Experiment No. 6
Input / Output Design
ECE 441

Peter Chinetti

November 14, 2013

Date Performed:	October 10,17, & 24, 2013
Partners:	Zelin Wu
Instructor:	Professor Saniie

1 Introduction

1.1 Purpose

The purpose of this experiment is to introduce the user to the concepts of memory mapped I/O and interrupts.

1.2 Background

1.3 Memory-Mapped Input / Output (I/O)

There are two different types of philosophies when it comes to interfacing to I/O devices. These are known as Memory Mapped I/O and Isolated I/O. The MC68000 uses the memory-mapped I/O philosophy. This implies that all I/O devices are accessed by reading and writing to memory locations within the microprocessors address space.

1.4 Interrupts

The MC68000 microprocessor is equipped with 3 interrupt request signals (*IPL2, *IPL1, *IPL0) which provide a maximum of 7 distinct interrupt levels, and a normal operating level (level 0). The Status Register contains three Interrupt Mask Bits (I2, I1, I0) which are the logical complement of the interrupt hardware signals. When dealing with interrupts, the device requesting service activates a hardware signal called an Interrupt Request line. (*IRQ). The interrupt request lines from several peripheral devices are prioritized, encoded and inputted to the three interrupt request lines of the MC68000. The interrupt requests are made pending until the CPU completes the current instruction being executed.

Once the instruction is completed, the current state of the processor is saved on the stack, and an interrupt acknowledge cycle begins. The MC68000 compares the incoming interrupt request to the current interrupt priority level stored in the Status Registers Interrupt Mask Bits. If the incoming level is less than or equal to the current interrupt priority level, then the interrupt is not serviced.

2 Lab Procedure and Equipment List

2.1 Equipment

- SANPER System
- Computer with TUTOR software

2.2 Procedure

Design and implement I/O device, then test with a test program.

3 Results, Analysis and Discussion

3.1 Test Program

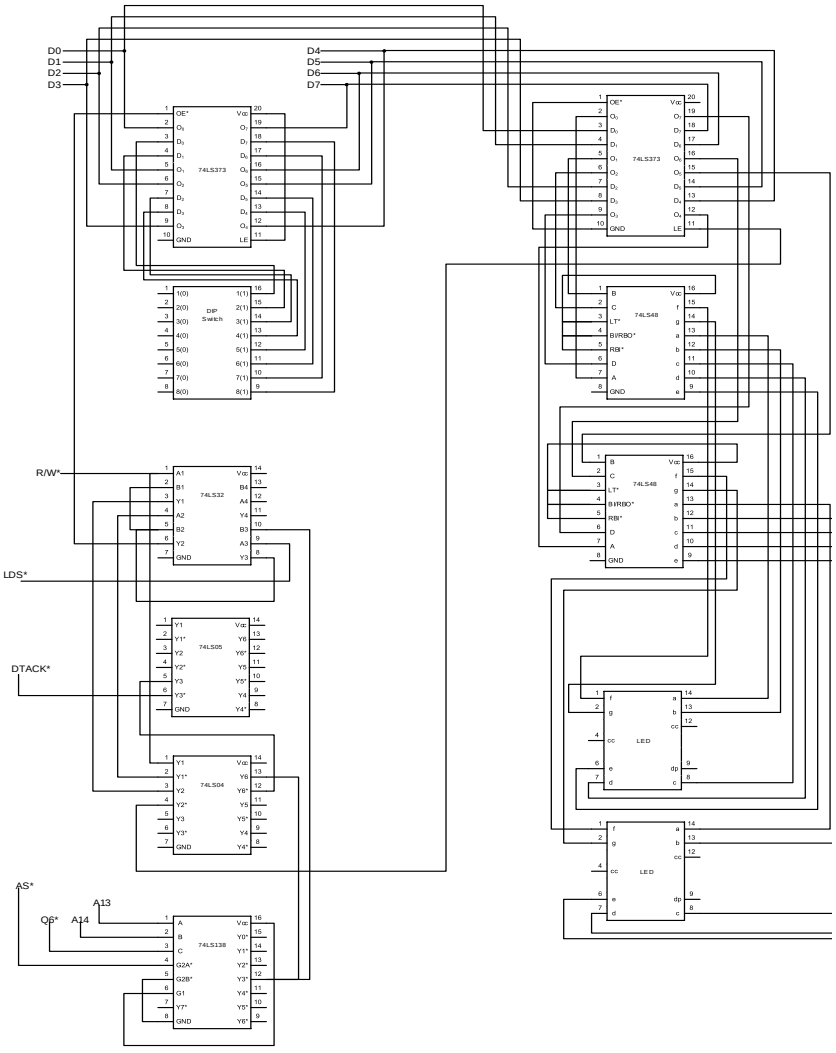
```
1           ORG $900

3   START   MOVEA.L #$06E001,A0 ;init with address
            MOVE.B (A0),D0      ;Read
5           MOVE.L #$000001,D1  ;init
            MOVE.L #$000099,D2
7           MOVE.W #$0000,D3
            MOVE.W #$FFFF,D4
9
    CHECK   CMP.B D2,D0         ;are we done
11          BEQ DONE
            MOVE.B D0,$06E001   ;write to LEDs
13
    WAIT    ADDI.W #$0001,D3    ;spin up D3
15          CMP D3,D4           ;check for doneness
            BEQ COUNT
17          BRA WAIT            ;spin again

19  COUNT   ABCD D1,D0          ;decimal add
            BRA CHECK
21
    DONE    MOVE.B D0,$06E001   ;write last value
23          MOVE.B #228,D7      ;back to tutor
            TRAP #14
25
    END START
```

lab6.X68

3.2 Diagram


D0
D1
D2
D3
D4
D5
D6
D7
74LS373
DIP Switch
74LS48
R/W*
74LS32
LDS*
74LS05
DTACK*
74LS04
AS*
A13
Q6*
A14
74LS138
LED

3.3 ABORT Switch

The diagram on page 2 of the User Manual has an error, so I can not be read. Instead, I discuss how I would design the circuit. I would wire the button into an encoder connected to the IRQ lines. When the button was pressed, it would encode a 7 (111) onto the lines. Interrupt number 7 is unmaskable,so it would ensure that the system would break out of the currently executing program. The ABORT button works by ensuring the vector generated by the interrupt level 7 points to a program which can recover back to TUTOR.

3.4 Auto-Vectored and User Vectored Interrupts

Auto-vectored interrupts do not have a vector provided by the interrupting peripheral, vectored interrupts do.

3.5 IACK Cycle

a. Peripheral signals on its interrupt line

b. Line is encoded into an interrupt level

c. Level is checked against SR

d. If higher, 111 is place on FC, and level is placed on $A_3 - A_2$.

e. $A_3 - A_2$ are decoded into a specific IACK line.

3.5.1 Vectored Interrupt

a. Peripheral writes it's vector onto the data bus and asserts DTACK

b. 68000 executes the handler pointed to by the vector

a. Peripheral writes it's vector onto the data bus and asserts DTACK

b. 68000 executes the handler pointed to by the vector

3.5.2 Auto-Vectored Interrupt

a. Peripheral asserts VPA

b. 68000 internally generates an interrupt vector, then executes it.

4 Conclusions

This experiment was accomplished. An I/0 device was built and tested.