Project Documentation

Project Title: CRT Display using mega16

Team Members: Mohit Agarwal, Aman Jain, Mridul Verma

Team Mentor: Anurag Sai

Basic aim:

To implement a display unit for a microcontroller capable of displaying color images. For this we used a CRT display and interfaced it with MCU. To test this we used external memory to display animation patterns and designed a real time game on CRT Monitor screen.

How do I get the Idea…??

Generally we start thinking our project Idea when we are forced to discuss it with Coordinators, I also followed the same rule but this idea came from my thoughts when I was a child. At that time we have a TV screen and a personal desktop. So, I preferred to see movies on big TV screen rather than on small desktop. So, I thought how to connect my CPU to TV screen, but due to lack of knowledge I was unaware about the VGA to AV convertor cable. This time when we were supposed to think project Ideas, I reminded that and I thought of making a box who will convert VGA Protocol to NTSC/PAL Protocol, But when I discussed my idea with seniors, less feasibility was there. So, my project Idea keeps on modulating and finally I ended up with controlling CRT Displays through a single MCU.

Theory:

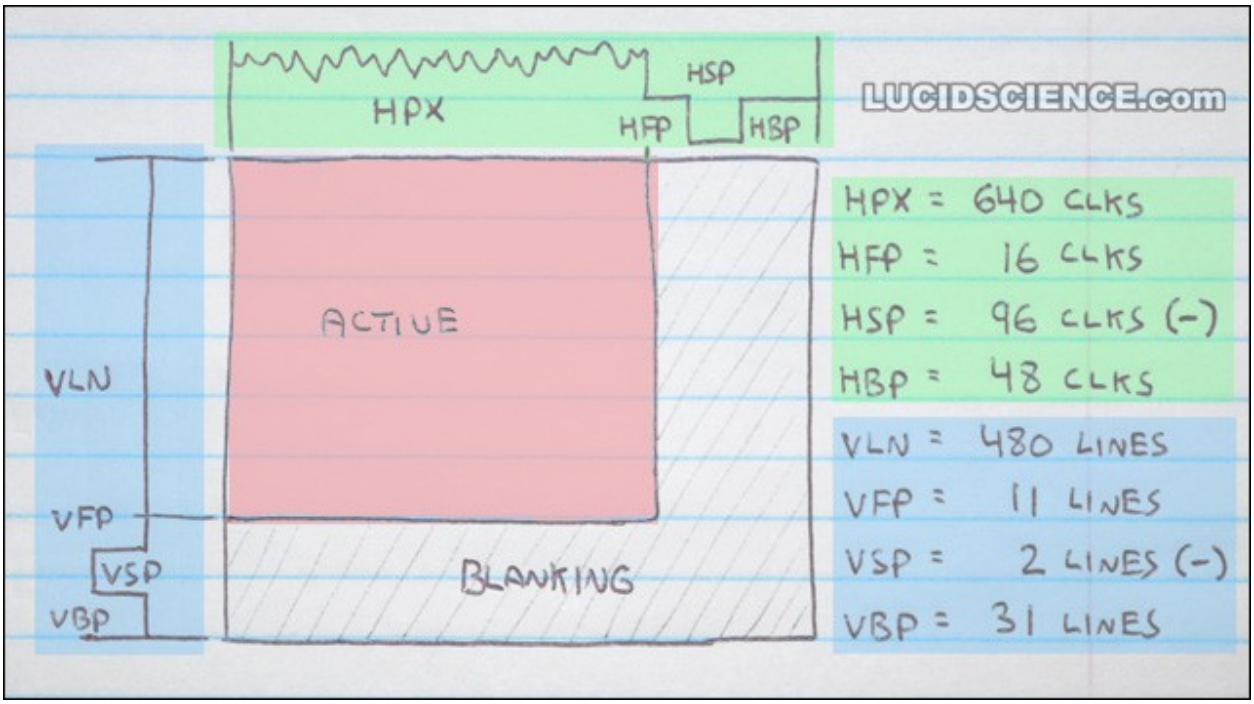
VGA Protocol-

First we will discuss about how a monitor display. Everyone has seen the VGA connector pin in monitor similar to below one which is usually remain connected to CPU's VGA out port.

This cable includes a total of 15 pins divided in set of three, each set includes 5 pins. First three i.e. pin no. 1,2 and 3 are of Red, Green, Blue respectively. Then Pin no.5,6,7,8,10 are Ground pins.Pin no. 13 and 14 represents H_Sync and V_Sync respectively. H_Sync and V_Sync signal pins are digital whereas RGB pins are analog in nature.Voltage between 0v to 0.7v decides the value of R,G,B (correspondly divides into from value of 0 to 255)

Horizontal and Vertical Sync. Signals significance and their timings:

When a CRT(Cathode Ray tube) starts painting the screen it starts from the very first pixel and move to pixels in same line. Then before starting second line a Horizontal Sync. Signal is needed to tell the monitor to move to next line. Like this after completing all 480 lines a Vertical Sync. Signal is required to tell the monitor to start new frame.H_Sync and V_Sync signals works same as ENTER and NEW command works in MS-Word.

