MX-909 500 in 1

Book #1
HARDWARE- Entry Course

CONTENTS

Book #2
HARDWARE - Advance Course

CONTENTS

9) More Digital circuits

10) Amusement in Electronic Circuits

11) Testing and Measuring Circuits

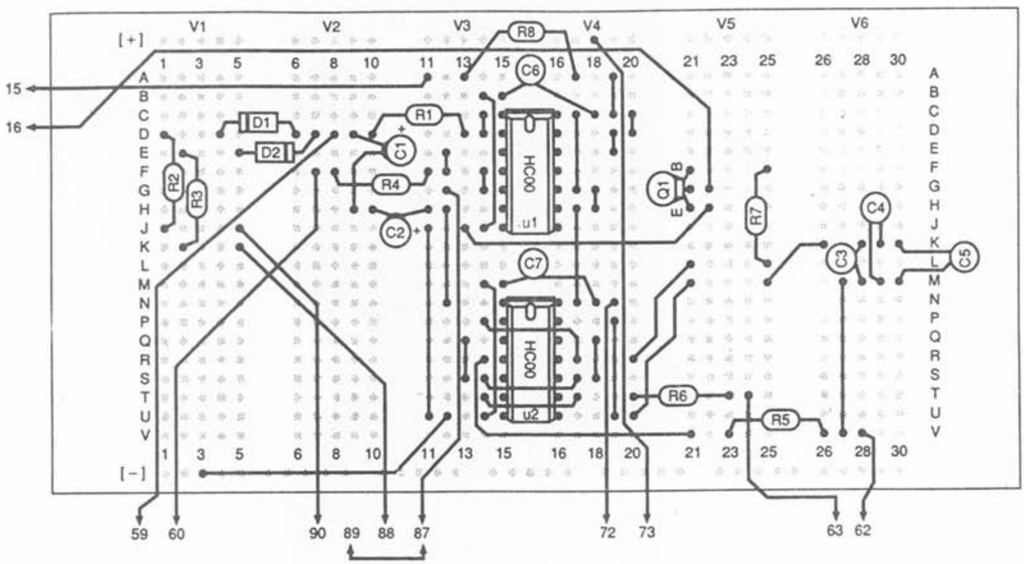
Book #3
SOFTWARE – Programming Course

CONTENTS

References:

Software Projects

PROJECT 216. TONE BURST GENERATOR



U1	74HC00	R2	47KΩ	R6	33KΩ	C2	10μF	C6	0.1μF
U2	74HC00	R3	47KΩ	R7	10KΩ	C3	0.1μF	C7	0.1μF
Q1	NPN	R4	33KΩ	R8	470Ω	C4	0.1μF	D1	Si
R1	220KΩ	R5	47KΩ	C1	10μF	C5	0.1μF	D2	Si

A tone generator is an oscillator that sends out signals repeated at regular intervals, as shown in Figure 1. As its name suggests, a speaker is usually used to let you hear the tone it makes. But in this project, we're going to use an **LED** to find out how it works.

You can see from the schematic, IC U2 is a tone generator whose frequency can be changed by 100K **control volume**. IC U1 is another generator which controls the start and stop of the tone generator. Its frequency can also be changed by turning 50K **control volume**, and its duty ratio can be adjusted using **S1** and **S2**. Remember what the duty ratio is? Refer back to project 53. The duty ratio is about 24% when **S1** is ON, and about 76% when **S2** is ON. This project can be used as a logic circuit whose waveform ends in an integer cycle.

When you finish assembling the project, turn power ON and see what the **LED** is doing. Does it blink ON and OFF as shown in Figure 1? Then, turn 50K **control volume** and press **S1** and **S2**, and see how the **LED** changes its blinking intervals.

