [5].

3. The Camera

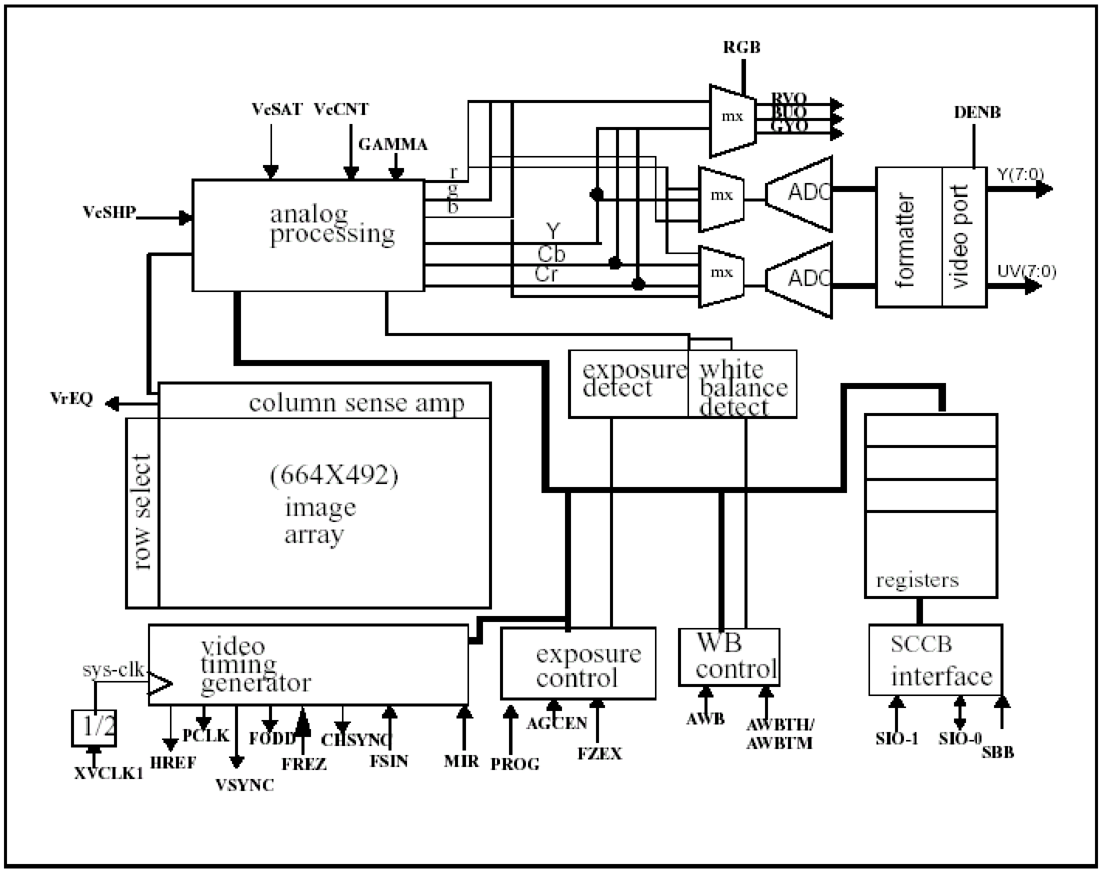


Figure 2: Above is the block diagram of the camera.

There are eight major parts of the camera, the image array, the video timing generator, the analog processor, the analog to digital converters, the output formatter, the digital video port, the SCCB bus, and the registers. The image array captures the image and sends it to the analog processor. The video timing generator controls the internal timing of the camera as well as the timing of the output. The analog processor performs all analog image functions. The analog to digital converters convert the analog output from the analog processor to a digital signal. The signal is sent to the output formatter where it is formatted and sent to the digital video port. The digital video port sends the signal output out of the camera. The SCCB bus provides input and output to control or read the registers, which control all camera functions.

In order to synchronize the camera with the lasers, insure the proper function of the camera, and improve the frame rate, one must understand how the camera works. To synchronize through the software, one must understand the period of the camera, as described in the synchronization section. In order to synchronize through the hardware, one must know which camera output or input to use and how it relates to the internal timing of the camera. One can have the camera fully synchronized and still not functioning properly! One problem that I encountered when I ran the camera was that frames would be randomly dropped an duplicated when using WDM or VIDCAP. One way to try to improve the frame rate would be to replace the 511+ chip with an EZUSB.

