Synchronized Strobe for Video Camera

Project By:
Jeff Baskett and Jason Zubo

Faculty Advisor:
Dr. James Irwin

May 12, 2001

Bradley University
College of Engineering and Technology
Department of Electrical and Computer Engineering

http://cegt201.bradley.edu/projects/proj2001/vidstrob

Abstract

The system controls a strobe that will be synchronized with the shutter on a video camera. Based on the shutter, the strobe will fire during each frame of video to freeze high-speed motion, preventing any blurring effects. Inputs into the system are the synchronized video signal from the camera and the intensity of the light read by the sensor. The output from the system is the intensity setting of the light to the strobe. The recorded image will be analyzed to determine the effectiveness of the system.

Table of Contents

Table of Figures

Introduction

Observing fast-moving objects using a video camera with a relatively slow shutter speed produces frames of video containing blurred objects. The Mechanical Engineering department at Bradley uses water tables to analyze fluids. Using a camera with a slow shutter speed does not allow them to obtain precise data for the velocity vectors of the fluids. By synchronizing a strobe light with the opening of the shutter, stop-motion effects will allow for accurate analysis of high-speed fluids

Functional Description

Objective of Research

The Mechanical Engineering (ME) department currently uses a video camera to analyze the motion of fluids in water table tests. Fluids moving at high speeds create blurred images on individual video frames. This greatly reduces the accuracy of measurements. The goal of this project is to eliminate these blurred images. The final synchronized system will create a stop-motion effect in every frame of video. With this stop-motion photography system, the ME department will be able to obtain more precise data of high speed motion in the analysis of their fluid systems.

Significance of Research

This project will allow for more detailed analysis of high-speed motion. Its applications are not limited to fluids research. This system could also be used in the study of kinetics. The motion of a runner, high-speed projectiles, and other numerous physics, athletic, and physical therapy applications could use this system to increase the accuracy of experimental measurements. While this research is not revolutionary, the systems applications are widespread and useful for many different disciplines.

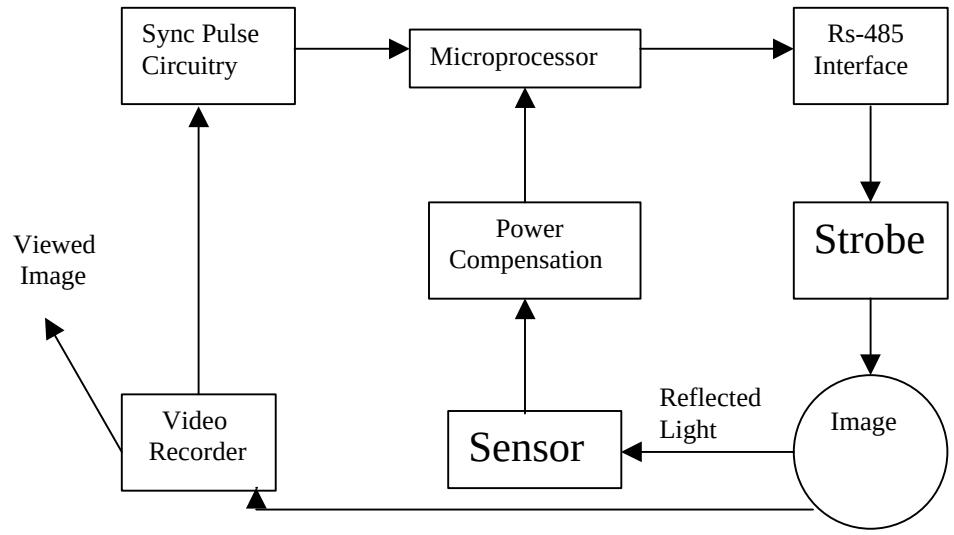
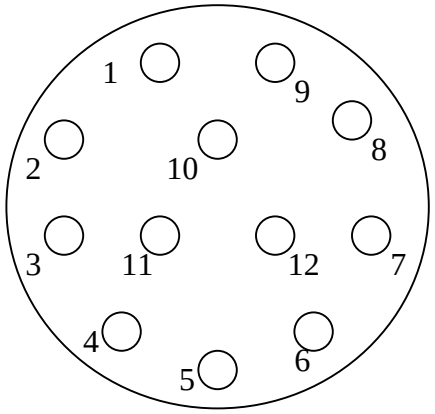


Figure 1 Block Diagram

Design

Video Recorder

The video recorder being used is the Sony XC-75 black and white module. The VD (vertical sync pulse) signal from the video camera contains the information telling when the shutter is going to be open. The VD signal from the camera was not functional, so the video signal, which has the VD signal embedded in it was manipulated. This video signal and its respective ground were obtained from an output from the 12-pin connector on the rear of the camera. We also used the 12-pin connector for supplying the necessary 12 volts dc to the camera. See Figure 2 for a diagram of the 12-pin connector. The video signal from the camera can be seen in Figure 3.



Pin Number	Function
1	Gnd
2	DC +12V
3	Video Output (Gnd)
4	Video Output
5	HD Output (Gnd)
6	HD Output
7	VD Output
8	Clock Output (Gnd)
9	Clock Output
10	Gnd
11	DC +12V
12	VD Output (Gnd)