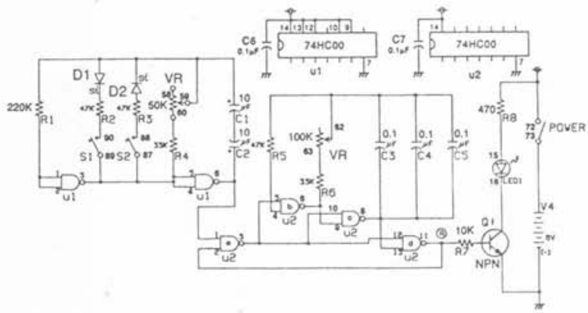
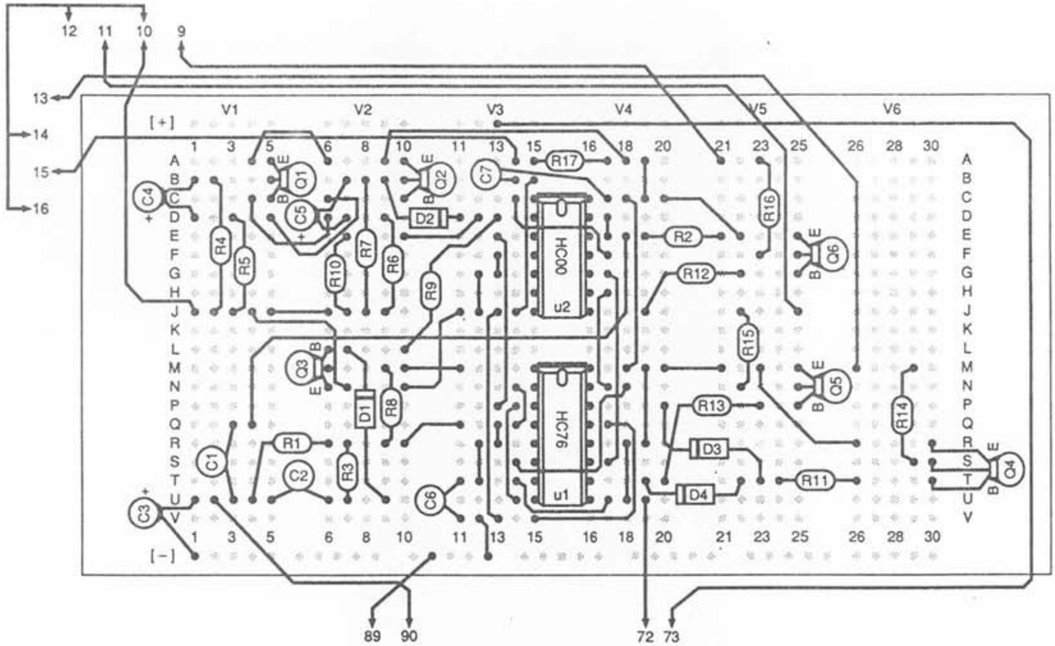


Figure 1

PROJECT 225. LEAPIN' LEDS

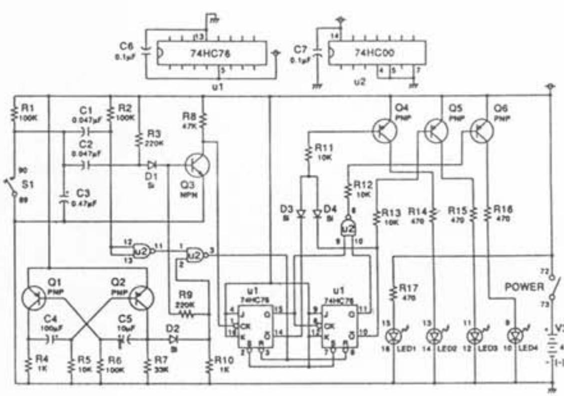


U1	74HC76	Q6	PNP	R7	33KΩ	R14	470Ω	C4	100µF
U2	74HC00	R1	100KΩ	R8	47KΩ	R15	470Ω	C5	10µF
Q1	PNP	R2	100KΩ	R9	220KΩ	R16	470Ω	C6	0.1µF
Q2	PNP	R3	220KΩ	R10	1KΩ	R17	470Ω	C7	0.1µF
Q3	NPN	R4	1KΩ	R11	10KΩ	C1	0.047µF	D1	Si
Q4	PNP	R5	10KΩ	R12	10KΩ	C2	0.047µF	D2	Si
Q5	PNP	R6	100KΩ	R13	10KΩ	C3	0.47µF	D3	Si
								D4	Si