4.1 Image Array

The image array captures the initial image. The size of the array is 664 pixels by 486 pixels. With the default values of the registers the effective size of the array is only 640 pixels by 480 pixel. The output formatter section has more information about this. Each pixel is composed of a small lens (which focuses the light) and a color filter. There are three colors, red, green and blue that the color filters allow to pass. The light from each pixel then shines on a photo diode, which measures its intensity. Capturing an image involve measuring the intensity in each pixel and sending it to the analog processing block. The array is in a pattern called a Bayer-Pattern as shown in figure 3.

R\C	1	2	3	4		641	642	643	644
1	B	G	B	G		B	G	B	G
2	G	R	G	R		G	R	G	R
3	B	G	B	G		B	G	B	G
4	G	R	G	R		G	R	G	R
5	B	G	B	G		B	G	B	G
481	B	G	B	G		B	G	B	G
482	G	R	G	R		G	R	G	R
483	B	G	B	G		B	G	B	G
484	G	R	G	R		G	R	G	R
485	B	G	B	G		B	G	B	G

Figure: 3 Above is the physical representation of image array of the camera. Where R, G, and B stand for pixels filtered for red, green, and blue light respectively.

4.2 Timing Generator

The timing generator controls all timing, including the pixel clock, the horizontal synchronize signal, the horizontal window reference output, and the vertical synchronize signal. The clock of the camera is provided by pin 27, Xclk1. The camera specifications recommend 27MHz as the camera frequency. The maximum frequency is 30MHz. The camera can be run at lower frequencies; however, this decreases the clock frequency, which is used to time everything. Practical concerns limit the clock frequency. The two major practical concerns are the maximum frame rate and the exposure time. The output can be formatted in several different ways; 16 bits or 8 bits, which are timed differently. The output can be progressive or interlace. These different output formats are also timed differently, with frames divided differently in interlace and progressive modes.

There are several timing output lines from the timing generator that relay information about which pixel is being sent from the digital video port. These outputs are the pixel clock (pin 33) the horizontal synchronize signal (pin 42) the horizontal window reference output (pin 18) and the vertical synchronize signal (pin 16). The polarity of these outputs can be changed by the registers. The pixel clock operates at the camera clock frequency in 16 bit mode and at twice the camera frequency in 8 bit mode. The pixel clock indicates when information about a pixel is being outputted on each period of the clock. In 16 bit format information about one pixel is sent in one clock period. In 8 bit format information about one pixel is sent in two clock periods. In the default setting, data is outputted at the falling edge and is stable at the rising edge. The data is outputted one row at a time. The horizontal synchronize signal indicates the start and the end of each row by allowing its output to go low. In the default setting, data starts to output when the horizontal synchronize signal is at the falling edge. The horizontal window reference indicates when a row is being outputted by allowing the output to go high. In the default setting, the data starts to output when the horizontal window reference is at the rising edge. The vertical synchronize signal indicates the start and end of a frame in progressive mode and the start and end of a field in interlace mode. Output is when it is low. In the default setting, the data starts output when the vertical synchronize signal is at the rising edge.

During 16 bit format the output is at the clock speed on 16 lines. These lines are the Y lines (pins 34-41) and the UV lines (pins 19-26). In the 8 bit format, the output is at twice the clock speed and on 8 lines (the eight lines are the Y lines).

In interlace mode (the default mode) the frame is divided in half. The two halves are called frames. There is an odd frame and an even frame. In interlace mode there is an additional output. This output is the odd field flag (pin 17). It is held high during the odd field and low during the even field. In progressive mode there is no division of the frame.

The Frame division is controlled by register 16. The last 6 bits control the separation between the frame or field. The number in the last 6 bits is the number of empty frames or fields before the next one. In interlace the first 2 bits in register 16 indicate which field the horizontal window reference is asserted. If the first two bits are 01 then the horizontal window reference is asserted in the odd field only. If they are 10 then the horizontal window reference is asserted in the even field only. If it is 11 the horizontal window reference is asserted in both fields and the first 6 bits are ignored. In progressive mode the if the first two bits are 10 or 01 the horizontal window reference is asserted in frames indicated by the last 6 bits. If it is 11 the horizontal window reference is asserted in all frames and the last 6 bits are ignored. In both modes if the first 2 bits are 00 then the horizontal window reference is not asserted at all, except in single frame transfer mode.