Figure 2 12 Pin Connector

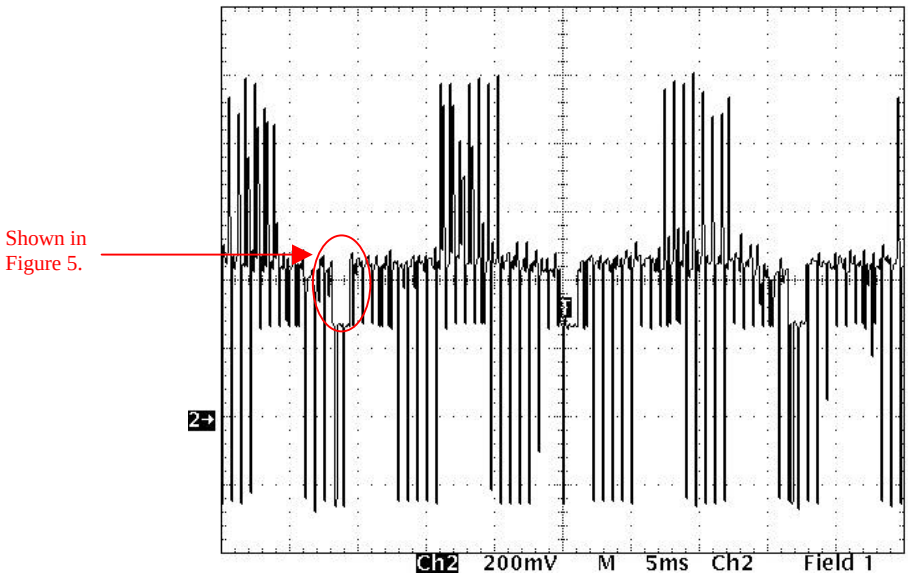


Figure 3 Video Signal

Sync Pulse Circuitry

The sync pulse circuitry will create the sync signal used trigger the firing of the strobe. First, the vertical sync pulse had to be obtained from the composite video signal. To obtain the sync pulse from the video signal seen above, we had to rectify and invert the signal with a precision rectifier and we used a schmitt trigger to make the pulse square. The schematic for the sync pulse circuitry can be seen in Figure 4. The final sync signal can be seen in Figure 5.

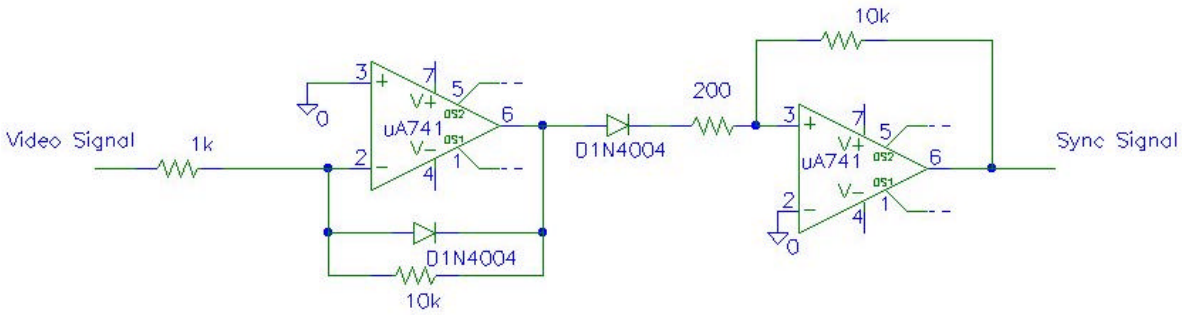


Figure 4 Sync Pulse Circuitry Schematic

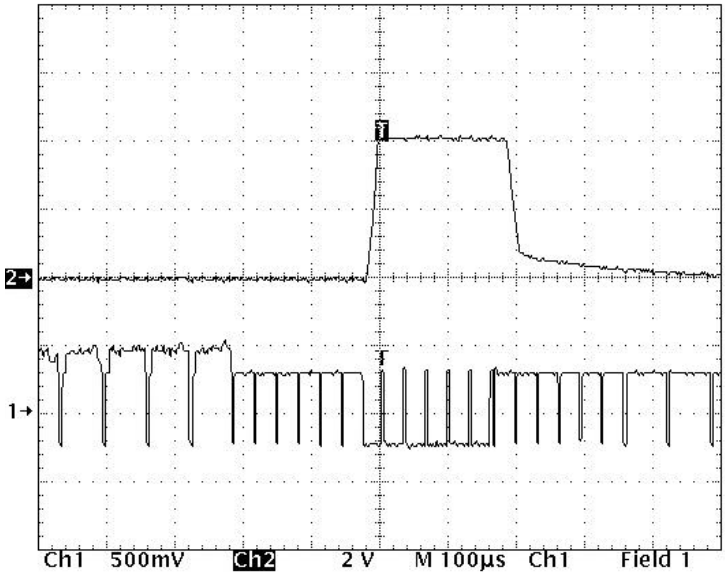


Figure 5 Sync signal

Strobe

Our design uses a single strobe to create a stop-motion effect for each frame of video. The strobe that was used is the Snapshot DMX/D. It was used because it allowed for control of rate, intensity, and duration. It was also capable of rates up to 60 flashes per second, which is twice as fast as the video camera takes pictures.

Sensor and Power Compensation

The sensor circuit utilized an EG&G Vactec PhotoDiode with a current to voltage converter. The sensor circuit gathers the reflected light from the image. The schematic for the sensor circuitry can be seen in Figure 6. Theoretically, the output of the sensor circuit is fed to the power compensation circuitry, which consists of a window detector. The sensor circuit was mainly used for determining when the strobe was firing. A low pass filter with a cutoff of 200 Hz was added to the sensor circuit to produce a smooth curve. The window detector has two outputs, increment and decrement. These outputs are inputs to the microprocessor and tell the software when to increment or decrement the intensity of the strobe. Although the window detector was built and tested, it was never used with the system. The output pulse from the sensor circuitry can be seen in Figure 7. The schematic for the window detector can be seen in Figure 8.