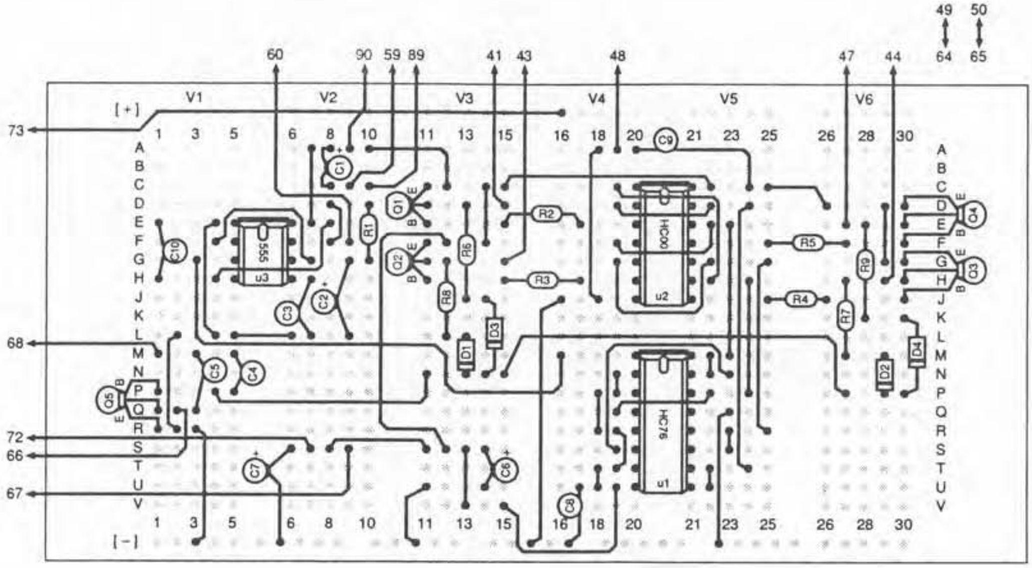
Here's game to see how fast you are on the trigger (or at least the key!). The object is to light **LEDs 1** through **4** as quickly as you are able or with as few presses of **S1** as you can.

To play, turn power ON. **LED 1** lights. Now press **S1** until **LED 2** lights. But, if you're not lucky, only **LED 1** lights. Continue to try to get all the **LEDs** to light up (**LED 1, 2, 3** and **4**).

The secret of this game is to press **S1** at exactly the right moment to light the **LED**. Timing's been an important part of all the digital circuits we've played with so far.



PROJECT 224. WHEEL OF FORTUNE



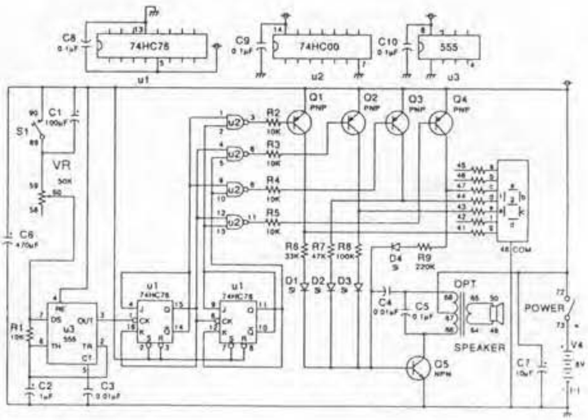
U1	74HC76	Q4	PNP	R5	10KΩ	C2	1μF	C8	0.1μF
U2	74HC00	Q5	NPN	R6	10KΩ	C3	0.01μF	C9	0.1μF
U3	555	R1	10KΩ	R7	47KΩ	C4	0.01μF	C10	0.1μF
Q1	PNP	R2	10KΩ	R8	100KΩ	C5	0.1μF	D1	Si
Q2	PNP	R3	10KΩ	R9	220KΩ	C6	470μF	D2	Si
Q3	PNP	R4	10KΩ	C1	100μF	C7	10μF	D3	Si
								D4	Si

You've probably seen a roulette wheel, or "wheel of fortune" type game in operation. You know how it works ... players try to guess where the wheel stops and they win if they guess right. we couldn't find room in this kit for the real thing, but we've included an electronic version!

