Project 2: Video Playback

Billy Neuson and Greg Lepley

EGR 424

Dr. Parikh

6/24/2015

Custom Functions

void UARTIntHandler(void)

The function UARTIntHandler is in charge of responding to an interrupt generated by the UART peripheral when receiving data from the host computer. The interrupt handler performs a couple of important tasks. First, UARTIntHandler gets that status of the UART interrupt. Then, it clears the UART interrupt so that the peripheral is able to interrupt again. The UARTIntHandler then stores the received character from UART to a custom buffer in memory called g_queue. When the buffer is full, indicating a full page has been received, a global flag called pageReceived is set true so that the main function knows to begin sending the data to the OLED display.

void SSIIntHandler(void)

The purpose of the SSIIntHandler function is used to update the status of the SSI transmitter whenever a transmit interrupt occurs. This interrupt is setup to fire whenever the TX FIFO is half empty or less. This interrupt is activated every time data is written by the RITWriteCommand and RITWriteData functions in the OLED driver. The interrupt is also enabled every time a SSIBusy is called. Every time the interrupt is enabled, the peripheral waits until the trigger condition is set, and then runs the code in the interrupt service routine. This routine simply sets a global flag with the value of the busy bit found in the SSI Status register. If the transmitter is found to be busy, bytes will not be added to the FIFO by higher level drivers, and the data/command mode will not be switched.

int main(void)

The first thing that the function does is set up the clock and the peripherals so that they can be used properly. Main sets up the clock to run directly from the crystal and enables processor interrupts. Additionally, main enables the functionality and interrupt capabilities of the SSI0 and UART0 peripherals. It sets up GPIO pins A0 and A1 as UART pins. UART is configured for a baud rate of 1,500,000 with 8-N-1 operation. The OLED display is initialized as well. After all initialization and peripheral setup is complete, the function sits in a while loop and waits for a full page of display data to be received. Once the global flag pageReceived is true, main sends the page to the OLED display through SSI0.

Python script, imageSend.py

The python script used to send video frames over the UART is designed to keep the heavy lifting on the side of the connected PC. This simplifies the work that the microcontroller has to do, and instead of converting image formats in the device, a simple protocol is used that is compatible with the RIT128x96x4ImageDraw function. The process of image conversion is as shown in Figure 1.

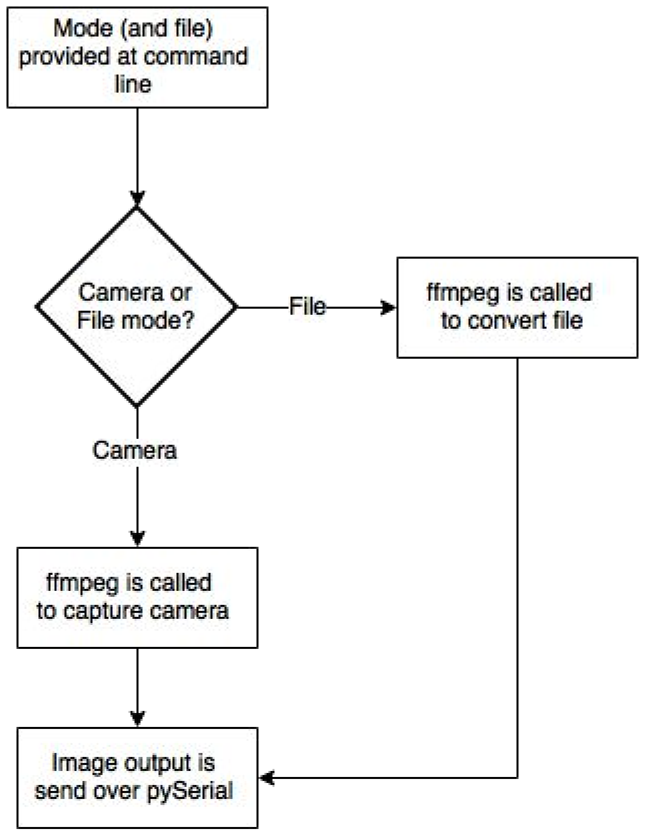
Mode (and file) provided at command line
Camera or File mode?
File
ffmpeg is called to convert file
Camera
ffmpeg is called to capture camera
Image output is send over pySerial

Figure 1: Python Script Flowgraph

At the first step, either ‘f’ or ‘c’ is provided in the command line execution. If a file is specified, the script calls on a video conversion program called ffmpeg to process the given file into PGM images. PGM images are perfect for this purpose due to the fact that they are 8 bit grayscale to begin with. The process of formatting the image for the device involved packing the most significant 4 bits of each pixel in order to accomplish a 8 bits per 2 pixel sequence to stream over the UART port. The generated image sequence is numbered such that python is able to iterate over all generated files before restarting the iteration. Each time the first frame is displayed, another program called ffplay attempts to play the input file to add sound from the PC speakers to the video on the OLED screen. Input is fault-tolerant and prevents the user from entering into situations where the program must be manually killed outside of a normal exit.

Extra Credit

For extra credit, we increased the frames per second of the video player. The project requirements state that the player needed to have at least a frame rate of 2 frames per second. Using a python script to convert video into image data and increasing the baud rate from 115,200 to 1,500,000 allowed us to increase our frame rate to between 10-20 frames per second depending on the performance of the connected computer. At this high of a UART speed, the time preparing a frame to be sent over the UART becomes a significant load, so this causes the variation in framerates from device to device.