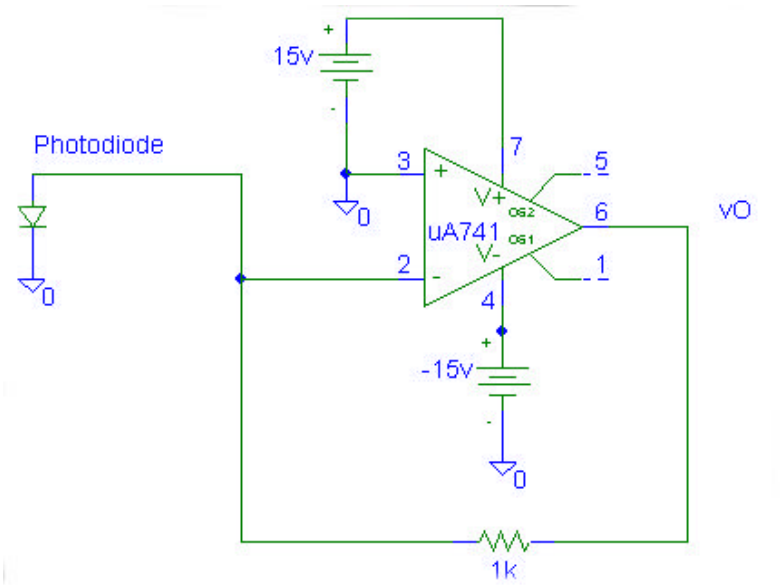


Figure 6 Schematic of Sensor Circuitry

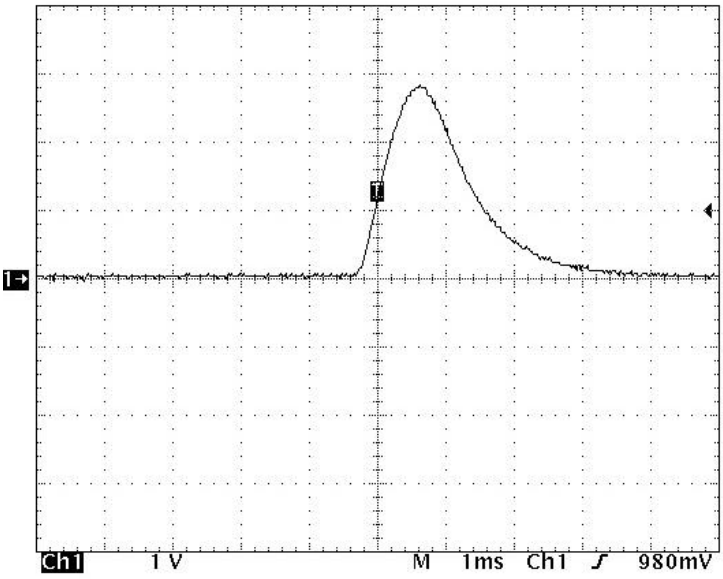


Figure 7 Output from Sensor Circuitry

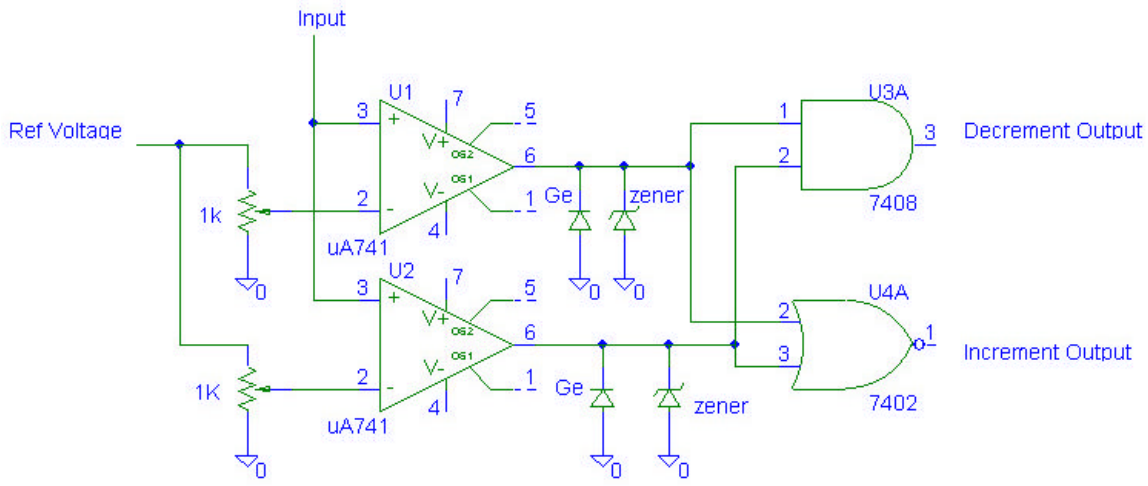


Figure 8 Schematic for Window Detector

Microprocessor and DMX interface

The microprocessor is responsible for producing an appropriate DMX signal when the sync signal triggers a software interrupt. The software has a preset value for the intensity of the light. This value was found through experimentation with various intensity levels to find one that produced the best results. Originally this value would be incremented or decremented based on the output from the power compensation circuitry, but we never implemented this function in the final design. A flowchart of the software is shown in Figure 9.

