SEVEN-SEGMENT DISPLAY

Revision 04.04.30

_____

Class

_____

Instructor / Professor

© 2003 Axiom Manufacturing
2813 Industrial Lane - Garland, TX 75041

Phone: 972-926-9303
http://www.axman.com
email: sales@axman.com

CONTENTS

1 REQUIREMENTS 3
1.1 HARDWARE 3
1.2 SOFTWARE 3
2 GETTING STARTED 4
3 LAB PROCEDURE 4
3.1 DESCRIPTION 4
3.2 DETAILED STEPS 5
4 SEVEN SEGMENT PROGRAM 8
4.1 PROGRAM DESCRIPTION 8
4.2 RUNNING THE PROGRAM 8
4.3 SEVEN SEGMENT PROGRAM SOURCE CODE 9
5 QUIZ 10
6 SCHEMATIC 11

1 REQUIREMENTS

1.1 Hardware

To complete this lab, **the following hardware is required**:

- Axiom CML-12C32 Development Kit
- PC running Windows OS
- Project Starter Kit, which includes:
  - (1) MAN6960 Seven Segment Display
  - (1) 470 ohm Resistor Array
  - (1) 74LS247 Segment Driver
  - (23) Jumper Wires

1.2 Software

The CML-12C32 board used in this experiment comes with all the software needed to complete this project.

There are many additional utilities included on the boards support CD that can make developing your own projects easier. The CD contains example source code, documentation and experiments for all Axiom development boards. You can even download the latest versions of the software and documentation free from our web site at: www.axman.com.

Also included is an integrated development environment, called AxIDE, for communicating with the board (via the serial port) and for reading and writing its flash memory. To complete this Lab, you should have this program installed on a PC running Microsoft Windows (95/98/2000/XP).

NOTE: This lab does not teach you how to use the AxIDE terminal interface or the MON12 Monitor program to modify memory and upload programs. It assumes you're already familiar with these procedures. Refer to your board manual for details on installing and using this software, including a tutorial for using AxIDE.

CAUTION

Devices used in this lab are static sensitive and easily damaged by mishandling. Use caution when installing wires and devices onto the board to prevent bending the leads.

Experiments should be laid out in an orderly fashion. Start your lab time with the bench clean and free of metal objects and leave the lab area in a clean condition by picking up loose parts, wires and small objects.

2 GETTING STARTED

This lab will show you how to add an external seven segment display device to your Axiom development board. In this experiment only one display is used, though you can easily add more to display larger numbers.

A seven segment display is a group of seven LED's arranged in segments. These segments are generally used to display numbers 0 thru 9. In this example, an I/O port on the microcontroller will be written to output a group of 4 bits representing the number. These bits are applied to the seven segment driver causing it to light up the proper segments for displaying that number.

This type of display is used in appliances, machinery, cars, clocks, timers and many other products. They come in several colors such as red, green, blue and white.

> **HOW IT WORKS**: All LED's in the display are common anode. By connecting the common anode to +5 and using a current limiting resistor on the cathode side, the driver is able to turn each segment on. Segment intensity is dependent on the current flow and should not exceed the limit of the segment.

3 LAB PROCEDURE

The lab is arranged in a series of steps. Each step should be completed before moving on to the next one, which builds on prior ones. Repeat each step as many times as necessary to become familiar with it. You will find it easier to complete more complex experiments after mastering the simple ones.

As an aid to keeping track of location it's a good idea to mark each step as it's completed, since the experiment will fail if anything is skipped.

3.1 Description

This example uses PORT B on the C32 microcontroller which, in single chip mode, can be used as a bi-directional (input/output) port. PORTB is located at address $0001 on the HC12. We will use bits 0-3 for the 4 bit BCD number. Writing directly to this port will apply a number to the seven segment driver. For example, writing $00 will display a "0" and writing $09 will display a "9".

Before using PORT B however, its Data Direction Register (DDRB) must be initialized. Writing $0F to the DDRB address, $0003, sets bits 0-3 as outputs. The detailed steps in the following section walk you thru this process of setting up the port for use and displaying different digits.

3.2 Detailed Steps

This section describes how to build the display and test it with the monitor running on the CML-12C32. In the next section, you'll see how to write a simple program that uses this display.

In the following steps, PORTB is the register byte at address $0001. DDRB is the one at address $0003

> **NOTE**: To complete these steps you must be familiar with modifying register contents on your board. For example, to write $0F to the DDRB register with the MON12 monitor, type `MM 0003` at the monitor prompt and press <enter>. Then type `0F` <enter> then `.` to end modify mode. Type Help for more commands.
>
> You can use a different monitor or debugger if you prefer, such as the GNU GDB.

