

Electronics Lab Manual

Volume 2

K. A. NAVAS, M Tech
Asst. Professor, ECE Dept.
College of Engineering Trivandrum
Thiruvananthapuram-695016

kanavas@rediffmail.com

Rajath Publishers

28/450-A, Club Road, Girinagar South, Kadavanthra, Kochi-682020

Electronics Lab Manual Volume 2
Fourth edition

Copyright ©2009 Rajath publishers and the author jointly

This book is sold subjected to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without publisher's prior written consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser and without limiting the rights under copyright reserved above, no part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), without prior permission of the copyright owner.

Published by
Rajath Publishers
28/450-A, Club Road, Girinagar South
Kadavanthra, Kochi-682020
Phone:0484-2313911
e-mail:rajathpbs@yahoo.com

Price Rs. 180.00

Type set in \LaTeX
Printed at Pioneer offset, Ravipuram, Kochi-15

Contents

1	Linear/Analog IC Lab	11
1.1	Familiarisation of op-amp	11
1.2	Measurements of op-amp parameters	12
1.3	Basic operational amplifier circuits	15
1.4	Schmitt trigger	25
1.5	Astable multivibrator	28
1.6	Monostable multivibrator	31
1.7	Half wave rectifier using op-amp	33
1.8	Full wave rectifier using op-amp	34
1.9	Triangular wave generator	34
1.10	Sawtooth wave generator using op-amp	37
1.11	RC phase shift oscillator	38
1.12	Wien bridge oscillator	39
1.13	Wien bridge oscillator with amplitude stabilisation	40
1.14	Window detector using op-amp	41
1.15	Window detector using comparator IC 311	42
1.16	Peak detector	43
1.17	Instrumentation amplifier	44
1.18	Voltage limiters	45
1.19	Clipping circuits	47
1.20	Clamping circuits	50
1.21	Filter using gyrator circuit	52
1.22	Three terminal voltage regulators	54
1.23	Low voltage regulator using 723 IC	55
1.24	High voltage regulator using 723 IC	58
1.25	LVR with current foldback	61
1.26	Counter ramp ADC	63
1.27	Study of ADC and DAC ICs	64
1.28	Study of waveform generator IC 8038	65

1.29	Sawtooth generator using 555	67
1.30	Examination questions	68
2	Analog and Digital Communication Engineering Lab	70
2.1	Constant-k LPF and HPF	70
2.2	m-derived LPF and HPF	74
2.3	AM generation	77
2.4	AM using multiplier IC AD534/AD633	83
2.5	AM detection	85
2.6	Mixer	86
2.7	IF tuned amplifier	88
2.8	FM using 566 IC	91
2.9	FM using 555 IC	94
2.10	Study of 565 PLL	96
2.11	FM generation and demodulation using 565	99
2.12	Frequency multiplier using 565	100
2.13	Study of CMOS PLL 4046	102
2.14	FM generation and demodulation using 4046	104
2.15	Frequency multiplier using 4046	105
2.16	PAM modulator and demodulator	106
2.17	PWM using 555	109
2.18	PWM using op-amp	110
2.19	PPM using 555	112
2.20	PCM	113
2.21	Delta modulation	115
2.22	TDM	117
2.23	BASK	119
2.24	BPSK	121
2.25	BFSK	123
2.26	Pre-emphasis de-emphasis	126
2.27	Error checking and correcting codes	127
2.28	Pseudo-random binary sequence generator	131
2.29	First order low pass filter	133
2.30	Second order low pass filter	135
2.31	First order high pass filter	136
2.32	Second order high pass filter	138
2.33	Band pass filter	139
2.34	Notch filter	141
2.35	Universal active filter	142
2.36	All pass filters	143