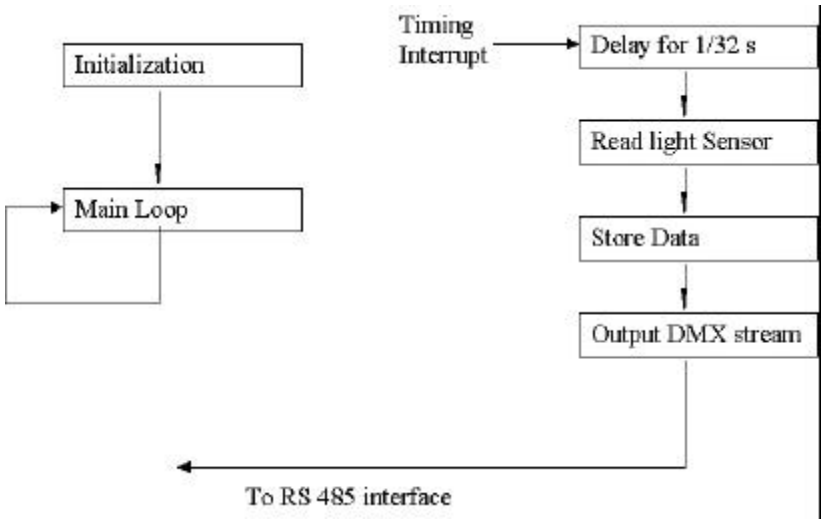


Figure 9 Software Flow Chart

The software simply waits for an interrupt and outputs the appropriate signal to the RS-485 interface. The main loop simply initializes the microprocessor and data then waits in an infinite loop. The timing interrupt triggers the DMX output. When the timing interrupt is triggered there is a delay added to output data at the next frame of picture (1/30 s). The software would then read the light sensor data, change the intensity value based on the sensor, and output the DMX signal. In the final design the delay and sensor feedback are removed due to inaccuracies with the strobe light. The signal is sent out using the external UART on the EMAC board. The DMX signal that is sent consists of a 22 bit break, a two bit mark after break, an 11 bit start code (low), one start bit, 8 data bits, and two stop bits. In TTL logic a break is high and a mark after break is low, while in RS-485 a break is low and a mark after break is high. This data must be sent at 250 kbaud. The signal from the microprocessor is shown in Figure 10. Channel 1 is the output from the microprocessor and channel 2 is diode rectified and inverted for the RS-485 interface. The DMX standard uses low breaks and high mark after breaks. The RS-485 interface did not invert the signal so a TTL inverter was used.

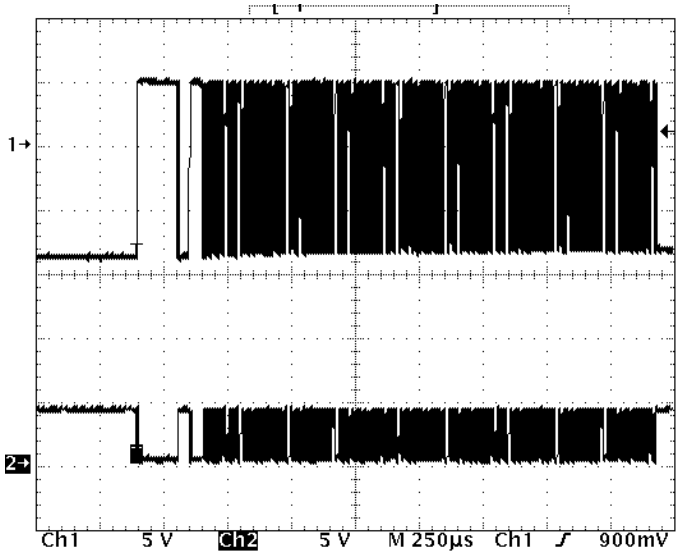


Figure 10 Microprocessor output

The output shown actually contains 25 packets of data. This was due to a problem with the strobe ignoring packets of data due to addressing problems.

RS-485 Interface

The RS-485 driver used was the MAX1480C. The DMX standard calls for differential inputs of an RS-485 signal. The TTL level signal from the microprocessor

was converted to a differential signal with this RS-485 driver. We chose the MAX1480C because of its ability to achieve a bit rate of 250 kbaud and for its full isolation. Because of the abundant noise inherent with firing a strobe light we needed full isolation to keep voltage spikes from reaching the EMAC board.

Trouble Shooting and Modifications

The first major problem encountered was in learning how to use the UART on the microprocessor. The initialization of the UART is a very involved process that was difficult to troubleshoot. Since the UART would not operate until initialized properly, it was extremely difficult to find problems. The major difficulty was that different library files were needed to operate the UART, after this problem was solved the UART operated properly.

Another problem we encountered was the need to invert the DMX signal from the microprocessor. We were able to fire the strobe without the inversion, but the strobe fired at full speed(60Hz) and we were unable to synchronize it with the DMX signal. Dr. Schertz found the problem and after inversion the strobe operated more consistently.

There were also problems with the number of data packets that were sent to the strobe light. The strobe light has a certain DMX address that determines which data packets it will read. This is how DMX-512 can control 512 lights with one signal. If we did not send a certain number of packets the strobe would ignore the data, and not flash. We also found that the more packets of data we sent, the more stable the strobe fired. We experimented with various amounts of packets until we were satisfied using 25 data packets, each containing identical intensity information.

Grounding our electronics and the strobe was a small problem. We originally had all grounds tied together to earth ground. This put the EMAC board in a precarious position, possibly enabling voltage spikes to reach the board through ground. After separating the electronics from earth ground and tying the power supply commons to the EMAC ground we eliminated noise and the possibility for damaging the board. A block schematic is seen in Figure 11.

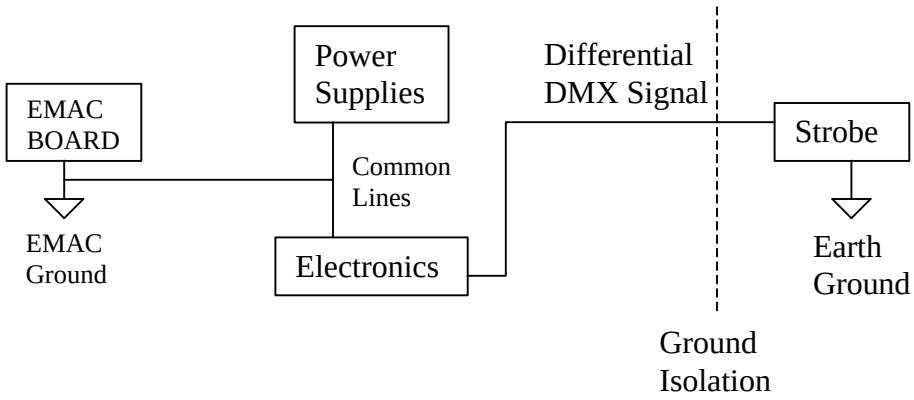


Figure 11 Ground Block Schematic

An apparently insurmountable obstacle was encountered when we discovered the inconsistency of the delay between when the strobe received the data and when it fired. Based on the specifications for the camera, we knew that we had a 4 ms window in which to fire the strobe. However, when we closely examined when the strobe was firing after it received the data, we found that the delay before the strobe flash varied from 1 ms to 10 ms thus making it impossible for us to know when the strobe was going to fire. The firing inconsistency would also prevent the synchronization of the strobe and the shutter. The variance in when the strobe fires is seen in Figures 12, 13, 14. Channel 1 is the DMX data stream and channel 2 is the output from the sensor circuit. After writing off our project altogether, we decided to test the system. The strobe's lack of precision proved to be a minor problem. Since the strobe was still flashing once for every frame of video we were still able to see stop-motion effects.