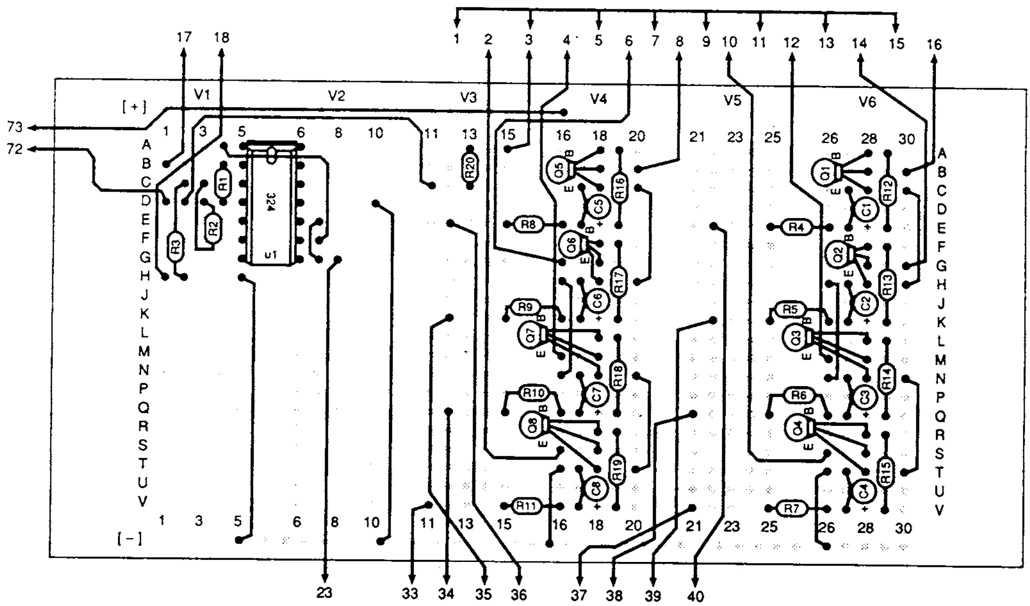
Using this electronic "wheel of fortune" is supersimple. Turn power ON and press **S1** down. You'll notice the lower half of the **LED display** lights up and seem to "spin" around. (It's not really spinning, of course - the different segments are just rapidly blinking on and off, one after the other.) During the "spinning" you'll hear a funny sound from the **speaker**. After a few moments both the "spinning" and sound slows down. Eventually, it stops with just one segment lit and a steady sound coming from the **speaker**.

You'll notice a couple of interesting things about this circuit. Each segment of the display has its own sound. And the speed at which the display "spins" depends upon the **control volume**. Try moving the **control volume** while the "wheel" is "spinning" ... notice how you can make it slow down or speed up.

You can use this project as a game by guessing which segment will be lit when the "wheel" finally stops "spinning." Or you can try to make the "wheel" stop at a certain segment by adjusting the **control volume** while it is still "spinning."



