8086 Experiments with Proteus

Experiment Three: Timer and Interrupt Applications

1. Goals

1) To learn how to use the PIT 8253 including its internal structure, control word format, addresses for each counter and the control word register, all modes of counters (especially, Mode 0, 2 and 3) and their usage.

2) To understand the basic concept of interrupts including the procedure of interrupts (i.e., NMI, maskable and software), the INT instruction, interrupt vector and interrupt vector table (IVT), interrupt service routines (ISRs).

3) To be familiar with the way to extend peripheral I/O devices.

2. System Design

Refer to the given Proteus file: 8086_experiment_three.DSN, and the 8253demo.asm file.

3. Requirements

1) **8255 operations:**
   a) Given the 8086_experiment_three.DSN file, give out the addresses of the four registers of the 8255 chip in the I/O address space. Notice how the data pins of 8255 are connected with the system data bus and how PA, PB and PC ports are used;
   b) Modify the demo program so that the 4-digit LED display simultaneously shows the last four digits of your student ID;
   c) Write a program to poll the status of the switch connected to PC0, and then output the value to PC6 without changing the status of other PC lines;

   Note: *it is straightforward to use BSR mode to set individual PC lines. It is reported, however, that Proteus has problem with BSR mode. Therefore, you need to use normal input/output modes of 8255.*

2) **8253 operations and timer applications:**
   a) Given the 8086_experiment_three.DSN file, give out the addresses of all counters and the control word register of the 8253 chip in the I/O address space;
   b) Connect PC6 of 8255 to the GATE0 pin of the 8253 and set up the proper mode and the initial count for Counter 0 so that it can generate a 10ms per cycle square wave. Run your program and study the following cases:
   ① Turn off the switch connected to PC0 and observe the change of OUT0 using the simulated oscilloscope;
   ② Turn on the switch connected to PC0 and observe the change of OUT0 using the simulated oscilloscope. You should be able to explain the reason of what you see.

c) Modify the diagram so that OUT0 is connected to CLK1 and set up the proper mode and the initial count for Counter 1 so that it can generate a 1s per cycle square wave.

3) Interrupt application 1:

a) You should understand the circuitries for generating an INTR signal and clearing the INTR when INTA is effective, respectively;

b) You need to pick your own interrupt type according to the following equation:

$$n = Last\ four\ digits\ of\ your\ student\ ID\ \%\ 224 + 32$$

c) Write an ISR which reads in the status of PC7, inverts the value and sends the inverted value back to PC7;

d) Try to use software interrupt (i.e., put **INT *n*** instruction in your main program) to test your main program and ISR both work. If success, each interrupt will change the status of LED D13.

e) Try to use hardware to generate maskable interrupts.

① Modify the diagram to connect the Q output pin of U15 to the INTR pin of the 8086 and connect the R input pin to the INTA pin of the 8086;

② Connect the OE pins of U11A and U11B to the INTA pin of the 8086 and set DSW1 switches according to the interrupt type n you have picked;

③ Setup the IVT according to your interrupt type n and ISR so that the CPU can find your ISR each time there is an external interrupt of type n happening;

④ Remove all **INT *n*** instructions from your main program and set the IF. You should notice that the changes of LED D13 each time you press the button BT1.

4) Interrupt application 2:

You need to write programs so that the last four digits of your student ID displays on the 4-digit LED display one by one in turn each time you press the button BT1.

Programming hints: *You can write a main program to conduct an infinite loop. Within each loop, the program displays one of the four digits controlled by a variant. You also need to write an ISR which varies the variant to select the next digit to display (the next digit of the fourth digit would be the first digit). In this way, each time when you press the button, there will be an interrupt which leads to the execution of your ISR.*

5) Interrupt application 3:

Use Counter 2 to function as a watchdog timer. Watchdog timers are often used in embedded systems to enforce the control of the system. For example, the system will set up a watchdog timer and if the system is not corrupted and just before the timer expires, the system will reset the timer. Otherwise, the watchdog timer will expire (this also indicates that the system is down or corrupted) and reset the whole system. After the reset, the system will take back the control. In this experiment, we ask you to set up Counter 2 to work in mode 0 which simulates a watchdog timer of 50ms. You need to write programs to use Count 0 or Count 1 to periodically reset the timer ("feed the dog") before it expires. If you fail to do that, the watchdog alarm (i.e., LS1) will sound and the watchdog LED (i.e., D11) will be on.

Programming hints: *You need to utilize Counter 0, Counter 1 or their combination* *to periodically generate interrupts so that the corresponding ISR can reset the initial count of Counter 2 which feeds the dog.*

a) Modify the diagram to connect OUT1 to the CLK pin of U15 so that each pulse of OUT1 will generate an interrupt request to the CPU. Describe your observation.
b) Modify the diagram to connect OUT0 to the CLK pin of U15 so that each pulse of OUT0 will generate an interrupt request to the CPU. Describe your observation.

6)* Interrupt application 4:

You need to write programs so that the number of interrupts happened is displayed on the 4-digit LED display.

4. Notes

1) As you need to change the diagrams several times, you'd better save each modified diagrams and your corresponding programs so that it is easier for tutors to check your results.
2) Requirements marked with a "*" is optional. You would get extra credits for those requirements.
3) Proteus has some bugs when doing 8086 simulations. We have add patch code for those bugs in the demo programs. You should simply ignore this part and leave it un-changed.

5. Results

1) You should be able to demonstrate your experimental results.
2) You should hand in your experiment report describing the observations you have observed and the lessons you have learned from the experiment and present your code and comments.