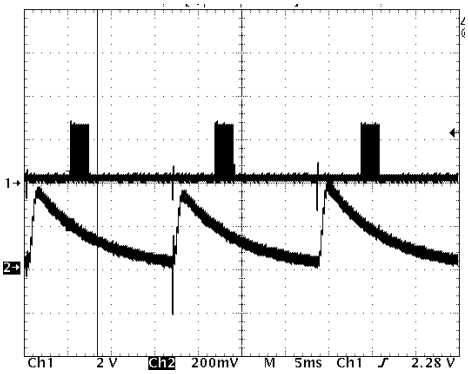


Figure 11 10 ms delay

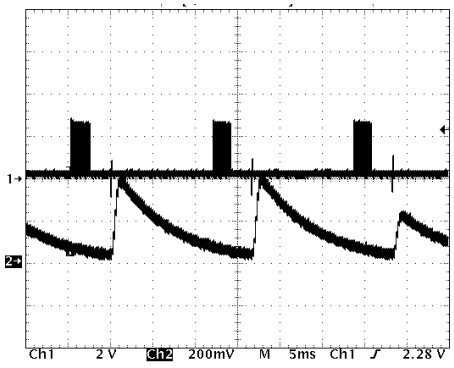


Figure 12 2.5 ms delay

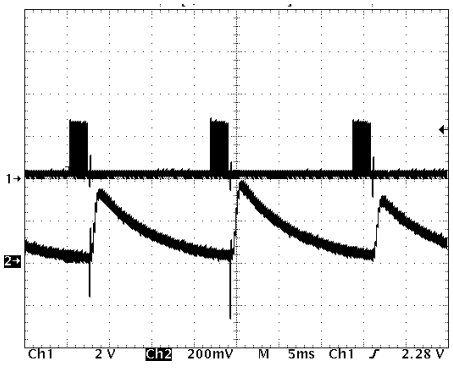


Figure 13 < 1 ms delay

The intensity of the strobe proved to be another inconsistency. The output from the sensor circuitry showed that the strobe's intensity fluctuated in a periodic manner. Figures 11, 12, and 13 also show the strobe intensity as being inconsistent. Channel 2 is the output from the sensor circuit. It is clearly seen Figure 13 that the three peaks on the sensor circuit output signal are all of different magnitudes. This rendered the power compensation circuitry useless. Buttons manually controlled the increment and decrement interrupts.

After setting up our expo display and attempting to run the system, we found that noise was slowly building on the sync signal causing the microprocessor to output data sporadically. In a moment of pure genius a low pass filter with a cutoff of 1 KHz was added to the sync signal. Problem: solved.

Results

The Synchronized Video Strobe was successfully demonstrated at the student expo. We performed several experiments in which we recorded high-speed motion with the synchronized strobe in operation and with room light. Experiments included a drop of water, swinging a golf club, spinning a wheel, breaking a light bulb, spinning a football, and spinning a frisbee. By viewing these experiments frame by frame, stop motion effects were clearly evident using the synchronized video strobe. These images and a short movie of the experiments can be viewed on the project web-site under week 16 accomplishments (http://cegt201.bradley.edu/projects/proj2001/vidstrob/). In the experiments with room lighting, images that were clear and precise with the strobe were blurry. The system would be very beneficial for obtaining precise data in an experiment involving high-speed motion.

Although the system proved useless to the ME department, a Caterpillar engineer has expressed interest in using the system for analyzing AC generators.

Parts List

- Strobe – American DJ Snap Shot DMX/D
- Video Camera - Sony XC-75
- RS-485 Interface - MAX1480C
- Sensor - EG&G Vactec PhotoDiode
- EMAC Evaluation Board

Schedule of Tasks

At the beginning of the semester we set up the dependency chart seen in Figure 15. We were fairly consistent with our schedule and we completed the project in time for the expo.