3	Microprocessor lab-8085	146
3.1	Familiarisation with microprocessor kit	146
3.2	Block transfer of data	148
3.3	Overlapping block transfer of data	149
3.4	Exchange of data between memory blocks	149
3.5	Number of occurrence	150
3.6	Largest/smallest in an array	151
3.7	Sorting in ascending/descending order	152
3.8	BCD to binary conversion	154
3.9	Binary to BCD conversion	155
3.10	Hex to ASCII conversion	156
3.11	ASCII to Hex conversion	157
3.12	Binary addition	158
3.13	Multi-byte addition	159
3.14	BCD addition	160
3.15	BCD subtraction	161
3.16	Binary multiplication	162
3.17	Binary division	164
3.18	Square root of a number	165
3.19	Factorial of a number	166
3.20	Stepper motor interface	167
3.21	Rolling display using 8279	169
3.22	Display character from keyboard	170
3.23	Generation of waveforms	171
3.24	Waveform generation using DAC	174
3.25	ADC interface	175
4	Microprocessor lab-8086 (using MP kit)	187
4.1	Familiarisation of MP trainer kit	187
4.2	Block move of data	189
4.3	Addition of 16-bit binary numbers	190
4.4	Addition of 32-bit binary numbers	191
4.5	Subtraction of 16-bit binary numbers	192
4.6	Subtraction of 32-bit binary numbers	193
4.7	Sum of array elements	194
4.8	Multibyte addition	195
4.9	BCD addition/subtraction	196
4.10	Multiplication of 16-bit binary numbers	197
4.11	Division of binary numbers	198
4.12	BCD to binary conversion	199

4.13	Binary to BCD conversion	200
4.14	ASCII to Hex conversion	201
4.15	Largest/smallest in the array	202
4.16	Sorting in ascending/descending order	203
4.17	Square root of a number	204
4.18	Finding LCM and GCD	205
4.19	Prime number generation	207
4.20	Fibonacci series	208
4.21	Multiplication of 32-bit binary numbers	209
4.22	Stepper motor interface	210
4.23	Rolling display	212
4.24	Waveform generation using DAC	213
4.25	ADC using interface card	215
5	Microprocessor Lab-8086 (using MASM/TASM)	219
5.1	Familiarisation with MASM/TASM	219
5.2	Familiarisation of DOS and BIOS interrupts	221
5.3	Display of strings	223
5.4	Addition of two single digit numbers	224
5.5	Addition of two two-digit numbers	227
5.6	Addition of a set of numbers	229
5.7	Reversing a string	230
5.8	Case conversion	232
5.9	Concatenation of strings	233
5.10	Checking for palindrome	234
5.11	Fibonacci sequence	236
5.12	Timer	238
5.13	Drawing line	239
5.14	Drawing triangle	240
6	Microprocessor lab-8086 (using DEBUG)	243
6.1	DEBUG Program	243
7	Microcontroller lab-8951 (using MC kit)	249
7.1	Familiarisation with MC trainer kit	249
7.2	Block move of data	251
7.3	Number of occurrence	252
7.4	Addition of two 8-bit binary numbers	253
7.5	Sum of array elements	253
7.6	BCD addition	255

7.7	Division of 8-bit numbers	256
7.8	Addition of multi-byte numbers	256
7.9	Finding largest	258
7.10	Sorting in ascending order	259
7.11	Factorial of a number	260
7.12	Binary to BCD conversion	261
7.13	BCD to binary conversion	262
7.14	Binary to gray conversion	264
7.15	Hex to ASCII hex code conversion	264
7.16	ASCII to hex code conversion	265
7.17	Square of a number	267
7.18	Square root of a number	267
7.19	Generation of square wave	269
7.20	Square wave using timer	270
7.21	Relay activation using optocoupler	271
7.22	Examination questions	273
8	Microwave lab	275
8.1	Familiarisation of microwave components	275
8.2	Gunn diode characteristics	280
8.3	Reflex klystron characteristics	281
8.4	Frequency and wavelength measurements	283
8.5	VSWR measurement	285
8.6	Magic tee measurement	287
8.7	Directional coupler characteristics	289
8.8	Antenna measurements	291
9	MATLAB	298
9.1	SIMULATION LAB	298
9.1.1	Introduction to MATLAB	298
9.1.2	Generation of simple signals	300
9.1.3	Generation of AM signal	303
9.1.4	Generation of FM signal	304
9.1.5	Generation of PWM signal	305
9.1.6	Examination questions	306
9.2	SIGNAL PROCESSING LAB	308
9.2.1	Impulse response of an LTI system	308
9.2.2	Impulse response from transfer function	309
9.2.3	Bode plot of a system	311
9.2.4	Linear convolution	312

