ENGR 323L: Microprocessor Systems
Department of Engineering, Trinity College
Instructor: Professor T. Ning

Laboratory #1: Design and build an 8051 embedded system to perform digital watch function via time-division-multiplexing (TDM) display

Problem Statement:
Among many microcontrollers that have found a wide spectrum of applications in automobiles, appliances, video games, medical equipment, and various modern tools, the 8051 microcontroller (single-chip microcomputer) invented by Intel in 1980 for embedded systems and its variants are the most widely utilized 8-bit microcontrollers. This digital watch lab is to familiarize students with the process of building an 8051 embedded system that can perform accurate timing functions; it involves schematic design of circuit layout, hardware construction, and assembly language programming using multiple interrupts and TDM control of multi-digit LED displays.

The target system will display minutes and seconds and also emulate the a stopwatch including resetting, starting, and freezing the count by a 100^{th} of second. Immediately after reset, the designed watch will perform normal minute-second timing, i.e., counting from "00.00" to "59.59" in one second intervals. The system will reset to "00.00" and repeat counting after one hour. The minute-second count will be displayed through a four 7-segment digit display (two digits for minutes and two for seconds).

The stop-watch mode will be activated by an external interrupt (e.g., IE0) through a push-button. The interrupt service routine is to change the display update routine to an interval by a hundredth of a second. The next activation of the same external interrupt should freeze the current count. When activated by another external interrupt (e.g., IE1), the stopwatch mode is disabled the target system resumes min-sec counting mode. *Note that the normal min-sec counting should continue in the background when the system is temporarily switched to stopwatch mode.*

Design Strategy:
A simple static control of four 7-segment digit displays demands many I/O pins and consumes too much energy. For example, through a BCD-7SEG decoder/driver IC (7447), four input bits are required to control one digit and sixteen input bits will be necessary for four digits. In addition, a lit single LED segment usually draws 10-20 mA current. To avoid the heavy demand of many output pins and energy consumption, a more sophisticated dynamic control method that utilizes time-division-multiplexing (TDM) is adopted in this lab.

In the TDM approach, several output pins (i.e., PORT1 pins) are shared by more than one 7-segment displays and each digit is only refreshed frequently enough to make it seem as

statically lit to human eyes. The design for refreshing $X_1 \rightarrow X_2 \rightarrow Y_1 \rightarrow Y_2 \rightarrow X_1 \ldots$ sequentially is illustrated in Fig.1. A TDM display approach has the following features:

- All four 7-seg displays share the same output pins
- Only one 7-seg display is lit at any instance to save energy
- Periodic interrupt-driven control frees CPU to other tasks

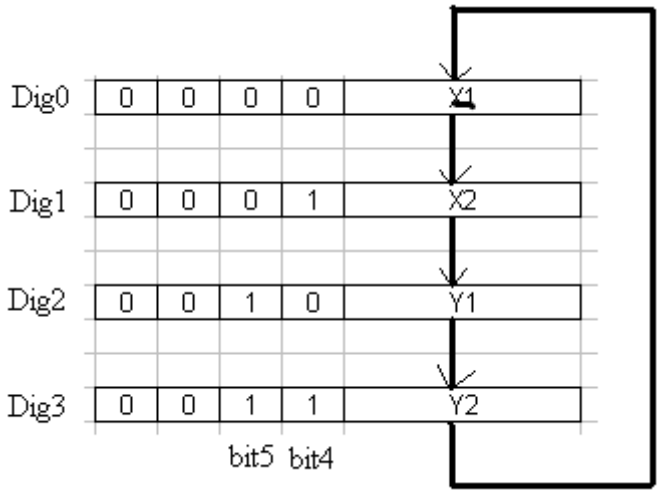


Fig. 1: Refreshing cycle with TDM approach

Purpose:

- To build an application specific 8051 embedded system using required hardware components and assembly program
- To become familiar with the TDM control and on-chip timers of an 8051 microcontroller
- To develop, implement, debug, and execute assembly programs on an embedded system
- To use timer-overflow and external interrupts to provide the timing control

Procedure:

I. Schematic design of an embedded system based on the 8051 µcontroller

Use ***WinDraft*** to create a complete schematic design based on an 8051 microcontroller. The design will include an external 2k x 8 EEPROM (2816) and an external circuit with BCD-7-SEG decoder (7447), demux (138/139) and four 7-SEG displays (LDQ-M514RI). Label clearly the connecting pins of PORT-1 and other chips. The schematic should also show all required connections of the 8051, e.g., $\overline{PSEN}$, ALE, $\overline{EA}$, etc. An example connection diagram is shown in Fig.2.