Single frame transfer mode is initiated by register 13 the second bit. When this bit is high it initiates single frame transfer. The first two bits of register 16 must be 00 before single frame transfer can be initiated. The first two bits of register 16 and the second bit in register 13 are automatically cleared after one frame transfer.

4.3 Analog Processor

The analog processor performs the automatic gain control and automatic white balance, and controls the image quality. These controls include sharpness, hue, gain, gamma control, and anti-blooming. Gain is discussed in the register section. The gamma control allows a non-linear relationship between the data and the output. The analog signals that are standard video signals come out of the analog processor are based on the following formulas:

$$Y = 0.59G + 0.31R + 0.11B \quad (1)$$
$$U = R - Y \quad (2)$$
$$V = B - Y \quad (3)$$
$$Cr = 0.713 \times (R - Y) \quad (4)$$
$$Cb = 0.564 \times (B - Y) \quad (5)$$

Where R, G, and B stand for the red, green, and blue pixel colors respectively and where Y, U, V, Cr, and Cb are output signals from the analog processor.

4.4 Analog to Digital Converters

The analog to digital converters convert the analog signal to a digital one. There are two ten bit analog to digital converters. They both have a muliplexer before them to allow several lines of information from the analog processor to be outputted on one line from each muliplexer.

4.5 Output Formatter

This is where output is formatted. The format includes the number of bits per pixel, the number of pixels, the designation of the 16 output lines, and interlacing. This camera can use a wide variety of standard video conventions. These conventions are summarized in figure 4.

		Interlaced		Progressive	
Resolution		640x480	320x240	640x480	320x240
YUV 4:2:2	16Bit	Y	Y	Y	Y
	8Bit	Y	Y	Y	Y
	CCIR656	Y	Y	Y	Y
RGB	16Bit	Y	Y	Y	Y
	8Bit	Y	Y	Y	Y
	CCIR656	Y	Y	Y	Y
Y/UV swap	16Bit	-	-	-	-
	8Bit	Y	Y	Y	Y
U/V swap	YUV	Y	Y	Y	Y
	RGB	Y	Y	Y	Y
YG	16Bit	Y	Y	Y	Y
	8Bit	-	-	-	-
One Line	16Bit	-	-	Y	-
	8Bit	-	-	-	-

Figure 4 Above is a table of the types of video format. Y indicates that the format is supported by the camera.

The size of the window is determined by registers 17-1A. The window size is the part of the array that is scanned into the analog processor and subsequently displayed on the screen. Register 17 determines which column to start where register 18 determines which one to end. Each value represents four pixels in full resolution and two pixels in QVGA resolution mode. Register 19 determines which row to start where register 1A determines which row to end. The default values of registers 17 through 1A set the window size to 640 by 480 pixels in VGA format and 320 by 240 pixels in QVGA format. Whether the window output format is 640 x 480 pixels, VGA format (the default), or 320 x 240 pixels, QVGA format, is determined by the sixth bit of register 14.

4.6 Digital Video Port

The digital output consist of 8 lines in 8 bit format or 16 lines in 16 bit format. Currently each output line has a transfer rate around 115 kilobytes per second. The maximum transfer rate for the output lines is almost 7 megabytes per second. The output currents are adjusted automatically as a function of loading.

4.7 Serial Camera Control Bus (SCCB)

The Serial Camera Control Bus, or SCCB, is used to set the registers in the camera. The only way to change from the default values in the registers is to communicate through the SCCB, which uses three wires to transfer information, read from and write to the camera. The three wires are the SBB (pin 12), SIO-1 (pin 45), and SIO-0 (pin 46). The SBB is held low when values are read from and written to the camera. At all other times, the SBB is held high. The SIO-1 is the serial clock. The SIO-0 is the wire were the data is transferred one byte at a time. The SCCB can support more than one camera. When there is more than one camera attached to the master the cameras simply have different ID addresses. In the SCCB the camera or cameras act as the slave device or devices and the master is the device to which the camera is connected. For example, if the camera is connected to a 511+ then the 511+ is the master.

There are three types of transmission cycles. They are 3-phase write transmission cycle, 2-phase write transmission cycle, and a 2-phase read transmission cycle. A phase contains 9 bits. The first 8 bits are the data.