9.2.5	Deconvolution	315
9.2.6	Circular convolution	316
9.2.7	Linear convolution using circular convolution	317
9.2.8	DFT and IDFT	318
9.2.9	DFT and IDFT using matlab function	320
9.2.10	Circular convolution using DFT	324
9.2.11	Linear convolution using DFT	325
9.2.12	Moving average filter	326
9.2.13	FIR filter using rectangular window	327
9.2.14	FIR filter using Hamming window	329
9.2.15	FIR filter using Blackman window	331
9.2.16	FIR filter using Kaiser window	332
9.2.17	FIR bandpass and highpass filters using windows	334
9.2.18	FIR highpass filter	336
9.2.19	IIR lowpass filter	338
9.2.20	IIR filter design using MATLAB functions	349
9.2.21	IIR filter with input frequencies in Hz	353
9.2.22	Introduction to DSP development system	356
9.2.23	Creating and building projects in CCS	359
9.2.24	Generation of sine wave using CCS	362
9.2.25	FIR filtering using CCS	363
9.2.26	IIR filtering using CCS	364
9.2.27	Examination questions	365

Chapter 1

MICROCONTROLLER LAB-8951 (USING MC KIT)

1.1 Familiarisation with MC trainer kit

Aim To familiarize with 8951 microcontroller trainer kit having in-built assembler.

8086 microprocessor MICRO-51 LC trainer kit manufactured by Vi Microsystems Private Ltd., Chennai has in built assembler and disassembler.

ASSEMBLER COMMANDS

#A Line assembler. This command is used to enter the mnemonics of 8951 and it gives the opcode for the mnemonics.

Syntax: #A<CR> Enter user RAM address and program.

#U: Unassembler. It gives the mnemonics and corresponding opcodes in the given address.

Syntax: #U<CR>

#SP, SD Substitute byte. This command is used to examine the contents of selected memory locations and modify them.

Syntax: #SP<addr><CR> for program memory

#SD<addr><CR> for data memory

#R To examine and modify the register contents of the CPU.

Syntax: #R<CR>

#GO Go and execute. The Go command is used to run the program.

Syntax: #GO<addr><CR>

Go with break point Syntax: #GO<start add ><end add><CR>

#TR Trace command. This command helps the user to execute program in steps, i.e., instruction by instruction. This command will be very helpful while debugging programs.

Syntax: #TR <addr><CR>

#FP, FD This command permits a block of RAM memory to be filled with desired data type.

Syntax: #FB <start addr><end addr><byte data><CR> for program memory

#FD <start addr><end addr><byte data><CR> for data memory

#MP, MD Byte move. This command moves the content at a specified block of memory to another block whose starting address is specified in RAM.

Syntax: #MP <start addr><end addr ><dest addr> for program memory

#MD <start addr><end addr ><dest addr> for data memory

#IP, ID This command inserts the specified bytes in the desired memory location.

Syntax: #IP <insert addr ><program end addr ><No. of bytes><CR> for program memory

#ID <insert addr ><program end addr ><No. of bytes><CR> for data memory

#EP, ED Delete. This command deletes blocks of bytes and words respectively from memory.

Syntax: #EP <start addr ><end addr><program end addr><CR> for program memory

#ED <start addr ><end addr><program end addr><CR> for data memory

#IN Input byte. This command inputs bytes from the desired port.

Syntax: IN <port addr><CR>

#O Output byte. This command outputs data to the desired port.

Syntax: O <port addr ><data><CR>

#IR This command is used to view the bytes in the internal RAM locations.

Syntax: IR <Addr ><CR>

HARDWARE SPECIFICATIONS

Monitor EPROM :	0000:3FFFH & FFFF
System RAM :	4000:BFFFH
User RAM :	4100:BFFFH

Procedure to enter a program

1. Switch on the 8086 kit, Type A and enter.
2. Type in the starting address and press enter.
3. Enter the mnemonics.

Procedure to enter data

1. Press INT key.
2. Type #SP address and press enter key.
3. Update the locations with desired bytes.

Procedure to execute the program

1. Press '.' and enter or reset key.
2. Type #GO starting address of code and press enter key.
3. Press '.' and enter.

1.2 Block move of data

Aim To write a program to move a block of memory from one location to another. Five numbers stored in memory locations starting from 4500H must be moved to 4600H onwards.