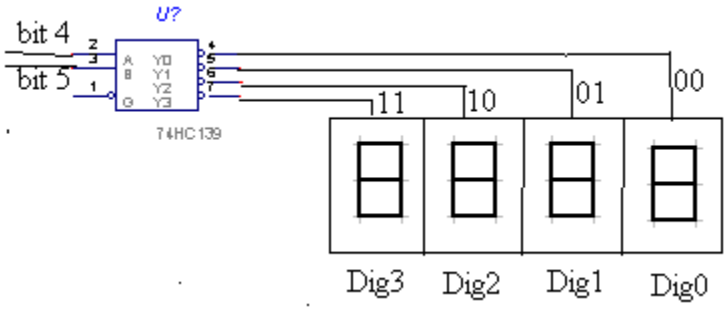
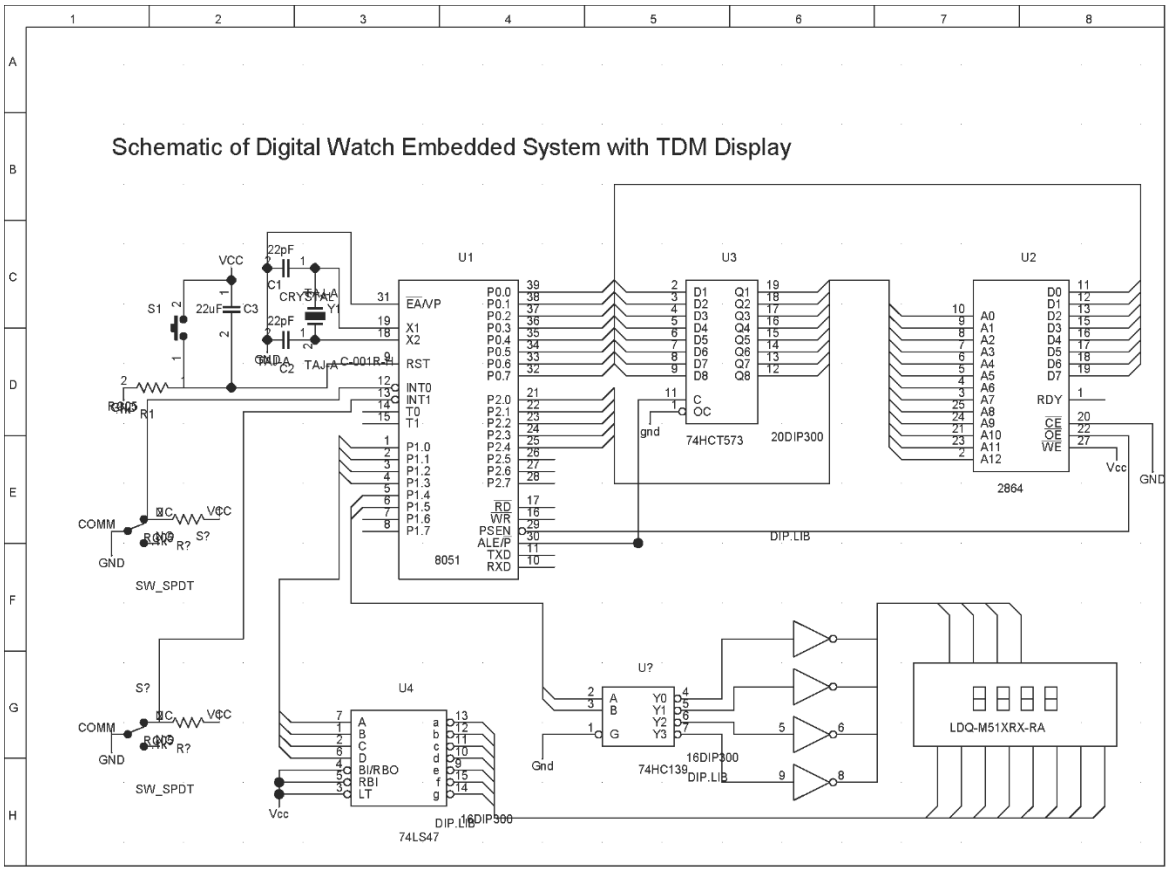


Fig. 2: An example to use bits in P1.4 and P1.5 to select the target display digit

A schematic of a complete system using TDM to display 4 digits is given below



II: Implement 8051 based embedded system

Use wire wrapper and a PC board to build the 8051 system that includes a multiplexing display circuit. Use name labels and colored wires to facilitate the wiring process. A complete wire-wrapped system is shown in Fig.3.