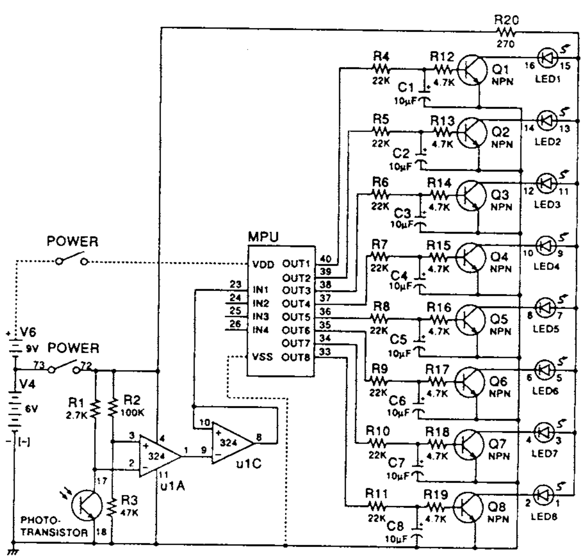
Project 470. Illumination Controlled by PHOTO-TRANSISTOR (1)



U1	324	Q8	NPN	R8	22KΩ	R16	4.7KΩ	C1	10μF
Q1	NPN	R1	2.7KΩ	R9	22KΩ	R17	4.7KΩ	C2	10μF
Q2	NPN	R2	100KΩ	R10	22KΩ	R18	4.7KΩ	C3	10μF
Q3	NPN	R3	47KΩ	R11	22KΩ	R19	4.7KΩ	C4	10μF
Q4	NPN	R4	22KΩ	R12	4.7KΩ	R20	270Ω	C5	10μF
Q5	NPN	R5	22KΩ	R13	4.7KΩ			C6	10μF
Q6	NPN	R6	22KΩ	R14	4.7KΩ			C7	10μF
Q7	NPN	R7	22KΩ	R15	4.7KΩ			C8	10μF

In this Project, we will make a program which starts illuminations on the LEDs by adjusting your finger position over the PHOTO-TRANSISTOR.

When the program is running, cover the PHOTO-TRANSISTOR with your finger tips for a moment and remove them. As you have waved a Checker Flag, it starts blinking of LEDs one by one from L1 thru L8.



Description:

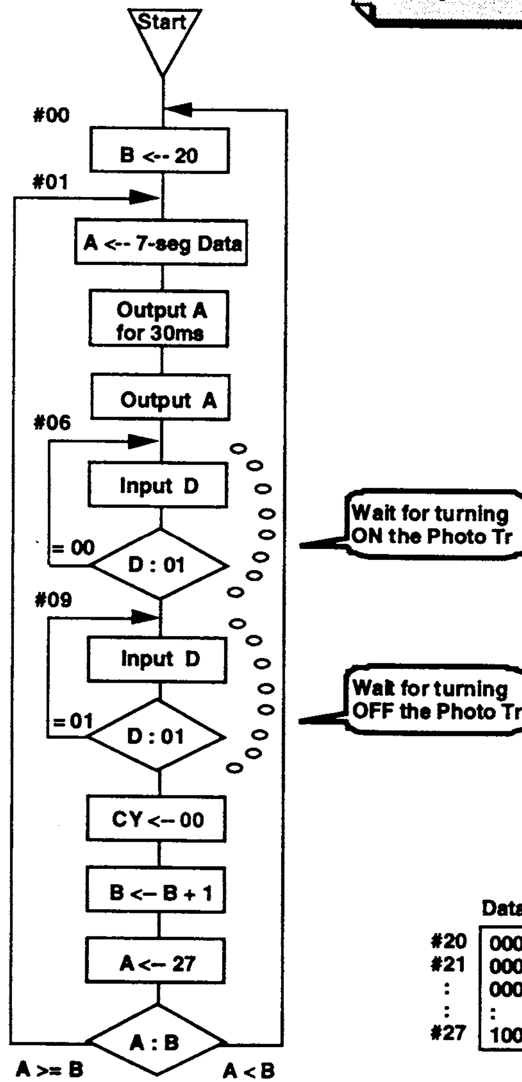
When you look at the flow-chart, you can get it quite easily as its flow of control is relatively simple. Now, let's look into the program.

