Final Project Report

Introduction

The chosen final project is to utilise MSP432 Launchpad to display colour blocks on a flat-panel display using VGA protocol. The colour data to display are stored in the main flash memory of the MSP432 Launchpad. The microcontroller will fetch the corresponding colour in memory and output the colour on screen accordingly. By pressing the right button, the launchpad fetches the next colour in the memory and display it on the screen. By pressing the left button, the launchpad changes between displaying 1 colour block or displaying 2 colour blocks on the screen, one colour from the current memory location and another from the following memory location. The colour LED onboard will indicate the expected colour from at the current memory location.

Block Diagram

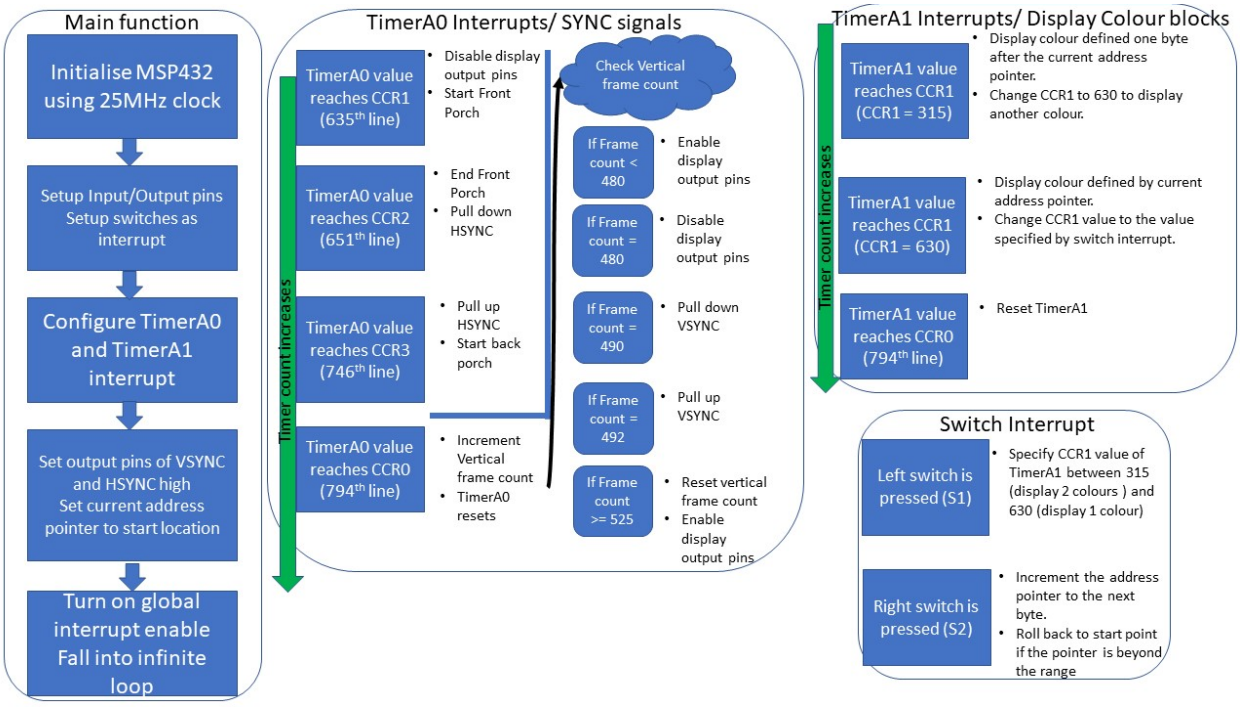

Image 1: Block Diagram

Description of Solution

Program

The MSP432 Launchpad has a 25MHz oscillator (MODOSC).[1] In this project, this oscillator is sourced as main clock in the system. Since 25MHz is close to the pixel frequency of VGA protocol at 640x480@60Hz, which is 25.175 MHz, 640x480@60Hz is the resolution mode selected in this project.[4] However, to compensate the slightly slower clock, the actual horizontal line count is 635 instead of 640. The new count values are calculated using formula $new\ line\ value = t_{scanline\ part} \times 25MHz$.[2] The time each

scanline part must spent is defined in the industry standard. After the calculation, one can find that 635th line ends the visible area and starts front porch; 651th line ends front porch and sets the horizontal sync signal to low; 746th line sets the horizontal signal back to high and starts back porch; 794th line ends back porch, swings back to left, and increment the frame count.