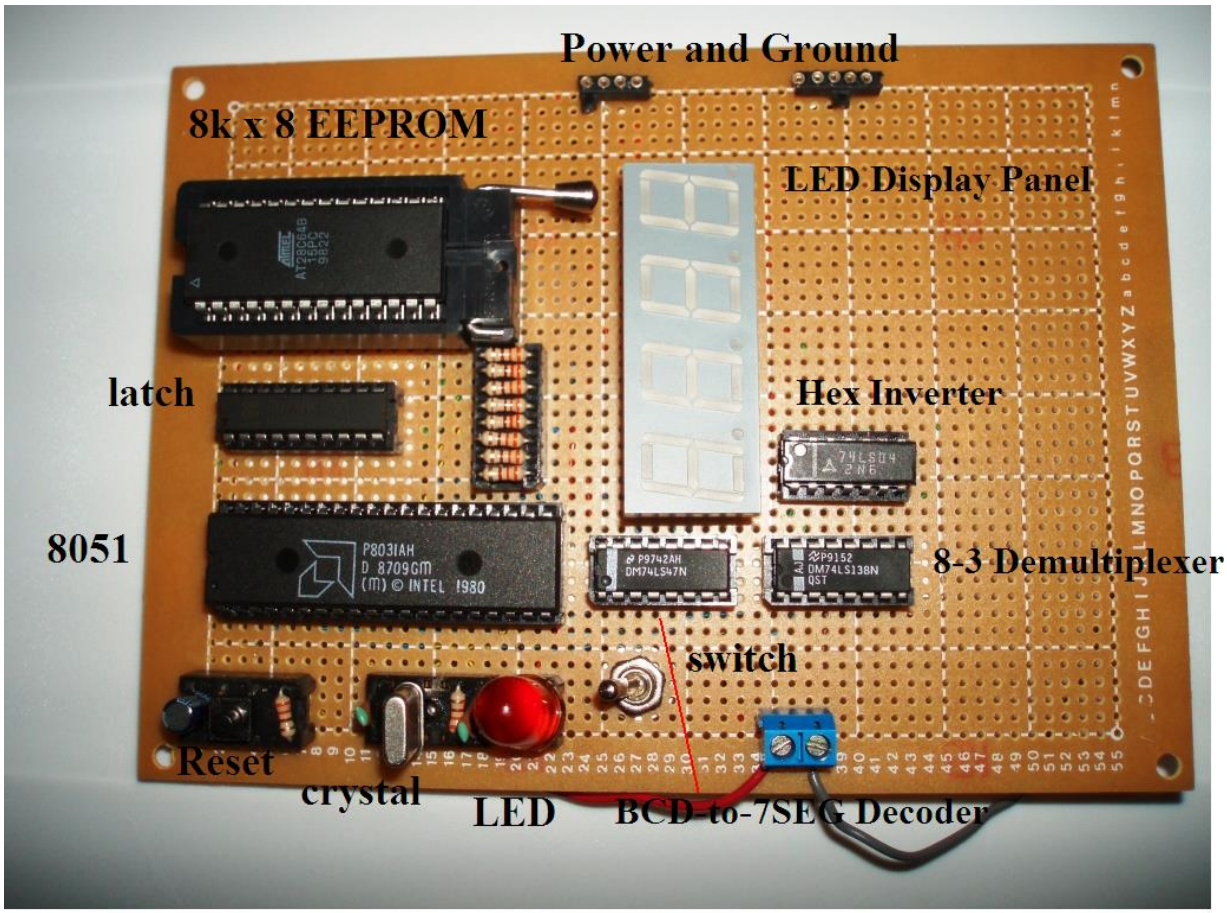


Fig. 3: A complete 8051 system with a 4-digit display and external interrupts

III: System Validation

Use Keil-µvision to develop and test the assembly program. Make sure that PORT-1 pins are properly controlled to match their intended use and wiring connections.