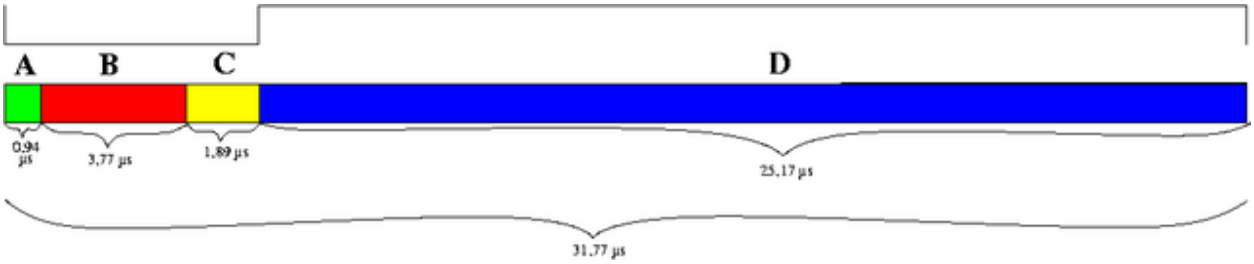


In the above Image, Timings of H_Sync and V_Sync signals are shown.

HPX refers to the pixel data, means pixels colour values are sent through RGB pins and H_Sync and V_Sync pins are kept high during this time.It requires 25.17us, i.e. 640 clocks while working at 25.175MHz.For 16MHz corresponding clocks can be calculated that comes out to be approx. 400 clocks.HFP refers to Horizontal Front Porch signal, this is a part of H_Sync. Signal.It requires 0.94us.During this time No Image Data should be sent and H_Sync pin should be kept high.HSP i.e. Horizontal Sync. Pulse requires 3.77us.H_Sync pin should be kept low.After HSP,follows HFP(Horizontal Front Porch) that demands 1.89us and at this time H_Sync pin should be kept low.

Similar to these H_Sync. Signals V_Sync signals are there, but they are counted in terms of line.Described in the Image above VLN(Vertical Lines)are 480,VFP(Vertical Front Porch) 11 lines, VSP(Vertical Sync. Pulse) 2 lines and VBP(Vertical Black Porch) 31 lines.Out of all these only at the time of VSP, V_Sync pin should be kept low and except that it will remain high.Image Data should be send

only during VLN.



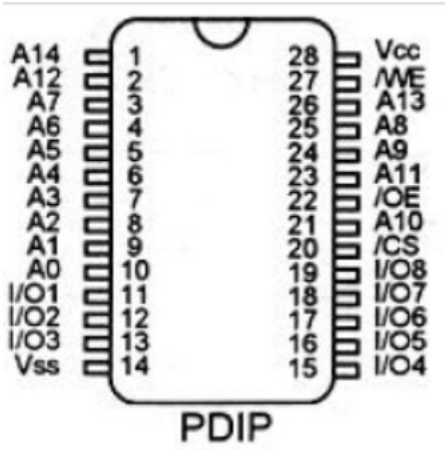
Hardware configuration:

MCU used is AtMega16 clocked to 16MHz.Parallel SRAM is also attached with mega16 to store whole frame data.Along with that a 74245 buffer ic is interfaced to decide when to send Image Data.

Clocking of ATMEGA using external Crystal Oscillator:

Simply attach the 16MHz Crystal Oscillator to the 12th and 13th pin of mega16 i.e. XTAL1 and XTAL2.If CPU frequency is defined change it to the frequency of external crystal i.e. 16MHz and finally while programming the MCU through AVR Studio go into fuse setting choose EXT Crystalwith maximum delay time and Program it.

Interfacing SRAM:



Pins A0 to A14, total of 15 pins are Address pins.At One address one byte of data can be stored.So, Capacity of SRAM can be calculated easily i.e. 2^15 bytes(32 kb).VSS pin and VCC pin are Ground and Live pins respectively.I/O 1 to I/O 8, a total of 8 pins are data pins. WE is Write enable, OE is output enable and CS is Chip Select, I think their names are enough to describe their work, and important thing

is that these pins do their work when they are in low state, ex. When CS pin is high, chip is deselected.Its connections with MCU is in Image attached below and Coding scheme of SRAM can be found .