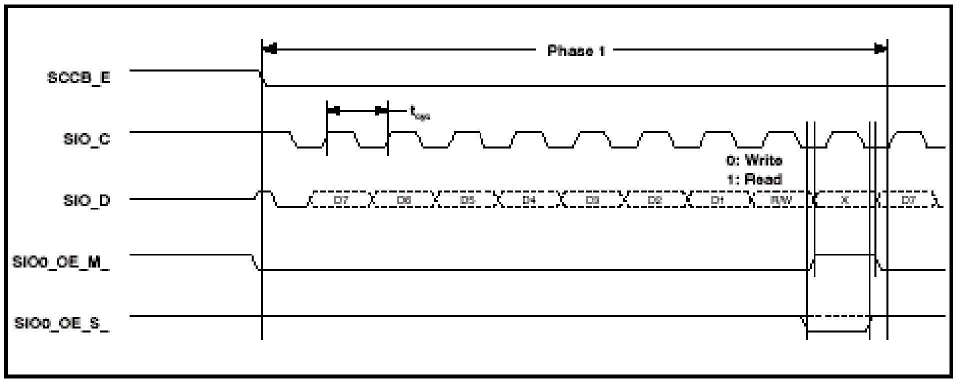


Figure 5: Above is a diagram that shows the timing of the first phase of a transmission cycle.

A 3-phase write transmission cycle is used to write one byte of data from the master to a register in a camera. The first phase of this cycle, and the other two as well, is the ID address of the camera. The second phase is the address of the register in the camera to which the data will be written. The third phase is the data that is being written to the camera's register.

The other two cycles are used to read data from a camera register to the master. Because the 2-phase read transmission cycle does not indicate the register that it is reading, it reads from the last register indicated in the previous write cycle. The 2-phase write transmission cycle is used to change registers before the 2-phase read transmission cycle. The 2-phase write transmission cycle consists of a camera ID address phase and a register address phase. The 2-phase read transmission cycle consists of a camera ID address phase and a phase that contains the data that is being read from the camera's register.

In the first phase, ID address phase, the ID address is the first seven bytes. The eighth byte selects whether this phase will be in a read or a write cycle. For a write cycle, the eighth bit is set low and for a read cycle, the eighth bit is set high. The first eight bits are asserted by the master. The ninth bit is an assertion from the camera. The camera asserts low if the data, the previous eight bits, has been written to it. It will assert high if it has not. The master does not check the value of the ninth bit. The master masks the input of the ninth bit to low. The default ID address for the camera, the first seven bits plus the eighth read/write bit, is 42 for write and 43 for read.

In the second phase of the write statements, the address of the register phase, the first eight bits are use to identify the register and are asserted by the master. The ninth bit is asserted in the same way as in the first phase.

In the second phase of 2-phase read transmission cycle, the first eight bits are the data that the register holds and is being read to the master, asserted by the camera. The ninth bit is asserted high by the master.

In the third phase of the 3-phase write transmission cycle, the first eight bits are the data being written to the register and are asserted by the master. The ninth bit is asserted the in the same way as in the first phase.

Whether the SIO-0 is sending input from the master to the clock or from the clock to the master is controlled by two internal lines, the SIO0_OE_M_ and the SIO_OE_S_. The SIO0_OE_M_ is inside the master. It can be thought of as being low when the master is writing and high when it is not. Since this is an internal line it can be the other way around as well depending on the design; however, it will still do the same job. The SIO_OE_S_ is inside the clock. It is low when the clock is writing to the master and high when it is not.

To better understand the SCCB bus we need to look at the timing. The minimum frequency of the serial clock, SIO-1, is 10 ∝s. When SIO-1 is low a byte of data is taken from the SIO-0. When SIO-1 is high the bit of data is transmitted. Protocol dictates that SIO-0 must be stable when SIO-1 is high. At the beginning and the end of the transmissions precautions must be made to insure that there is no propagation of an unknown bus state. The SIO-0 must be driven high for a period before and after the SBB goes from high to low at the start of a transmission. The SIO-1 must be high during this time. The minimum amount of time that SIO-0 is high before the SBB goes from high to low is 15 ns. The minimum period after SBB goes from high to low that SIO-0 must stay high before it may go low is 1.25 µs. The SIO-0 must also be driven high for a period before and after the SBB goes from low too high at the end of the transmission. Just as at the beginning of the transmission, the SIO-1 must be high during this time. The minimum amount of time that SIO-0 is high before the SBB goes from low to high is 15 ns. There is no minimum period after SBB goes from low too high that SIO-0 must stay high before it may go low. A 2-phase transmission cycle takes about 200 μs to transmit. A 3-phase transmission cycle takes about 290 ∝s to transmit.