The first step is to produce both the horizontal sync signal and vertical sync signal. For the horizontal sync signal, TimerA0 is used to manage operations at specific frame time. TimerA0 uses SMCLK as source, making the count of TimerA0 change at 25MHz. Since display output needs to be off when outside the visible area, CCR1 is set to 635 to interrupt the MSP432 and disable the output pin. CCR2 and CCR3 are set to 651 and 746, respectively, to interrupt and toggle the horizontal sync pin. As 794th line is the last line before starting from the left edge, CCR0 is set to 794 to interrupt, flag that the horizontal sync is ready to display, and perform checks on vertical sync line. P2.7 is the output for horizontal sync signal. [2]

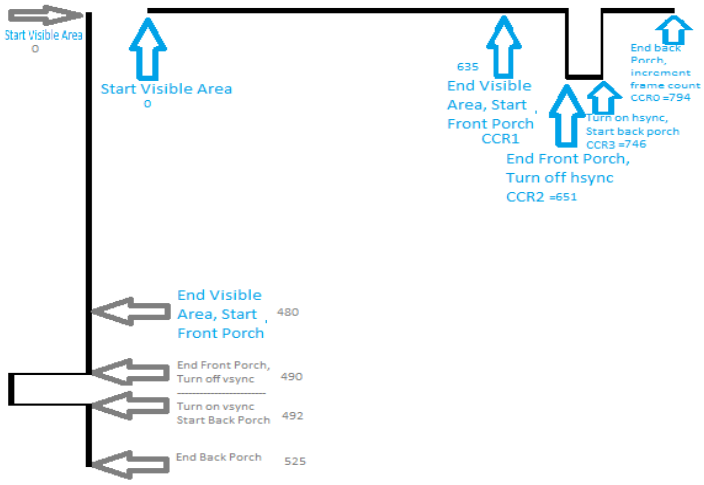

Image 2: VGA Timing [2]

For the vertical sync signal, if the frame count is less than 480, it is in the visible area and the vertical sync is ready to display. If the frame count is 480, vertical sync is not ready and display output will be disabled. The vertical sync blinks between 490th frame and 492th frame. Eventually, once the vertical frame count reaches 525th frame, the vertical count will be zeroed and the display is ready to draw from the upper left corner of the screen again. P2.6 is the output for vertical sync signal. [2]

The next step is to represent colour information in 8 bits, and fetch them to the output pins when drawing the display. By defining 2 bits to control each basic colour: red, green, and blue, we can identify 4 different intensities for each basic colour. Overall, the total amount of bits used are the lower 6 bits and a total of 64 colours can be produced by matching different combination of colours using different intensities. (XXRRGGBB) [3] Colour information is prepared in the .bin file from location 0x1CC0 to 0x1D00. To output those colours, the value stored in the memory location is written to the output port directly. In this project, Port 4 is used to output colour. The 2 blue pins in the order of increasing resistance are P4.0 and P4.1. The 2 green pins and the 2 red pins in the same order are P4.2, P4.3, P4.4 and P4.5, respectively.

When there is one colour on the screen, CCR1 interrupt set in TimerA1 will only trigger once at the end of visible area. When 2 colours are on the screen simultaneously, CCR1 interrupt will be triggered when the horizontal line count is half way in the visible area. Afterwards, CCR1 will be summed with itself and produce another interrupt at the end of the visible area to have the next colour ready. Since the microcontroller is unable to process the data fast enough, the change of colour is visible on the flat screen monitor.

The last step is to configure the buttons used as interrupt. The left button (P1.1) switches between 1 colour block and 2 colour blocks on screen. It is accomplished by changing the CCR1 value of TimerA1 when a press is detected through interrupt. The right button (P1.4) brings the next colour stored in memory to the display. It is accomplished by incrementing the current address pointer to the next byte. If the current address pointer reaches the pixel end location, the pointer will roll back to pixel start location.

With the functioning code, it is compiled and flash into the MSP432 right away with Code Composer Studio. Nevertheless, the colours to display are not yet defined in the specific range in MSP432's memory. To flash the colour in memory, UniFlash tool is used to save the information on flash in a .bin file. Then HxD is used to modify contents at location 0x1CC0 to 0x1D00. The .bin file is then flashed back on MSP432.