The key to understanding this program is how to start illumination by the movement of your finger tips over the PHOTO-TRANSISTOR. You can see one solution to that concern; the program code from #06 thru #08 and #09 thru #0B are what you have seen in several previous Projects. The code from #06 thru #08 captures the input data 01, that is, this short loop processing is over with your action of covering the PHOTO-TRANSISTOR with your fingers. Then comes the 2nd gate, the #09 thru #0B, where the program waits for you to stop covering the PHOTO-TRANSISTOR with your fingers.

Project-470

Program:

```
00  MOV  B,#20H   ; Get data table addr
01  MOV  A,@B     ; Get lighting data
02  OUT  A        ; Light LEDs
03  TM1  #03H     ;   for 30ms
04  MOV  A,#00H   ;
05  OUT  A        ; Turn OFF LEDs
06  IN   D        ; Get input
07  AND  D,#01H   ;
08  JZ   L06H     ; Wait for turning ON PTr
09  IN   D        ; Get input
0A  AND  D,#01H   ;
0B  JNZ  L09H     ; Wait for turning OFF PTr
0C  CLC           ; Clear carry-flag
0D  INC  B        ; Increase table addr
0E  MOV  A,#27H   ;
0F  CMP  A,B      ;
10  JC   L00H     ; If end of table, return
11  JMP  L01H     ; In the mid of table; continue
12  NOP           ;
:
20  HEX  #01H     ; (0000 0001)
21  HEX  #02H     ; (0000 0010)
22  HEX  #04H     ; (0000 0100)
23  HEX  #08H     ; (0000 1000)
24  HEX  #10H     ; (0001 0000)
25  HEX  #20H     ; (0010 0000)
26  HEX  #40H     ; (0100 0000)
27  HEX  #80H     ; (1000 0000)
```



Data Table

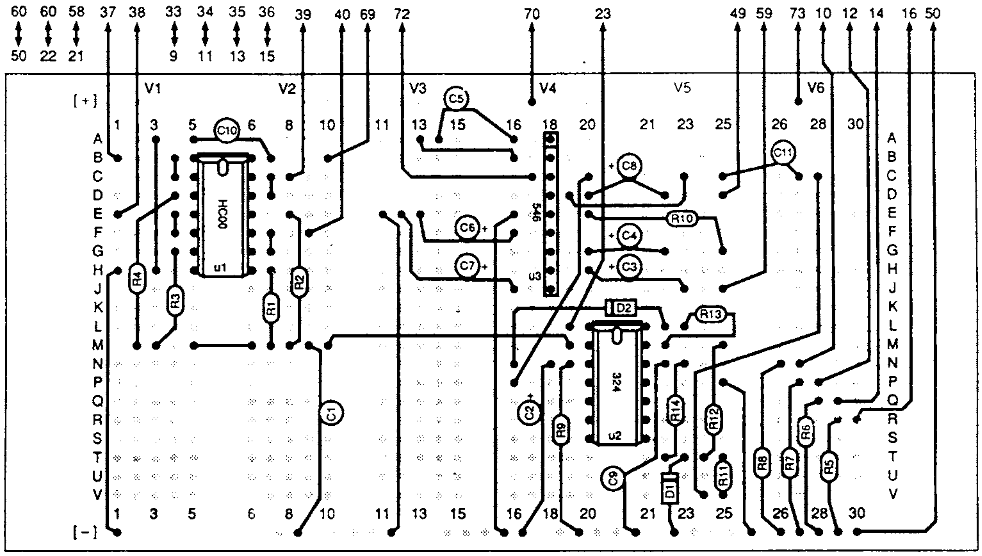
#20	0000 0001
#21	0000 0010
:	0000 0100
:	:
#27	1000 0000

Note: In the above program, NOP instructions in the lines #13 thru #1F are omitted. You are expected to enter these NOPs if you run the program.

Now, let's look the whole processing on the flow-chart. The program first gets the initial count value 20H at the #00. It then goes into a loop processing which includes the codes of sensing your finger action described above. This loop processing continue until its count becomes 28H. Since the initial count value is 20H, this loop repeats 8 times per one cycle. And this is also because that the output data table starting at address 20H contains 8 patterns of illumination.