4.8 Registers

The 125 registers control all the functions of the camera. Seventy of the registers are used exclusively for internal use. There are only a few of the other registers that are important to know.

One of the most important registers is register 13. When bit 0 is low it enables manual control, which allows one to manually change the values in registers 00 - 02. When bit 0 is high it enables auto adjust mode. When bit 1 is high it initiates single frame transfer, as described by the timing generator section. Bit 5 allows one to select 8 bit format when high and 16 bit format when low.

Register 12 contains bits that can manipulate register 00 - 02 in auto control. Bit 2 when low holds register 01 - 02 at the last updated value. When it is high it is controlled by internal functions. Bit 5 when low sets register 00 to default value. When it is high it is controlled by internal functions. Bit 7 when high resets all registers to their default value.

Registers 01 and 02 are blue gain control and red gain control respectively. The values of these registers may need to be changed if the colors have low intensities. Bit seven indicates the gain is increasing when high or decreasing when low. The other seven bits compute the gain by the following formula:

$$\text{Gain} = 1 + (\langle 6:0\rangle - 80)/100 \qquad (6)$$

<6:0> is the value in bits six through zero.

Register 10 controls the exposure timing of the clock. One must be in manual adjust mode in order to write values effectively. If the clock is not in manual adjust mode this register is updated by internal functions. Exposure timing is determined by the following formulas:

For interlace:
exposure time = line time x <7:0> (7)

For progressive:
exposure time = line time x <7:0> (8)

Line time = frame time/525 if clock is 27 MHz line time = 63.5 (9)

Register 14 bit 5 selects output format to be 320 x 240 when high and 640 x 480 when low.

Register 28 bit 5 selects progressive scan when high and interlace scan when low.

Register 61 bit 7 selects YUV output when high or raw data when low. YUV is the default value.

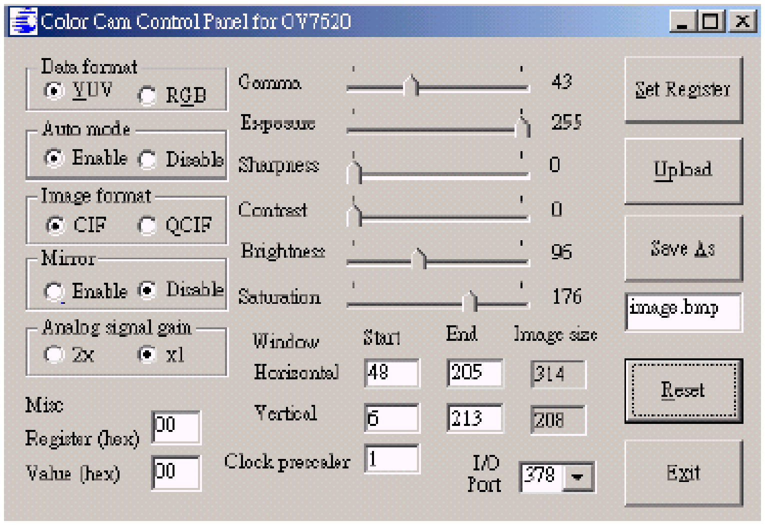
5. Interface

There are two ways to interface the camera with a computer that should be considered. The camera can interface with the computer through a parallel port or through a USB connection by going through other external hardware. A parallel port connection connects to a computer through a pin connection, where a USB connection connects through a USB connection. A parallel port connection is slower than a USB connection. A USB connection has a max rate of around 12 megabytes per second. One can use a buffer to hold data before it is transferred if one wants to take pictures at a faster rate. The EZUSB and 511+ interface with the computer through a USB connection where the 38EV/48EV interfaces through a parallel port connection.

6. Additional Hardware and Their Drivers

Drivers are used as a way to set the registers of the camera through software that can be used by a computer. There are three pieces of hardware that I have considered to run the camera. Each one has it's own driver that it uses to run the camera.