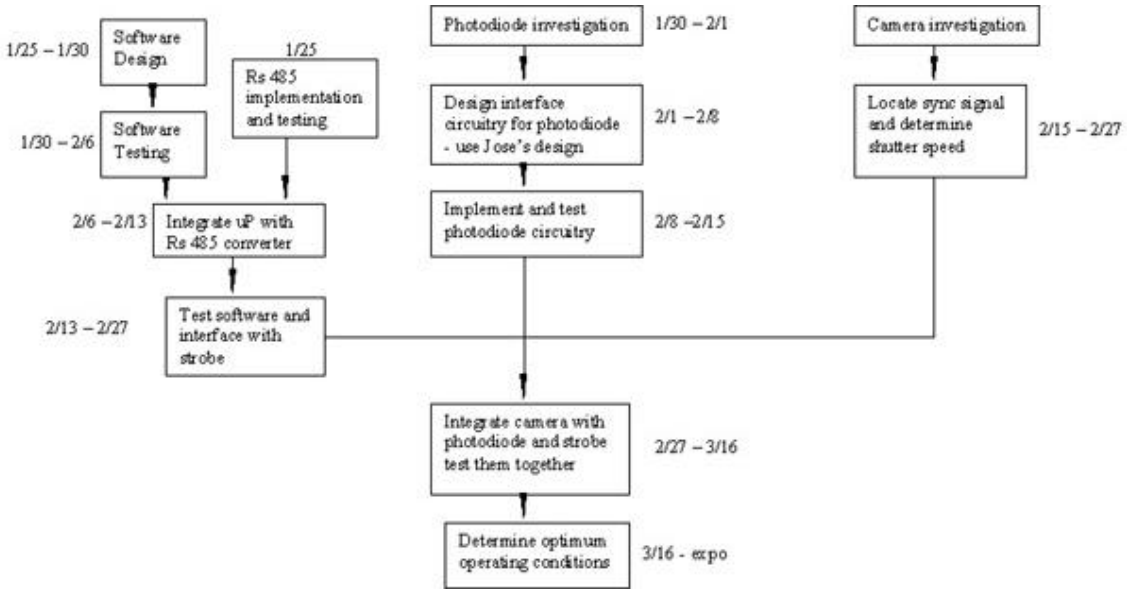


Figure 15 Dependency Chart

Appendix A. EMAC programs

```
       Final Program
;*************************************
;Synchronized Video Strobe interrupt driven DMX controller
;Senior Project
;Jason Zubo and Jeff Baskett
;Advisor: Dr. Irwin
;*************************************

;*************************************
;
;
;              Initialization Code
;*************************************
;$NOMOD51             ; disable predefined 8051 registers
;$INCLUDE(reg515.inc)
$include(mod515.inc)
START   equ            8000h

;*************************************
;Jump table for interrupts

       org     start + 4Bh           ;EX2 interrupt (timing)
       ljmp    timing

    org start + 53H     ;EX3 interrupt (increment)

    ljmp       increment

    org start + 5Bh           ;EX4 interrupt (decrement)
    ljmp       decrement

       org     START + 100h
       jmp            setup
       ;UART initialization
MR0ADAT  EQU 00000001B ;extend baudrate 1
MR1ADAT  EQU 00010011B ;no RTS, Rx int on RxRDY, char mode, no parity, 8 data
MR2ADAT  EQU 00001111B ;normal, no TxRTS, no CTS, 2 stop bit
MR1BDAT  EQU 00010011B ;no RTS, Tx int on RxRDY, char mode, no parity, 8 data
MR2BDAT  EQU 00001111B ;normal, no TxRTS, no CTS, 2 stop bit

MR0A     EQU 00H
MR1A     EQU 00H  ;Mode register (MR1A,MR2A) (rd/wr)
CSRA     EQU 01H  ;Clock select register A (wr)
SRA      EQU 01H  ;Status Register A
CRA      EQU 02H  ;Command Register A (wr)
THRA     EQU 03H  ;Tx holding register
ACR      EQU 04H  ;Auxiliary Control Register (wr)
MR1B     EQU 08H  ;Mode Register B
CSRB     EQU 09H  ;Clock Register B
SRB      EQU 09H  ;Status Register B
CRB      EQU 0AH  ;Command Register B
THRB     EQU 0BH  ;Tx holding Register
UARTIN   EQU 0dh
IPCR     EQU 04H
setup:
       mov     IEN0,#90h     ;IENO (A8h) is interrupt enable 0.
                                    ;Bit7 = 1 = global interrupt enabled.
                                    ;Bit4 = 1 = 80C535 serial COM0
                                    ;interrupt enabled.

       mov     IEN1,#00h     ;Disable all individual interrupts.
                                    ;IEN1 (B8h) is interrupt enable 1.

       mov     SP,#2Fh       ;Initialize stack pointer.
;
;Initializations specific to
;the 80C535

    setb       p5.5                        ;reset
       clr            p5.5
       setb    P5.0           ;Make bit A16 of 128K Ram "high".
       clr            P5.2                ;Disable EEPROM.
;
;Note port P5.1 = 1 is required to write to D/A converter
```

```
;since the D/A converter is included in the MMIO (Memory Mapped I/O).
;
       clr     P5.1          ;Enable MMIO (memory mapped IO).
       setb    EAL                  ;Enable interrupts
       setb    EX2                  ;Enable timing interrupt(#2)
       setb    EX3                  ;Enable increment interrupt
       setb    EX4           ;enable decrement interrupt
       setb    I2FR          ;set interrupt 2 (timing) to positive edge triggered
       setb    I3FR          ;set increment interrupt to positive edge trigger

       MOV  R0,#7FH  ; clear 128 bytes of RAM

CLR_RAM:

        MOV  @R0,#0
        DJNZ R0,clr_ram

    mov        a,      #01010000B            ;reset ports
CRINIT:



        MOV  P2,#CRA
        MOVX @R1,A
        MOV  P2,#CRB
        MOV  @R1,A
        ADD  A,#-16
        JNZ  CRINIT       ;Subtract 1 from uper nibble until loop is zero

        mov  a, #10110000B ;set MR to zero
        mov  P2, #CRA
        movx @R1, a

        mov  P2,#MR0A
        mov  a,#00000001B
        movx @R1,a

        MOV  P2,#MR1A     ;Setup protocol for PORT A
        MOV  A,#MR1ADAT
        MOVX @R1,A
        MOV  A,#MR2ADAT
        MOVX @R1,A


        MOV  P2,#MR1B     ;Setup protocol for PORT B
        MOV  A,#MR1BDAT
        MOVX @R1,A
        MOV  A,#MR2BDAT
        MOVX @R1,A

        MOV  P2,#ACR      ;select baud rate
        MOV  A,#00H
        MOVX @R1,A        ;select set 1 of baud rates
        MOV  P2,#CSRA
        MOV  A,#11001100B
        MOVX @R1,A        ;Rx and Tx at 9600 for A
        MOV  P2,#CSRB
        MOVX @R1,A        ;Rx and Tx at 9600 for B
        MOV  P2,#CRA
        MOV  A,#00000101B ;Enable Txer and Rxer
        MOV  @R1,A
        MOV  P2,#CRB
        MOVX @R1,A        ;same for B
        mov           8400h,  #0011101000111010B    ;intensity initialization

;
;End 80C535 memory and I/O initialization.
;

 main :

       cpl p4.7
       sjmp main
```

```
timing:
    clr EAL           ;disable all interrupts
    cpl p4.1
    nop
    jb  p4.1,  exit

        mov  a, #01100000B    ;start break
    mov  P2, #CRB