1. Verify power is **NOT** applied to development board.

2. Install the seven-segment display, the driver IC and the resistor array on the breadboard as shown in **Figure 1**. Also make sure the circle ° on the corner of the display is lined up as shown.

3. Install the jumper wires on the board as follows:

MCU PORT - Breadboard

GND----------GND(-)
+5V----------+5V
PB0----------U1-7
PB1----------U1-1
PB2----------U1-2
PB3----------U1-6

Breadboard --- Breadboard

GND(-) ---------- U1-8
+5V ---------- U1-16
+5V ---------- U2-8
U1-9---------- RN1-9
U1-10 ---------- RN1-10
U1-11 ---------- RN1-11
U1-12 ---------- RN1-12
U1-13 ---------- RN1-13
U1-14 ---------- RN1-14
U1-15 ---------- RN1-15
RN1-2---------- U2-9
RN1-3---------- U2-10
RN1-4---------- U2-7
RN1-5---------- U2-6
RN1-6---------- U2-4
RN1-7---------- U2-2
RN1-8---------- U2-1

4. Configure the CML-12C32 board to start the MON12 debugger by setting the jumpers to their default positions (No_Auto and MEM_EN are ON.)

5. Apply power to the board.

6. Using the monitor, write $0F to DDRB. This forces bits 0,1,2,3 of port B to be outputs.

7. Write $00 to PORTB, verify display is a “0”. This forces all outputs low, thus representing a zero in BCD.

8. Write $01 to PORTB, verify display is a “1”. This forces PORTB bit 0-3 to a $01, thus representing a one in BCD.

9. Write $02 to PORTB, verify display is a “2”. This forces PORTB bit 0-4 to a $02, thus representing a two in BCD.

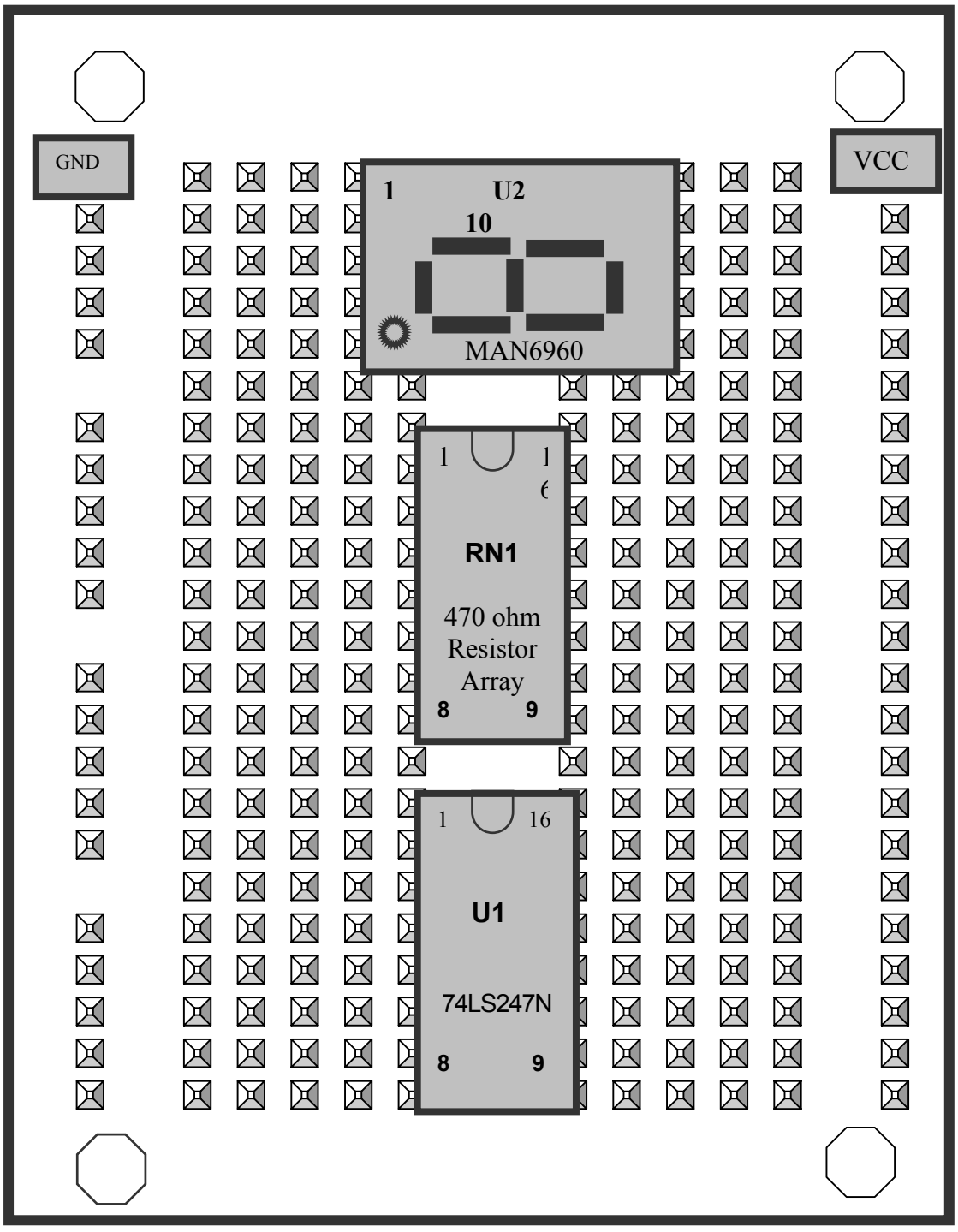
10. Write the following hex values in order: 03, 04, 05, 06, 07, 08, 09 to PORTB, verify each number is displayed “3”, “4”, “5”, “6”, “7”, “8” and “9”.