The first is the 38EV/48EV control board. The 38EV ($83.95) can be used with color camera modules. The 48EV ($78.95) cannot be used with color camera modules [1]. The 38EV/48EV has a parallel port interface and the software of the 38EV/48EV appears to be easy to use. The figure below is the control panel for the 38EV/48EV.



Data format: [✓ YUV] RGB
Auto mode: [✓ Enable] Disable
Image format: [✓ CIF] QCIF
Mirror: Enable [✓ Disable]
Analog signal gain: 2x [✓ x1]

Gamma	43
Exposure	255
Sharpness	0
Contrast	0
Brightness	96
Saturation	176

Window	Start	End	Image size
Horizontal	48	205	314
Vertical	6	213	208

Clock prescaler: 1
Misc
Register (hex): 00
Value (hex): 00
I/O Port: 378
image.bmp

Figure 6: Above is the control panel for the 38EV/48EV.

The second is the 511+ chip. When the 511+ chip is purchased with the camera module ($160.00) [2] it has a USB connection. The drivers that comes with the 511+ are difficult to use for the experiment. To set the frame rate using either one driver or the other is difficult. The 511+ also randomly drops frames and pads the frame rate with duplicate frame when it drops a frame while using one of these drivers. There are some Linux drivers that have been written for the 511+ [3], unfortunately, the registers of the 511+ are complex and not all of the code has been written.

The third is to use a EZUSB chip to connect the camera module to the computer through a USB connection. This would require software code to be written. The problem with the approach is the question of how to set up the transfer protocol to and from the SCCB. The EZUSB controls its own protocol but it is not known if they are the same type as the camera's protocols.

7. Data

Pictures were taken of the explosion, after the problems with synchronizing the camera were solved.

