**NOTES:**

* LEDs count to 7 bits

1. push ax  
2. push ds  
3. push dx  
4. mov dx, $379  
5. in al, dx  
6. mov ax, seg counter  
7. mov ds, ax  
8. inc counter  
9. mov al, EOI  
10. out PIC, al  
11. pop dx  
12. pop ds  
13. pop ax  
14. iret

P1Q1. Comment the provided assembly code shown above from the ISR of LAB2A.pas. Explain the necessity/role of each line in the overall functionality of the ISR. Use your own words and do not include the comments already provided in the .pas file \*

Lines 1-3 : Save the current values contained in register ax, ds and dx this is important because it allows the previous being performed operation to continue after ISR has been completed .\

Line 4: Moving address 379 to dx register

Line 5: Reading in the value of contained in dx which is 379$ in to al register

Line 6: Moving the address of the segment counter to the ax register

Line 7: moving ax to data segment counter

Line 8: Incrementing the value of the counter which allows for the LEDs to

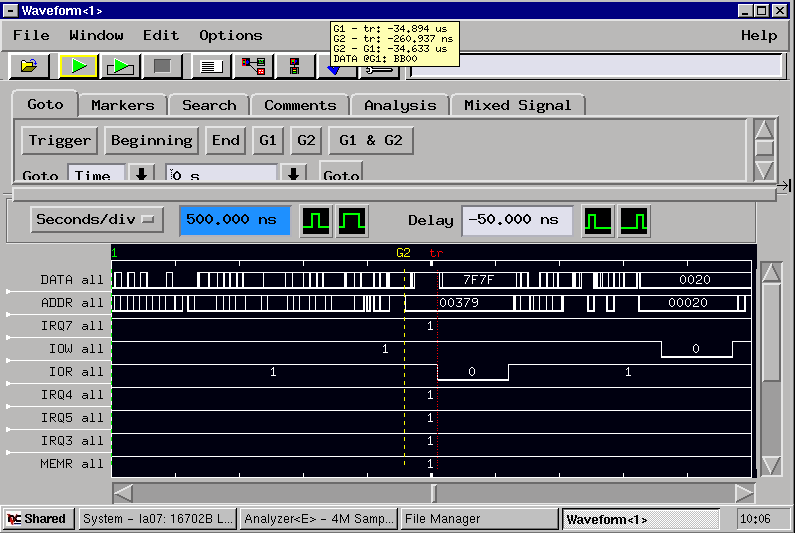
Line 9: Saving End of Interrupt address to al register

Line 10 : moving al to priority interrupt controller to signify end of interrupt routine

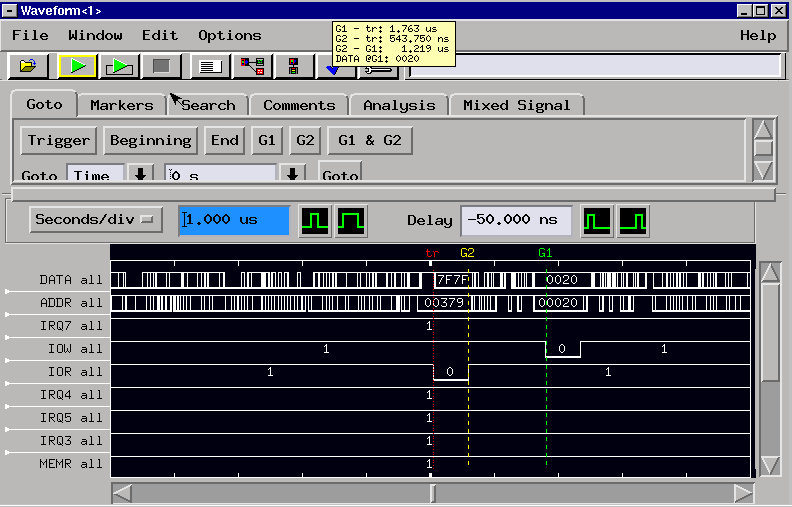
Line 11-13: Puting the values that were present in registers ax, ds and dx at start of the ISR