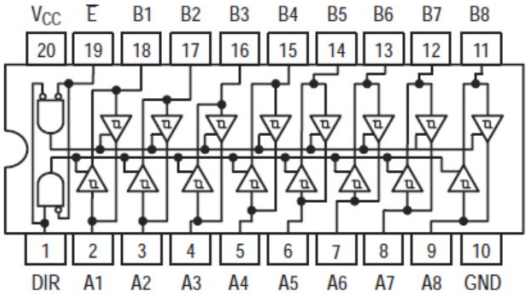
Interfacing and use of 74245 buffer ic:

Its work is just to transfer data from left port to right one or from right port to left one.Pin diagram is shown below. Truth Table pic is also attached, Just toggling one pin we can easily transfer One Port data to another one.As in our module, we need to transfer Image data only when no Sync Signals are going otherwise monitor fails to display anything.

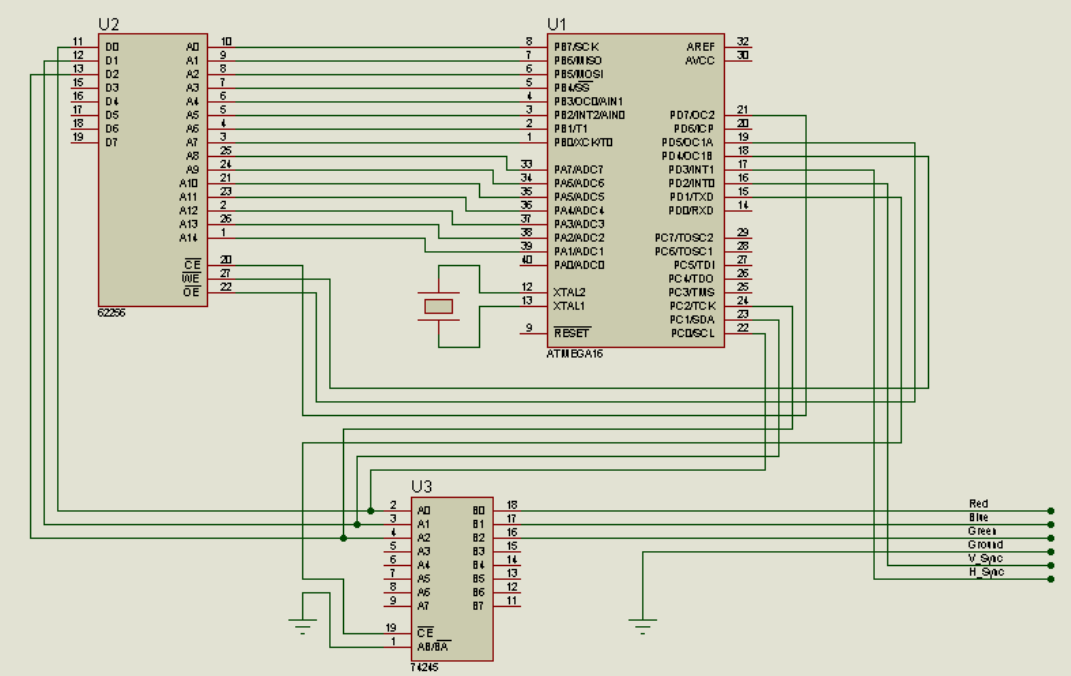
TRUTH TABLE

INPUTS		OUTPUT
$\overline{E}$	DIR	
L	L	Bus B Data to Bus A
L	H	Bus A Data to Bus B
H	X	Isolation

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial



Schematic is shown here:



6 pins are described. A 6 pin connector can be used to interface it with VGA connector.

Software Configuration:

Generating Sync. Signals through MCU:

Timer0(Normal Timer) is used fro generating Sync. Signals.Timer frequency is 2MHz and each time timer initializes from 195 and count upto 255 with 2MHz, at 255 overflow interrupt is called and all Sync. Signal code is described there. First 45 lines are V_Sync lines from which line no. 10 and 11 corresponds to VSP, at that time V_Sync pin is kept low and H_Sync signals are produced using NOP commands i.e. No Operation Performed.NOP command just kills off the 1 instruction cycle, it is basically assembly language command but can be used in C using asm("nop").Any assembly language command can be used in C using asm("") command and writing assembly language command inside inverted commas.

S_RAM coding:

Initializing SRAM- Code is written in "sram.h" and "sram.c" under function SRAM_Init(). In this function, WE,CS,OE turned high and all Addressial ports are defined output.