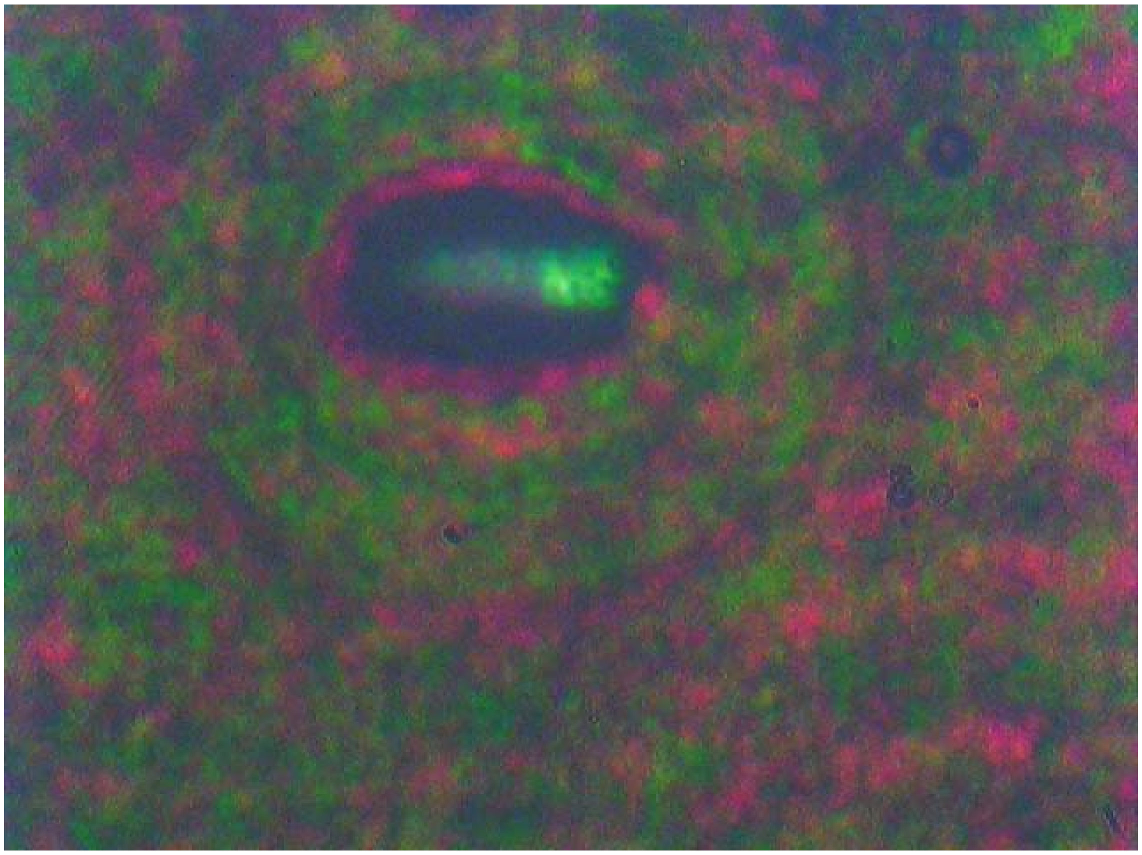


Figure 7: Above is a picture of an explosion taken by the camera.

After examining image array of the camera, figure 3, it was discovered that the pixels may be shifted. Figure 3 shows the Bayer Pattern which shows that the R (red),G (green), and B (blue) pixels are at different locations. Each pixel in the array is a single color but the format of a single pixel is determined by all three colors. The other two colors must come from the surrounding pixels. The unknown algorithm that combines the different colors into an individual pixel may shift the information of the individual color intensity that the pixel is filtered by the lens (as described in the image array section) to collect into one or more of the surrounding pixels.

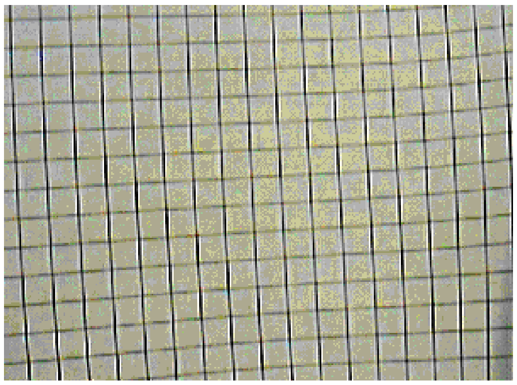


Figure 8 Above is the grid pattern that was imaged on the camera.

We devised an experiment to test if there is a pixel shift. The first step of the experiment was to create a grid pattern. The printed grid had a cell size of 0.19 inches X 0.19 inches, and the line width was almost one hundredth of an inch. The grid was placed approximately 20 cm away from the camera lens and imaged on the camera. The cell size when imaged onto the camera was approximately 20 X 20 pixels and the line width was one pixel. The intensities of the three colors were examined near the grid lines.

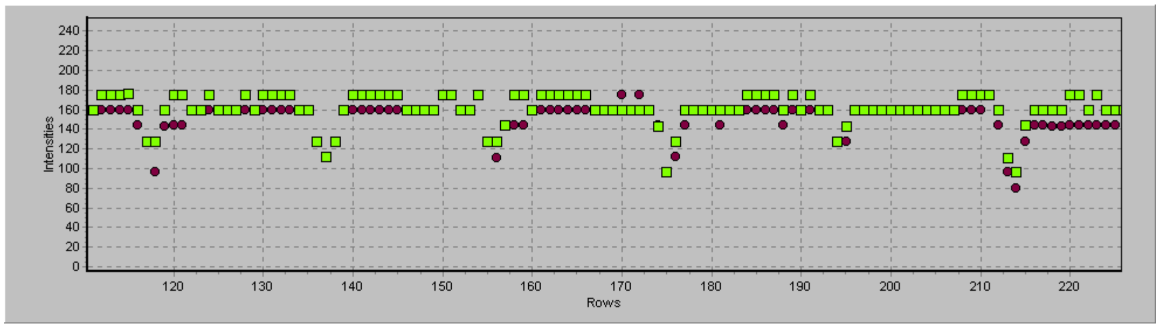


Figure 9: Above is graph of the intensities of the red and green colors down Column 130 in the bitmap of a picture taken of the grid. A black dividing line is at about every 20 rows, where the intensities drop.