Line 14: making an interrupt return call to return to normal operation

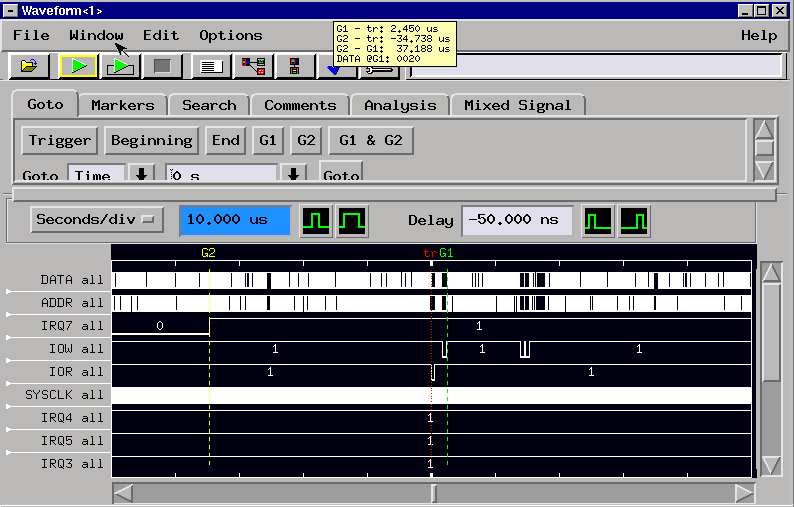
P1Q2. Upload a screenshot with timing from the interrupt request signal (IRQ7) to the first visible marker in the ISR. \*



P1Q3. Upload a screenshot with timing between two visible markers in the ISR. \*



P1Q4(a). Calculate the full interrupt service time (i.e time from interrupt request signal to ISR end) in microseconds and enter the result of your calculations below. Hint: There are few instructions at the end of the ISR that were not included in your timing but can be taken into account knowing the SYSCLK frequency. \*



37.188us

P1Q4(b). Explain how you calculated the full interrupt service time. \*

Measured time from when IRQ7 goes high to the end of when IOW goes low

P1Q5(a). Explain what you observed when the square wave frequency was increased. \*

Leds were to count up sequentially faster with the increasing square wave frequency till it was set to a frequency where the counting was no longer visible. bound this point the counting

P1Q5(b). Upload a screenshot showing how the system malfunctioned when the square wave frequency was increased. \*

P1Q5(c). At what square wave frequency did the system start to malfunction? Give your answer in MHz. \*

40 KHz = 0.04Mhz

P1Q5(d). Explain how the system malfunctions at and above that frequency. \*

* Stops visibly counting, looks more like a wave of lights – gets worse, IRQ7 triggers but doesn’t read or write because its occurring too fast

P1Q6(a). Why would someone enable interrupts inside an ISR? \*

P1Q6(b). How can a system fail if interrupts are enabled inside an ISR? \*

P1Q7. How can you decrease the latency of the ISR in the LAB2A? \*

make the interrupt routine shorter - no interrupt priorities

PART 2 OF THE LAB

link for serial communication stuff

<http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/matlab_external/ch_seri8.html>

P2Q1. Upload a completed LAB2B code fully commented in .txt format. Explain the necessity/role of each line in the overall functionality of the program. Use your own words and do not include the comments already provided in the .pas file. \*

P2Q2(a). Briefly explain the following parameter of the serial port setup: "Baud Rate". \*

Baud rate is the maximum bit per second data transfer rate that a serial port possess

P2Q2(b). Briefly explain the following parameter of the serial port setup: "Data Bit". \*

Data bits are the bits in the sent signal that contain the information that is being sent i.e and error message or commands to external devices

P2Q2(c). Briefly explain the following parameter of the serial port setup: "Stop Bit". \*

signifies the end of data signal

P2Q2(d). Briefly explain the following parameter of the serial port setup: "Parity". \*

parity bit provides a method of error checking

P2Q3(a). In serial communication, what are some symptoms of sending bits too fast? \*

Errors will occur at the receiving end as the device will not be able to sample and process the data fast enough

(b). In serial communication, what are RTS and CTS hardware flow control mechanisms? \*

RTS (Request To Send) and CTS(Clear To Send). Both mechanism check to see if it is okay to proceed with the data transfer

P2Q4(a). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 0 of the LSR. \*

shows that the data is ready, a byte was received by UART, buffer is ready for reading

P2Q4(b). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 1 of the LSR. \*

responsible for overrun error existence - the error occurs when the data is read slower than they are received

P2Q4(c). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 2 of the LSR. \*

parity error existence when parity doesn’t occur with the parameters that were set

P2Q4(d). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 3 of the LSR. \*

responsible for framing error existence - for when the last bit isn’t a stop bit

P2Q4(e). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 4 of the LSR. \*

break interrupt settings, if the received data is held at a ‘0’

P2Q4(f). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 5 of the LSR. \*

shows transmitter holding register is empty

P2Q4(g). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 6 of the LSR. \*

empty data holding registers

P2Q4(h). Search about the Line Status Register (LSR) of the serial communication, and briefly, explain bit 7 of the LSR. \*

error in received FIFO -