Algorithm

1. Set byte counter.
2. Get the number from the source and copy it in destination.
3. Decrement counter and repeat the above step until the byte counter is reset.

Program

```

4100: 79 0A          MOV R1,05H           ; Length of block in R1
4102: 90 45 00       MOV DPTR,#4500H      ; First address in DPTR
4105: E0           NEXT:  MOVX A,@DPTR         ; First no. in Acc.
4106: 75 83 46       MOV DPH,46           ; 4600H in DPTR
4109: F0           MOVX @DPTR,A         ; Number moved to 4600H
410A: 75 83 45       MOV DPH,45H          ; 4500H in DPTR
410D: A3           INC DPTR              ; Points next location
410E: D9 F5         DJNZ R1,NEXT (4105H) ; Continue till counter is reset
4110: 80 FE          HLT:  SJMP HLT (4110H) ; Halt

```

Procedure

1. Enter the program in memory locations starting from 4100H.
2. Enter five numbers in memory locations starting from 4500H.
3. Execute the program and verify the result in memory locations starting from 4600H.

Test data

Input: 4500: 01 02 03 04 05

Output: 4600: 01 02 03 04 05

1.6 BCD addition

Aim To write a program to add two BCD numbers.

Description One 8-bit packed BCD number is available in the memory location 4300H and another number is available in the memory location 4301H. Result is available in the location 4302H and carry in 4303H.

Algorithm

1. Get the BCD numbers from the memory locations.
2. Add the BCD numbers with the help of DAA.
3. Store the sum and carry in the memory.

Program

```

4100: 90 43 00      MOV DPTR,#4300H    ; Address of first BCD number
4103: 79 00        MOV R1,#00         ; Carry is reset
4105: E0           MOVX A,@DPTR        ; First number in A
4106: A3           INC DPTR            ; Pointing to second number
4107: F8           MOV R0,A           ; First number in R0
4108: E0           MOVX A,@DPTR        ; Second number in A
4109: 28           ADD A,R0           ; Add both numbers
410A: D4           DA A             ; Convert to BCD
410B: 50 01        JNC NOC (410E)     ; If no carry, skip
410D: 09           INC R1             ; Else increment carry
410E: A3      NOC:  INC DPTR
410F: F0           MOVX @DPTR,A        ; Result in destination address
4110: E9           MOV A,R1           ; Carry in A
4111: A3           INC DPTR
4112: F0           MOVX @DPTR,A        ; Store carry
4113: 80 FE      HLT:  SJMP HLT (4113)    ; Halt

```

Procedure

1. Enter the program in memory locations starting from 4100H.
2. Enter data in memory locations 4300H and 4301H.
3. Execute the program and verify the sum in memory location 4302H and carry in 4303H.

Test data

Input: 4300: 75 4301: 65 Output: 4302: 40 4303: 01

1.14 Binary to gray conversion

Aim To write a program to convert an 8-bit binary number to equivalent gray code.

Algorithm

1. The MSB in the gray code is the same as the corresponding bit in a binary number.
2. Going from left to right, add each adjacent pair of binary digits to get the next gray code digit. Disregard carries.

Program

```

4100: 90 42 00          MOV DPTR,#4200H    ; Address of the number
4103: E0              MOVX A,@DPTR       ; Keep number in A
4104: C3              CLR C              ; Clear carry for rotation
4105: 13              RRC A                ; Rotate through carry
4106: F9              MOV R1,A           ; Rotated number in R1
4107: E0              MOVX A,@DPTR       ; Number again in A
4108: 69              XRL A,R1          ; XOR rotated and original numbers
4109: A3              INC DPTR
410A: F0              MOVX @DPTR,A       ; Store result in 4201
410B: 80 FE          HLT: SJMP HLT(410B)

```

Procedure

1. Enter the program in memory starting from 4100H.
2. Enter the number in memory location 4200H.
3. Execute the program and verify the result in 4201H.

Test data

Input:	Output
4200: 90	4201: D8

1.15 Hex to ASCII hex code conversion

Aim To write an assembly language program to convert an Hexadecimal number to its ASCII Hex Code equivalent. Hexadecimal number is in the memory location 4200H and ASCII number must be stored in the memory location 4201H.