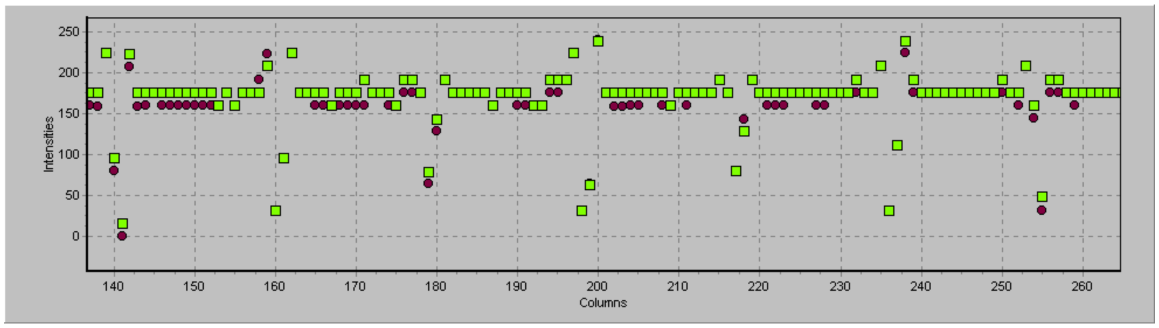


Figure 10: Above is a graph of the intensities of the red and green colors across row 108 in the bitmap of a picture taken of the grid. A black dividing line is at about every 20 columns, where the intensities drop.

The intensities of both colors in figure 8 and figure 9 decrease dramatically at about every 20 pixels were the black dividing line is. The width of this sudden decrease is only a couple pixels which indicates that there are no significant pixel shifts.

8. Conclusion

One can use an inexpensive CMOS camera as a piece of scientific equipment. One can synchronize the camera with the other instruments through the hardware so that the pixel clock of the camera runs the laser. The 511+ is an effective interface between the camera and the USB connection to a computer. One can use custom C++ programs to capture the images at known time intervals. One can also use another custom C++ programs to filter for one of the three colors, Red, Blue, or Green. We have proved that there are no significant pixel shifts of the data collected in individual pixels. A CMOS camera can be a less expensive alternative to a scientific camera.

Although the CMOS camera is synchronized with the equipment and takes pictures at known times, one can still improve the camera's performance. The camera currently takes pictures at one frame per second. However, the maximum frame rate of the camera is 60 frames per second.

One way one may attempt to improve the frame rate is to changing some of the software. The function call to video for windows that is currently being used in our custom C++ program (capGrabFrameNoStop(hWnd)) to capture a frame is the slowest of the three commands available (capGrabFrameNoStop(hWnd), capGrabFrame(hWnd), and capCaptureSingleFrame(hWnd)). When the fastest of the three function calls (capCaptureSingleFrame(hWnd)) was used with the camera only the frame rate was 5 frames per second. When the picture preview was disabled the frame rate was 15 frames per second with this new function call. One could try using this new function call instead of the old one while the camera is synchronized to the camera. One could also try to use a program similar to windows media player instead of capturing a bit map with a C++ program. This would allow software that is newer than video for windows to control the drivers.

Another approach is to try using an EZUSB instead of a 511+. The advantage gained by this approach is the elimination of all of the software inside the 511+. The camera and the EZUSB are well documented. The disadvantage to this approach is that one must write the programs that allow the EZUSB to control the camera. However, one still maintains a USB connection with this approach.

One may also try the 48EV. I have not tried using this yet. The major disadvantage with this approach is that the 48EV has a parallel port connection.

References

[1] Electronics 123, RADIO, TV AND VIDEO,
http://www.electronics123.com/amazon/catalogue/c3-1-5.htm,
November 2002

[2] AllAmerican Direct,
http://www.allamerican.com/direct/Results.asp?id=&WHAT=NUM&W
TYPE=S&
W1=&W2JOIN=OR&W2=&MFG=_OVT%2C&show=10&search+now.
x=64&se
arch+now.y=7, December 2002

[3] Mark McClelland, Linux OvCam Drivers,
http://alpha.dyndns.org/ov511/,
November 2002

[4] Omnivision technologies inc., advanced information preliminary
OV7620/OV7120, http://www.ovt.com/pdfs/ds_7620.pdf, December
2002

[5] Koch Brian, Undergraduate Physics Honors Thesis, College of
William and
Mary, 2003

Figure 2. OV7620 block diagram is from,
http://www.ovt.com/pdfs/ds_7620.pdf

Figure 5. The timing diagram for the first phase of a transmission
cycle is from,
http://www.ovt.com/pdfs/ds_note.pdf

Figure 6. Control panel for an OV7620 camera using a 38EV driver is
fromHttp://www.electronics123.com/amazon/datasheet/Cev38.pdf