    movx @R1, a

               mov            R5,            #0001H
        wait2:
               mov            R6,            #0047h
        wait1:
               djnz   R6,            wait1
               djnz   R5,            wait2          ;88us wait

        mov  a, #01110000B ;stop break (mark after break)
    mov  P2, #CRB
    movx @R1, a

               mov            R6,            #000Fh
        wait3:
               djnz   R6,            wait3

        mov  a, #01100000B ;start code
    mov  P2, #CRB
    movx @R1, a

               mov            R5,            #0001H
        wait6:
               mov            R6,            #015h
        wait5:
               djnz   R6,            wait5
               djnz   R5,            wait6

        mov  a, #01110000B ;stop break (mark after break)
    mov  P2, #CRB
    movx @R1, a

    mov        a,             8400h
    mov    R6,      #025h

SEROUTB:
        mov            P2,            #SRB
        push   acc                           ;save char

intensity_out:

    movx       a,             @R1
    jnb        acc.2,  intensity_out  ;loop till ready
    pop        acc
        mov    P2,            #THRB                 ;output intensity
        movx   @R1,   a
        nop
    djnz       R6,            seroutB

exit:
    setb EAL
reti

increment:
        clr EAL        ;disable all interrupts
        mov            a,                    8400h
        inc            a
        mov            8400h,         a
        cpl    p4.6
        setb EAL
reti
decrement:
        clr EAL        ;disable all interrupts
        mov            a,                    8400h
        dec            a
        mov            8400h,         a
        cpl    p4.5
```

```
        setb EAL
reti

end

        Jose's sample UART code
;*************************************************
;Jose Sanchez and Matthew Rickert
;serial.a51 - Test program for EMAC using Serial Port
;Last Updated - March 30, 2000
;*************************************************


$INCLUDE(mod515.a51)

STARD    EQU 8000H  ; start address for program
         ORG stard
         JMP SETUP

MR1ADAT  EQU 00010011B ;no RTS, Rx int on RxRDY, char mode, no parity, 8 data
MR2ADAT  EQU 00000111B ;normal, no TxRTS, no CTS, 1 stop bit
MR1BDAT  EQU 00010011B ;no RTS, Tx int on RxRDY, char mode, no parity, 8 data
MR2BDAT  EQU 00000111B ;normal, no TxRTS, no CTS, 1 stop bit
;
MR1A     EQU 00H  ;Mode register (MR1A,MR2A) (rd/wr)
CSRA     EQU 01H  ;Clock select register A (wr)
SRA      EQU 01H  ;Status Register A
CRA      EQU 02H  ;Command Register A (wr)
THRA     EQU 03H  ;Tx holding register
ACR      EQU 04H  ;Auxiliary Control Register (wr)
MR1B     EQU 08H  ;Mode Register B
CSRB     EQU 09H  ;Clock Register B
SRB      EQU 09H  ;Status Register B
CRB      EQU 0AH  ;Command Register B
THRB     EQU 0BH  ;Tx holding Register

SETUP:

         MOV  IEN0,#0  ; Disable all interrupts
         MOV  SP,#70H  ; Initialize STACK

                       ; * 80535 initialization requirements
         SETB P5.5     ; do a reset
         CLR  P5.5     ; bring it low
         SETB P5.0     ; make A16 of 128K Ram, high
         CLR  P5.2     ; disable EEPROM
         clr  P5.1     ; enable memory mapped IO
                       ; end 80535 stuff

         MOV  R0,#7FH  ; clear 128 bytes of RAM

CLR_RAM:

         MOV  @R0,#0
         DJNZ R0,clr_ram

INIT2681:

         MOV  A,#01010000B ;Do Reset command for ports A and B.

CRINIT:

         MOV  P2,#CRA
         MOVX @R1,A
         MOV  P2,#CRB
         MOV  @R1,A
         ADD  A,#-16
         JNZ  CRINIT       ;Subtract 1 from uper nibble until loop is zero

         MOV  P2,#MR1A     ;Setup protocol for PORT A
         MOV  A,#MR1ADAT
         MOVX @R1,A
         MOV  A,#MR2ADAT
         MOVX @R1,A

         MOV  P2,#MR1B     ;Setup protocol for PORT B
```

```
         MOV  A,#MR1BDAT
         MOVX @R1,A
         MOV  A,#MR2BDAT
         MOVX @R1,A

         MOV  P2,#ACR      ;select baud rate
         MOV  A,#80H
         MOVX @R1,A        ;select set 2 of baud rates
         MOV  P2,#CSRA
         MOV  A,#10111011B
         MOVX @R1,A        ;Rx and Tx at 9600 for A
         MOV  P2,#CSRB
         MOVX @R1,A        ;Rx and Tx at 9600 for B
         MOV  P2,#CRA
         MOV  A,#00000101B ;Enable Txer and Rxer
         MOV  @R1,A
         MOV  P2,#CRB
         MOVX @R1,A        ;same for B


LOOP:

         MOV  A,#33h

SEROUTB:

         MOV  P2,#SRB
         PUSH ACC          ; SAVE CHAR

SOUTB1:

         MOVX A,@R1
         JNB  ACC.2,SOUTB1 ; LOOP TILL TXrdy
         POP  ACC
         MOV  P2,#THRB     ; SEND IT OUT
         MOVX @R1,A

END
```