Wiring

To produce colours in different intensities, each basic colour is controlled by 2 pins. One pin connects with higher resistance and the other connects with the lower resistance. When only the higher resistance pin is turned on, the colour has the least intensity. When only the lower resistance pin is turned on, the colour has medium intensity. When both pins are turned on, the higher resistance network and the lower resistance network form a set of parallel resistors, giving out the lowest resistance and the strongest intensity. Since all colour pins has 75 Ω resistor built into the monitor with a 0.7V peak-to-peak limit, and that MSP432 pins output 3.3V, the higher resistance network has a resistance of 1000Ω; the lower resistance network has a resistance of 460Ω; When 2 networks are in parallel, the equivalence resistance is 315.07Ω, so the intensities increases by roughly 1/3 each time. [3]

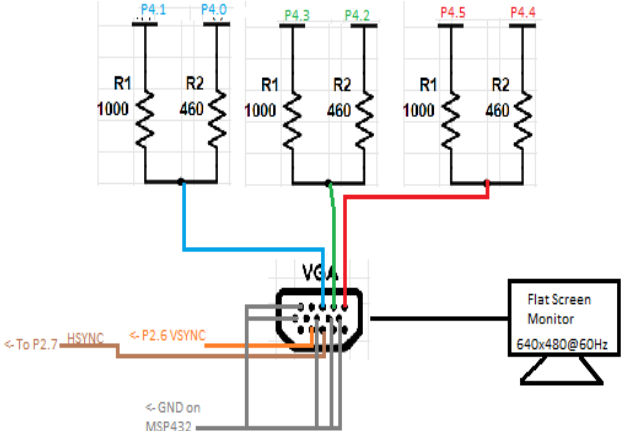

Image 3: VGA Schematic [3] [5]

With the regulated RGB signals, along with the horizontal and vertical signals produced earlier, we can pass them into a VGA pinout. After connecting all VGA ground pins (Pins No. 5, 6, 7, 10) together to the GND pin on the MSP432 launchpad, we can wire the red signal to Pin1, green signal to Pin2, and blue signal to Pin3. For the sync signals, horizontal sync signal goes to Pin13 and vertical sync signal goes to Pin14.[5]

Problems, Learning and Experience

One of the greatest problems in the project is that MSP432 cannot process fast enough to output pixels on time. Originally, a more complex picture with more pixels were considered to be the output. Since there is too little time between finishing processing the output of previous pixel to interrupting and fetching the next pixel, if a more complex image is loaded, nothing can be seen on the screen.

Another problem faced in the project is the difficulty in debugging. Since VGA protocol requires lots of work related to timing, it is hard to debug at home without an

oscilloscope. In the earlier phase, programming the horizontal sync and vertical sync signal took days of trial and error without an oscilloscope.

Through this project, I have more understanding in how a compute image is produced using an old-school method, and how people were able to produce a full screen of pixels without controlling each individual LEDs like what we use today. I have also gained more practise in using interrupts, timers, and the built-in flash memory.

Possible Extension and Improvement

The main threshold for this project is that MSP432 cannot output images stored in its flash memory to the output pins on time. If the image or pixel is stored in an external device, such as a flash or ROM, it is possible to display an entire image on the screen because the MSP432 no longer needs to wait for the pixel to arrive before being interrupt again. The MSP432 can now just sends the flash or ROM the address it needs and leave the output to the flash or ROM. [3]

Another way to improve and extend the project is to use FPGAs instead of MSP432. Since FPGAs can process instructions faster than MSP432, FPGAs should be able to handle the frequent interrupt along with the frequent memory access. Due to the enhanced performance in FPGA, it is even possible to extend the project and show animated images on the screen.

Work Cited

[1] Texas Instruments Technical Staff, *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*, Texas Instruments, 2019.

[2] *The world's worst video card?*, Jul. 05, 2019. Accessed on: Nov. 20, 2020. [Video file]. Available: https://www.youtube.com/watch?v=l7rce6IQDWs.

[3] *World's worst video card? The exciting conclusion*, Jul. 13, 2019. Accessed on: Nov. 20, 2020. [Video file]. Available: https://www.youtube.com/watch?v=uqY3FMuMuRo.

[4] VGA Signal 640 x 480 @ 60 Hz Industry standard timing, *SECONS Ltd*, 2008. [Online]. Available: http://tinyvga.com/vga-timing/640x480@60Hz. [Accessed: Nov. 20, 2020].

[5] VGA Connector, *Components 101*, 2018. [Online]. Available: https://components101.com/connectors/vga-connector-pinout-datasheet. [Accessed: Nov. 20, 2020].