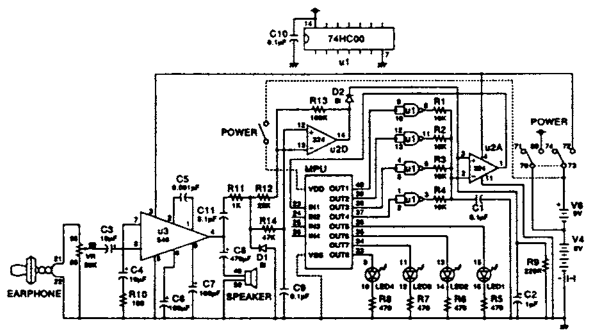
The pattern of illumination is simple; it lights LEDs one by one from L1 to L8 repeatedly. You can change the data table values starting at #20 and get your own pattern.

Project 484. Audio Level Meter With Peak-Holding Capability



U1	74HC00	R4	10KΩ	R10	100Ω	C2	1µF	C8	470µF
U2	324	R5	470Ω	R11	1KΩ	C3	10µF	C9	0.1µF
U3	546	R6	470Ω	R12	22KΩ	C4	10µF	C10	0.1µF
R1	10KΩ	R7	470Ω	R13	100KΩ	C5	0.001µF	C11	0.1µF
R2	10KΩ	R8	470Ω	R14	47KΩ	C6	100µF	D1	Si
R3	10KΩ	R9	220KΩ	C1	0.1µF	C7	100µF	D2	Si

In this Project, the Audio Level Meter introduced by the Project-483 is enhanced so that it can capture the maximum input level and hold the data for a few second. This capability is convenient for checking the maximum sound signal level for example. You can change the holding period by altering the data value in register F.



Description:

The program works rather in a complicated way. The flow-chart will help you follow the move of program control. From the chart, you can find here again the inner loop processing; it starts from the #06 and ends at the #16. Let's trace the control of program by watching both the chart and the coding.

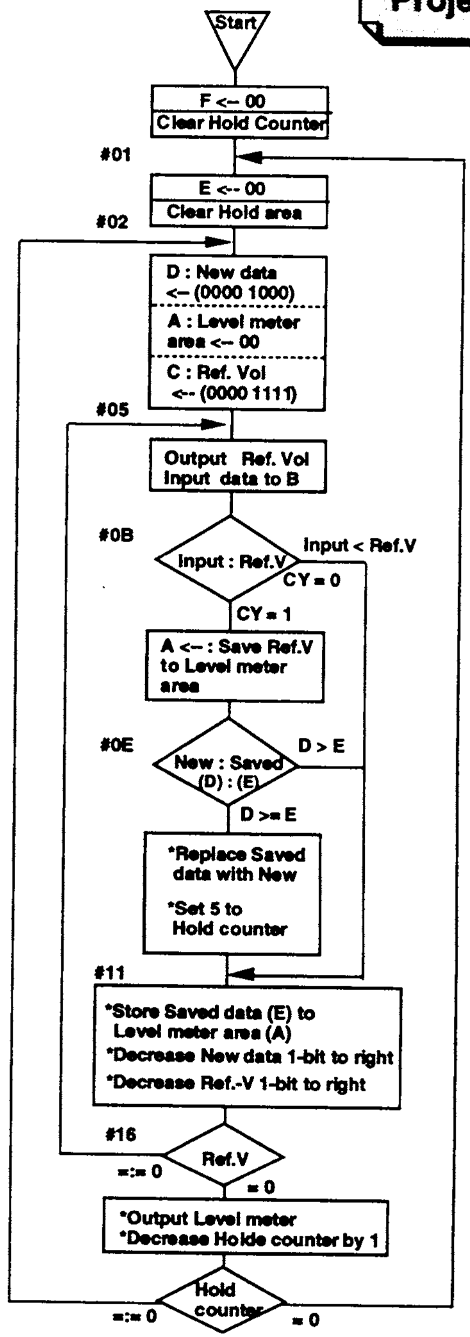
From the #06 thru #09, the program outputs the reference voltage data, wait for 40ms, and tries to get input data which is actually a single bit data from the comparator. If the input voltage to the IC2A exceeds the reference voltage, the comparator outputs high level of signal, and it is fed to the input port IN1 of the MPU. This is a 1-bit data from the comparator. If no data bit comes in, the control of program goes to the #11 at #0B. If 1-bit data comes in, the control of program goes forward to the #0C and the then-current reference voltage data in the register C is saved to accumulator A.