Mod515.inc

```
; 80515 MOD FILE
; REV. 1.1 Feb 24, 2000

$SAVE
$NOLIST

IEN0 DATA 0A8H ;INTERRUPT ENABLE REGISTER 0
IP0 DATA 0A9H ;INTERRUPT PRIORITY REGISTER 0
IEN1 DATA 0B8H ;INTERRUPT ENABLE REGISTER 1
IP1 DATA 0B9H ;INTERRUPT PRIORITY REGISTER 1
IRCON DATA 0C0H ;INTERRUPT REQUEST CONTROL
CCEN DATA 0C1H ;COMPARE/CAPTURE ENABLE
CCL1 DATA 0C2H ;COMPARE/CAPTURE REGISTER 1 - LOW BYTE
CCH1 DATA 0C3H ;COMPARE/CAPTURE REGISTER 1 - HIGH BYTE
CCL2 DATA 0C4H ;COMPARE/CAPTURE REGISTER 2 - LOW BYTE
CCH2 DATA 0C5H ;COMPARE/CAPTURE REGISTER 2 - HIGH BYTE
CCL3 DATA 0C6H ;COMPARE/CAPTURE REGISTER 3 - LOW BYTE
CCH3 DATA 0C7H ;COMPARE/CAPTURE REGISTER 3 - HIGH BYTE
T2CON DATA 0C8H ;TIMER 2 CONTROL
CRCL DATA 0CAH ;COMPARE/RELOAD/CAPTURE - LOW BYTE
CRCH DATA 0CBH ;COMPARE/RELOAD/CAPTURE - HIGH BYTE
TL2 DATA 0CCH ;TIMER 2 - LOW BYTE
TH2 DATA 0CDH ;TIMER 2 - HIGH BYTE
ADCON DATA 0D8H ;A/D CONVERTER CONTROL
ADDAT DATA 0D9H ;A/D CONVERTER DATA
DAPR DATA 0DAH ;D/A CONVERTER PROGRAM REGISTER
P4 DATA 0E8H ;PORT 4
P5 DATA 0F8H ;PORT 5
INT3 BIT 090H ;P1.0 - EXTERNAL INTERRUPT 3/CAPTURE 0/COMPARE 0
INT4 BIT 091H ;P1.1 - EXTERNAL INTERRUPT 4/CAPTURE 1/COMPARE 1
INT5 BIT 092H ;P1.2 - EXTERNAL INTERRUPT 5/CAPTURE 2/COMPARE 2
```

```
INT6 BIT 093H ;P1.3 - EXTERNAL INTERRUPT 6/CAPTURE 3/COMPARE 3
INT2 BIT 094H ;P1.4 - EXTERNAL INTERRUPT 2
T2EX BIT 095H ;P1.5 - TIMER 2 EXTERNAL RELOAD TRIGGER INPUT
CLKOUT BIT 096H ;P1.6 - SYSTEM CLOCK OUTPUT
T2 BIT 097H ;P1.7 - TIMER 2 INPUT
ET2 BIT 0ADH ;IEN0.5 - TIMER 2 INTERRUPT ENABLE
WDT BIT 0AEH ;IEN0.6 - WATCHDOG TIMER RESET
EAL BIT 0AFH ;IEN0.7 - GLOBAL INTERRUPT ENABLE
EADC BIT 0B8H ;IEN1.0 - A/D CONVERTER INTERRUPT ENABLE
EX2 BIT 0B9H ;IEN1.1 - EXTERNAL INTERRUPT 2 ENABLE
EX3 BIT 0BAH ;IEN1.2 - EXTERNAL INTERRUPT 3/CAPTURE/COMPARE INTERRUPT 0 ENABLE
EX4 BIT 0BBH ;IEN1.3 - EXTERNAL INTERRUPT 4/CAPTURE/COMPARE INTERRUPT 1 ENABLE
EX5 BIT 0BCH ;IEN1.4 - EXTERNAL INTERRUPT 5/CAPTURE/COMPARE INTERRUPT 2 ENABLE
EX6 BIT 0BDH ;IEN1.5 - EXTERNAL INTERRUPT 6/CAPTURE/COMPARE INTERRUPT 3 ENABLE
SWDT BIT 0BEH ;IEN1.6 - WATCHDOG TIMER START
EXEN2 BIT 0BFH ;IEN1.7 - TIMER 2 EXTERNAL RELOAD INTERRUPT ENABLE
IADC BIT 0C0H ;IRCON.0 - A/D CONVERTER INTERRUPT REQUEST
IEX2 BIT 0C1H ;IRCON.1 - EXTERNAL INTERRUPT 2 EDGE FLAG
IEX3 BIT 0C2H ;IRCON.2 - EXTERNAL INTERRUPT 3 EDGE FLAG
IEX4 BIT 0C3H ;IRCON.3 - EXTERNAL INTERRUPT 4 EDGE FLAG
IEX5 BIT 0C4H ;IRCON.4 - EXTERNAL INTERRUPT 5 EDGE FLAG
IEX6 BIT 0C5H ;IRCON.5 - EXTERNAL INTERRUPT 6 EDGE FLAG
TF2 BIT 0C6H ;IRCON.6 - TIMER 2 OVERFLOW FLAG
EXF2 BIT 0C7H ;IRCON.7 - TIMER 2 EXTERNAL RELOAD FLAG
T2I0 BIT 0C8H ;T2CON.0 - TIMER 2 INPUT SELECT BIT 0
T2I1 BIT 0C9H ;T2CON.1 - TIMER 2 INPUT SELECT BIT 1
T2CM BIT 0CAH ;T2CON.2 - COMPARE MODE
T2R0 BIT 0CBH ;T2CON.3 - TIMER 2 RELOAD MODE SELECT BIT 0
T2R1 BIT 0CCH ;T2CON.4 - TIMER 2 RELOAD MODE SELECT BIT 1
I2FR BIT 0CDH ;T2CON.5 - EXTERNAL INTERRUPT 2 FALLING/RISING EDGE FLAG
I3FR BIT 0CEH ;T2CON.6 - EXTERNAL INTERRUPT 3 FALLING/RISING EDGE FLAG
T2PS BIT 0CFH ;T2CON.7 - PRESCALER SELECT BIT
F1 BIT 0D1H ;PSW.1 - FLAG 1
MX0 BIT 0D8H ;ADCON.0 - ANALOG INPUT CHANNEL SELECT BIT 0
MX1 BIT 0D9H ;ADCON.1 - ANALOG INPUT CHANNEL SELECT BIT 1
MX2 BIT 0DAH ;ADCON.2 - ANALOG INPUT CHANNEL SELECT BIT 2
ADM BIT 0DBH ;ADCON.3 - A/D CONVERSION MODE
BSY BIT 0DCH ;ADCON.4 - BUSY FLAG
CLK BIT 0DEH ;ADCON.6 - SYSTEM CLOCK ENABLE
BD BIT 0DFH ;ADCON.7 - BAUD RATE ENABLE
$RESTORE
```