Theory The American Standard Code for Information Interchange (known as ASCII) is a 7-bit code with 128 combinations and each combination is assigned to a letter, a decimal number,

1.17 Square of a number

Aim To find the square of a given number.

Theory The square of a number can be calculated by adding consecutive odd numbers starting from 1. In order to find the square of 3, add three odd numbers starting from 1. $1 + 3 + 5 = 9$. To find the square of 5, $1 + 3 + 5 + 7 + 9 = 25$.

Algorithm

1. Set the number whose square is to be calculated as the counter.
2. Add odd numbers starting from zero until counter is zero.

Program

```

4100: 90 42 00          MOV DPTR,#4200H
4103: E0              MOVX A,@DPTR          ; Number in Acc.
4104: 60 0A          JZ RESULT (4110)     ; If number is zero, store it
4106: FA              MOV R2,A             ; Set number as counter
4107: 79 01          MOV R1,#01          ; First odd number
4109: 74 00          MOV A,#00           ;
410B: 29          LOOP: ADD A,R1     ; Add progressively
410C: 09              INC R1              ; Next odd number
410D: 09              INC R1
411E: DA FB          DJNZ R2, LOOP (410B) ; Continue till counter = 0
4110: A3          RESULT: INC DPTR ; Store result
4111: F0              MOVX @DPTR,A
4112: 80 FE          HLT: SJMP HLT (4112)

```

Procedure

1. Enter the program in memory starting from 4100H.
2. Enter the number in memory location 4200H.
3. Execute the program and verify the result in 4201H.

Test data

Input:	Output	
4200: 05	4201: 19	$19H = 25_{10}$

1.18 Square root of a number

Aim To calculate the square root of a given number in the address 4200H and store it in 4201H if it is a perfect square. If not, store EE in 4201H to prompt error message.

Procedure

1. Set up the circuit on a bread board and enter the program in memory starting from 4100H.
2. Execute the program and connect the P1.0 pin of microcontroller to optocoupler.
3. Observe that the bulb blinks intermittently. Change the delay and observe the change in timing.

1.22 Examination questions

1. Write an assembly language program to light two sets of LEDs alternatively.
2. Write an assembly language program to check a string whether it is a palindrome.
3. 20 numbers are stored in location 4400H onwards. Arrange the numbers in the reverse order.
4. Write an assembly language program to exchange a block of data.
5. Write a program to subtract two multi-byte BCD numbers.
6. Write a program to subtract two 4-digit decimal numbers.
7. Write a program to sort negative and positive numbers.
8. Write an ALP to obtain $x^2 + y^2$, where x and y are two nibbles.
9. Write an ALP to prove Pythagoras theorem.
10. Write a program to separate the addresses in an array where odd numbers are stored (Only the lower bytes of addresses).
11. A set of non-zero bytes are stored in locations 4100H onwards. The last byte in the data set is indicated by the entry 00H. Write a program to arrange it in ascending order.
12. An array of numbers $a_1, a_2, a_3, \dots, a_{10}$ are stored in location starting from XX00. Rearrange the array as shown below: $a_2, a_1, a_4, a_3, a_6, a_5, a_8, a_7, a_{10}, a_9$. Assume values for numbers.
13. An array of numbers are given. Write a program to shift the array to memory locations starting from 4200H if the sum of array elements is odd. If the sum of array elements is even, shift the array to 4300H.
14. Write a program to add two 3×3 matrices.
15. Write an ALP (assembly Language program) to find the transpose of a 3×3 matrix.
16. Write an ALP to multiply a 3×2 matrix by another 2×3 matrix.
17. Write an ALP to generate a square wave of frequency 1 kHz and duty cycle 40%.
18. Generate a triangular waveform and display it on the C.R.O.
19. Write an ALP to rotate a stepper motor clock wise for 360° degree.
20. Write an ALP to rotate a stepper motor counter clockwise for 1.5 rotations.
21. An array of numbers is given. If the sum of the numbers is odd, rotate the stepper motor in clockwise direction. If it is even, rotate in counter clockwise direction.

For Copies,

Rajath Publishers

28/450-A, Club Road, Girinagar South

Kadavanthra, Kochi-682020

Phone: 0484-2313911

e-mail:rajathpbs@yahoo.com