- Develop the assembly program, an assembly program ended with an extension of *A51*, e.g. *myfile.A51,* under Keil-µvision.
- Debug the assembly program through simulator. Step through to verify its performance.
- Build the project and generate a HEX file to be programmed onto the EEPROM.

- ***Build and test the 4-digit display peripheral circuit. Confirm its functionality using the logic switches of the Cadet board to emulate Port_1 pins designed to control the MUX and BCD chips.***

- ***Build and verity the 8051 system by outputting a single number on a designated digit. Repeat the steps several times to verify that the 8051 system can correctly control the 4-digit display component.***

8051 Special Function Registers (SFR)

Initialize the jump address for Timer_0/Timer_1 interrupt service routine

Interrupt	Jump loc.	Default Priority
IE0	3H	1 (Highest)
TF0	BH	2
IEI	13H	3
TF1	1BH	4
RI-or-TI	23H	5 (Lowest)

Configuring Timer Mode Control (TMOD) and Timer Control (TCON) SFRs
TMOD (89h)

Bit	Name	Explanation of Function
7	GATE1	When this bit is set the timer will only run when INT1 (P3.3) is high. When this bit is clear the timer will run regardless of the state of INT1.
6	C/T1	When this bit is set the timer will count events on T1 (P3.5). When this bit is clear the timer will be incremented every machine cycle.
5	T1M1	Timer-1 mode bit
4	T1M0	Timer-1 mode bit
3	GATE0	When this bit is set the timer will only run when INT0 (P3.2) is high. When this bit is clear the timer will run regardless of the state of INT0.
2	C/T0	When this bit is set the timer will count events on T0 (P3.4). When this bit is clear the timer will be incremented every machine cycle.
1	T0M1	Timer-0 mode bit
0	T0M0	Timer-0 mode bit

TCON (88h) – bit addressable

Bit	Name	Explanation of Function
7	TF1	Timer-1 Overflow. This bit is set by the microcontroller when Timer 1 overflows.
6	TR1	Timer-1 Run. When this bit is set Timer 1 is turned on. When this bit is clear Timer 1 is off.
5	TF0	Timer-0 Overflow. This bit is set by the microcontroller when Timer 0 overflows.
4	TR0	Timer-0 Run. When this bit is set Timer 0 is turned on. When this bit is clear Timer 0 is off.
3	IE1	External interrupt-1 edge flat. This bit is set when high-to-low signal received on P3.3, cleared when the ISR located at 0013h is served.
2	IT1	External interrupt-1signal type control. Set to enable falling edge trigger of IE1, cleared to allow low-level signal on P3.3 to generate an interrupt.
1	IE0	External interrupt-0 edge flat. This bit is set when high-to-low signal received on P3.3, cleared when the ISR located at 0003h is served.
0	IT0	External interrupt-0 signal type control. Set to enable falling edge trigger of IE1, cleared to allow low-level signal on P3.3 to generate an interrupt.

Timer Operation Mode Selection

TxM1	TxM0	Mode	Operation
0	0	0	13-bit timer/counter TLx (5 bits) THx (8 bits)
0	1	1	16-bit timer/counter
1	0	2	8-bit auto-reload (THx stores the reload value)
1	1	3	split-mode: two 8-bit timers

The following command will configure Timer-0 and Timer-1 as 16-bit timers.

```
MOV TMOD, #11h
```

The following command turns on Timer-0 to run.

```
SETB TR0
```

Upon executing these two instructions, Timer-0 will start to count every machine cycle and reset to 0x0000 after it has reached 0xFFFF.

Example: To set the desired interrupt interval in machine cycles (reload value for THx and TLx)

e.g., initialize Timer_0 timer with a desired value

```
MOV TH0, #0ECh
MOV TL0, #77h
```

Example: Enter an interrupt service routine

e.g., setup a periodic interrupt service routine using Timer-0 Overflow Interrupt (TF0).

```
    ORG 000Bh
    AJMP ISRTF0:

ISRTF0:
    CLR EA                  ; disable all interrupts to protect the critical region
    MOV TH0, #0ECh          ; reload Timer-0 startup value
    MOV TL0, #77h
    SETB EA                 ; re-enable all interrupts
    …..
    …..
    RETI
```