Then it comes to the #0D, where the Incoming data (register D) is compared with the Saved data (register E). If the Incoming data is less than the Saved data, the control of program goes to the #11. However, if the incoming data is greater than or equal to the Saved data, the program replaces the Saved data with the Incoming data at the #0F. After that, the Hold counter is set to 5 at the #10. And note that these two steps (#0F and #10) are performed only once within one Hold counter cycle! Let's prove it.

Project-484

Program:

```
00  MOV  F,#00H   ; F = Hold counter
01  MOV  E,#00H   ; E = Hold-save area
02  MOV  D,#08H   ; D = Incoming data
03  MOV  A,#00H   ; A = Level metetr data
04  MOV  C,#0FH   ; C = Reference voltage data
05  XOR  C,#0FH   ; Get lower 4-bit
06  OUT  C        ; Output reference voltage
07  XOR  C,#0FH   ; Restore C
08  TM1  #04H     ;   and wait for 40ms
09  IN   B        ; Get input from comparator
0A  ROR  B        ; Move 1-bit data to carry FF
0B  JNC  L11H     ; If input-V < ref-V; to #11
0C  OR   A,C      ; Input-V > ref-V; save ref-V
0D  CMP  D,E      ; (Incoming)-(Saved)
0E  JC   L11      ; If incoming < saved; to #11
0F  MOV  E,D      ; Replace Saved with Incoming
10  MOV  F,#05H   ; Set hold counter to 5H
11  OR   A,E      ; Store Saved to M meter
12  CLC           ; Clear carry FF
13  ROR  D        ; Decrease Incoming voltage
14  CLC           ; Clear carry FF
15  ROR  C        ; Decrease ref-V
16  JNZ  L05H     ; If ref-V is not zero, to #05
17  CLC           ; If ref-V is zero; clear FF
18  ROL  A        ; Move lower 4-bit to left
19  ROL  A        ;
1A  ROL  A        ;
1B  ROL  A        ;
1C  OUT  A        ; Output level meter data
1D  TM1  #05H     ;   and wait for 50ms
1E  DEC  F        ; Decrease Hold-counter by 1
1F  JNZ  L02H     ; If not zero, go to 02#
20  JMP  L01H     ; If zero, go to #01
```



Let's assume that the program is in the first pass of the first Hold counter cycle; therefore, the Incoming data (register D) is 08H (0000 1000) while the Saved data (register E) is 00H (0000 0000). This situation is possible only when the input voltage exceeds the reference voltage. Now, the control of program comes in the #0F, where the Saved data (register E) is replaced with the Incoming Data (register D). Because of this replacement of the data, these two steps can never be performed within the same Hold counter cycle. If the control returns to the same code at the #0D and #0E, the Incoming Data (register D) is ALWAYS smaller than the Saved data (register E) since the Incoming data is decreased by 1-bit shift to right at the #13 while the Saved data does not change. Remember that with these few steps, the largest Incoming data is captured within the Hold counter cycle.

Let's continue to trace the following codes. At the #11, the program stores the Saved data to the accumulator A as a level meter data, then it decreases the Incoming data at #13 and reference voltage data at #15 by shifting each register 1-bit to right.

The above inner loop processing continues until the reference voltage data becomes zero at the #16. Once this one cycle is over, the Level Meter data in the accumulator A is output and the Hold counter is decreased by 1. The program continues to perform these above processing until the Hold counter becomes zero.