Writing on SRAM- Importants points of writing cycle

1). Set data port data direction to output (since we're writing data)
2). Make sure the chip is deselected (CE high)
3). Put the address on the address bus
4). Set WE and OE control lines to high before-hand

5). Select the chip (CE low)
6). Put the data on the data bus
7). Set WE low
8). This is when the chip inputs the data
9). Set WE high
10). End the write cycle by deselecting the chip

Code is described under function SRAM_Write()

Reading through SRAM- Important points of Reading Cycle

1). Set data port data direction to input (since we're reading data)
2). Make sure the chip is deselected (CE high)
3). Put the address on the address bus
4). Set WE and OE to high before-hand
5). Select the chip (CE low)
6). Set OE low
7). This is when the chip outputs the data
8). Read from the data bus
9). Set OE high
10). End the write cycle by deselecting the chip

In SRAM_init() function sleep function is enabled and MCU is set in sleep mode in the starting line of while(1) loop.MCU wakes up from sleep mode when interrupt is called and after producing Sync. Signals pointer returns back and starts producing Image Data signal. If SRAM_Read cycle is performed as it is described it will consume a lot of time which obviously reduces our resolution. When I tried my code practically it was giving a resolution of 50 in horizontal, so I kept 240 resolution in vertical otherwise image will look like horizontally stretched. I have used a bit programming and technical skill and set up all to minimize time of Reading cycle. I have stored every pixel data in SRAM which is of 8 bytes out of them I have used first three bits only. Information is organized like every pixel has two coordinates x and y i.e. x is no. of pixel starting from yth line and y refers to the line number. As SRAM have 15 pins two ports(ADDRL and ADDRH) of ATMEGA will be required to drive signals in SRAM.8 pins of ADDRL ports are used and 7 pins of ADDRH ports are used. So, ADDRH port can define a maximum of 256 value while ADDRL can define up to 128. So, I defined ADDRH port as line number and ADDRL port as pixel number. Now from start, I set up ADDRL port as in each loop line number is constant, shifted it to right by one bit to convert 480 lines to 240 lines. After this every time I define ADDRH port and do OE high and low. One NOP is used between Turning OE high and low because minimum time required to read two successive data values from SRAM requires 60 to 70 nanoseconds delay.

Moving to Animation:

Till now, we have learnt how to display static data stored in SRAM on CRT screen. Now for display animating data we use to update data in SRAM for which we do not have any time left. So I started up with various ideas to assemble some time to update SRAM but everything was failed. Then finally I thought of skipping alternating frames i.e. displaying data in one frame and keeping whole screen black in another frame.This will not look awkward as frame rate is much high 60 frmes per second only a slight

flickering appears on screen. Code for the same is written first in interrupt code where framecount variable is counted and even no. of frames are skipped and odd no. are displayed.

Introduction to VGA Library:

Here I am releasing version 1.0 of VGA library designed by myself.
Now using above decribed code, I have designed VGA Library not exactly similar to the GLCD/LCD Library we have but very similar to that. Its coding is done in "vgalib.c".Uptill now, I have included only Rectangle function. Basic funda behind VGA Library is that just if we want to display Rectangle on Screen we require its diagonal coordinates and colour, using that write on SRAM pixel values i.e. colour only for those whose pixel value should be changed.Like this more functions can be defined in VGA Library ex. Circle,Line,Ovals,Text etc. and this library can also be made rich like GLCD have one.

Using VGA Library:

Under the scope which is commented "Write your Global variables here" is meant for writing initial code i.e. for very first frame and under the scope which is commented "Write your Local variables here" code for the next frames can be written. Also, it should be kept in mind that Global variables can be defined only in Global variable section and that can also be accessed under local variable section.

Demo Game Designing using VGA Library:

For designing PING PONG game, I have completely utilized VGA library and Images. Moving ball is defined as rectangle whose coordinates are changed in every frame (done by removing previous rectangle and designing new one).Moving bar is also defined as rectangle and its horizontal coordinates are changed according to button pressed. Finally normal gaming programming is done scoring according to comparing coordinates of ball and bar.Score and PING PONG label are stored in lash memory by designing in Paint and converting it to image data using OpenCV application.

Further developments and Future Scope:

1). Slideshow displayer using SD card.
2). High Resolution Display using fast SRAM and high frequency Atmega.
3). Video Player from SD Card/USB interface.
4). MS-Paint designing using PS2 interface.
5). Zapper gun can also be interfaced to make games more interactive.
6). Can be made to display on LCD/TFT monitors or with projectors.

Utility:

This can be a good replacement of expensive video hardware. Can be used for products advertisements at any trade fair or for banners, for giving a attractive look without spending a lot of money on costly video drivers. This module costs only about 300Rs. More video games can be designed and of good professional quality.

Useful Links:

SRAM interface: http://dev.frozeneskimo.com/embedded_projects/parallel_sram
VGA: http://www.serasidis.gr/circuits/AVR_VGA/avr_vga.htm
http://www.lucidscience.com/pro-vga%20video%20generator-1.aspx

A word of thanks

I would specially like to thank my team mentor Anurag Sai, Club Coordinators RAJAT ARORA, RISHABH MAHESHWARI and GANESH to contribute required help and guide given by them and making summers full of chill and learning experience.