11. Write $0F to PORTB then verify the display is blank. This forces PORTB bits 0-3 to $0F, representing an invalid decimal number and no number is displayed.

These steps show how easy a display can be added to a microcontroller. Just one digit was used in this example, but you can easily add additional digits.

A four-digit bank, for example, can be used by multiplexing the digits or using one of the many other chips available on the market that have all necessary logic inside for driving several digits or LED’s. A clock, for example, would require at least four digits or six digits with seconds. A temperature monitor would require three digits on those hot days of 113 F. A digital voltmeter could require eight digits for precision readings.

Figure 1


GND
VCC
1
U2
10
MAN6960
1
RN1
470 ohm
Resistor
Array
8
9
1
16
U1
74LS247N
8
9

4 SEVEN SEGMENT PROGRAM

The previous section described a method of controlling the display manually by entering codes by hand. While this method is useful for testing and experimenting, once the hardware is working you'll want to write a software program to control the display.

This section will describe how to write such a program in assembly language. The full source code listing is at the end of this section. Both source code and assembled executable for this example can also be downloaded from the Axiom web site: **www.axman.com**.

If viewing this on your PC, you can also copy and paste the source code below into a text editor (such as notepad) then save and assemble it using AxIDE.

Refer to the owner's manual of your board for instructions on creating software and running programs for your development board.

4.1 Program Description

The program must first initialize PORTB to all outputs. Register A is cleared to $00 then Register A is written to PORTB. This displays the digit 0 which will remain on the display for the length of the delay subroutine.

Next register A is increment by one. The new value of A is checked to see if it is greater then 9. If so, the program starts over at the beginning. Otherwise the next digit is written to PORTB. Digits 0 thru 9 are displayed then the program starts over at the beginning.

4.2 Running the Program

1. Upload the assembled program 7-SEGCD.S19 into the RAM on your board. This program starts at address $0800, which is internal memory. The source code for this program is shown below.
2. Execute the program by typing `call 0800` in the terminal and pressing <enter>.
3. The seven-segment display will cycle thru digits 0 to 9 then repeat forever.
4. Press the reset button on the board or remove power to stop the program.

4.3 Seven Segment Program Source Code

This routine counts up from 0-9 on the seven segment display.

```
; Example Seven Segment Display on an CML-12C32

; Register Equates
PORTB:      equ     $0001        ; port B output register
DDRB:       equ     $0003        ; port B direction register

; This subroutine counts up on the seven segment display
            org     $0800        ; program starts here
            movb    #$1F,DDRB    ; all outputs
Loop1:
            clra                 ; clear register A
Loop2:
            Ldab    PORTB
            andb    #$10         ; get decimal point
            aba
            staa    PORTB        ; write digit to port T
            bsr     Toggle       ; Toggle Decimal Point
            bsr     Delay        ; delay for visual
            inca                 ; increment A
            anda    #$0F
            cmpa    #$0A         ; if greater then 9
            beq     Loop1        ; restart with digit 0
            bra     Loop2        ; display next digit

Delay:
            ldab    #$20
DelayB:
            ldy     #$FFFF
DelayA:
            dey
            bne     DelayA
            decb
            bne     DelayB
            rts
;
Toggle:
            brset   PORTB,#$10,Dec_High  ; get decimal point value
            bset    PORTB,#$10      ; set decimal high
            rts
Dec_High:
            bclr    PORTB,#$10   ; set decimal low
            rts
;
```

5 QUIZ

Answer the following questions based on the example presented in this lab.

1. What is the largest digit that can be displayed?
A. 5 **B**. F **C**. 10 **D**. 9

2. How many bits are required for one digit?
A. 8 **B**. 16 **C**. 4 **D**. 9

3. How many segments are in the display?
A. 5 **B**. 7 **C**. 10 **D**. 9

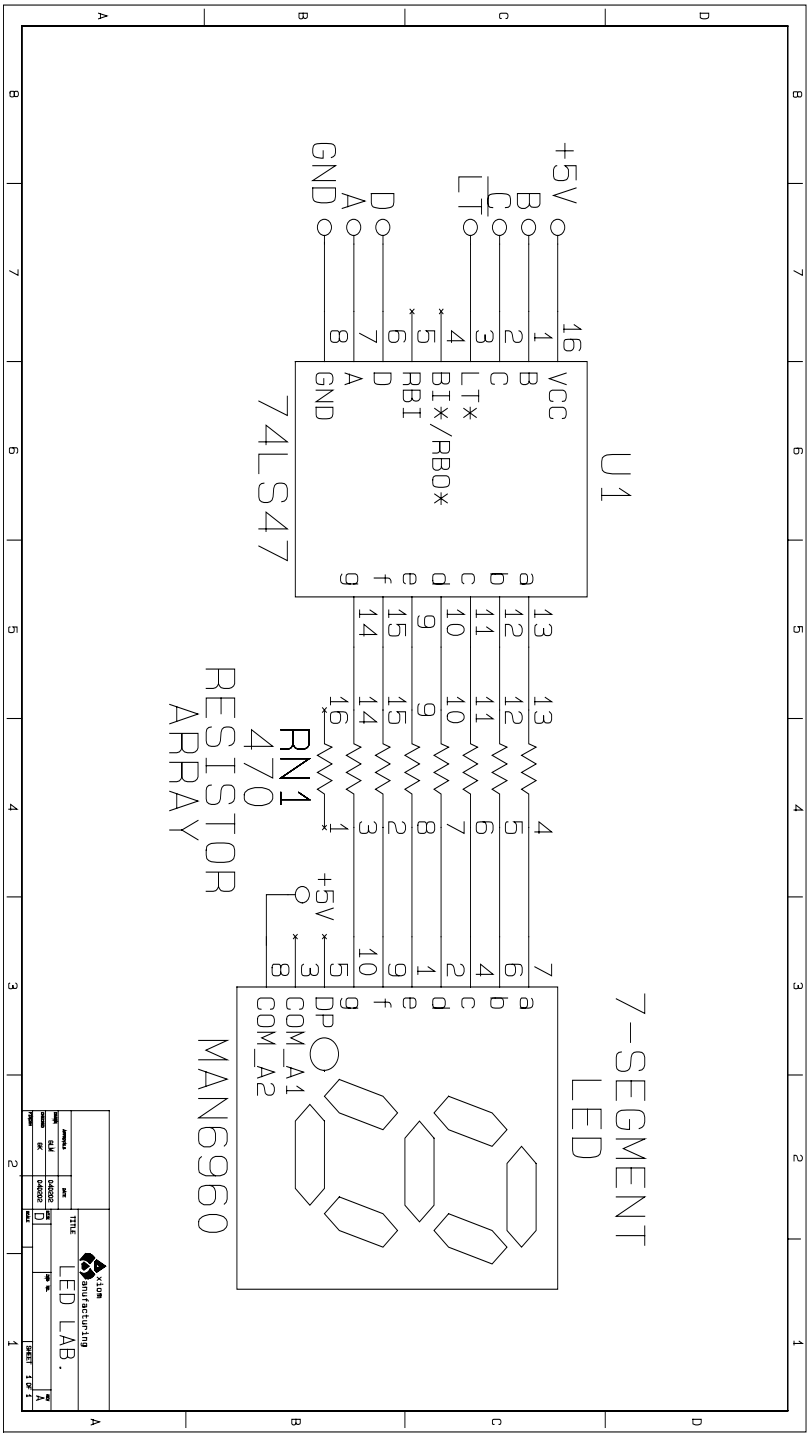
4. Which digit has the most segments lit?
A. 0 **B**. 5 **C**. 1 **D**. 8

5. What controls the segments of the display?
A. Digit Driver **B**. HC12 **C**. Segment Driver **D**. PORT B

6. All the LEDS in the display are?
A. Common drive **B**. Common Cathode **C**. Forward **D**. Common Anode

BONUS QUESTION: What controls the intensity of the segments?
A. Resistors **B**. Digit **C**. Segment driver **D**. Micro controller

6 SCHEMATIC


+5V
B
C
LT
D
A
GND
16
1
2
3
4
5
6
7
8
VCC
B
C
LT*
BI*/RBO*
RBI
D
A
GND
U1
74LS47
a
b
c
d
e
f
g
13
12
11
10
9
15
14
RN1
470
RESISTOR
ARRAY
4
5
6
7
8
2
3
1
+5V
DP
COM_A1
COM_A2
MAN6960
7-SEGMENT
LED
LED LAB.
TITLE