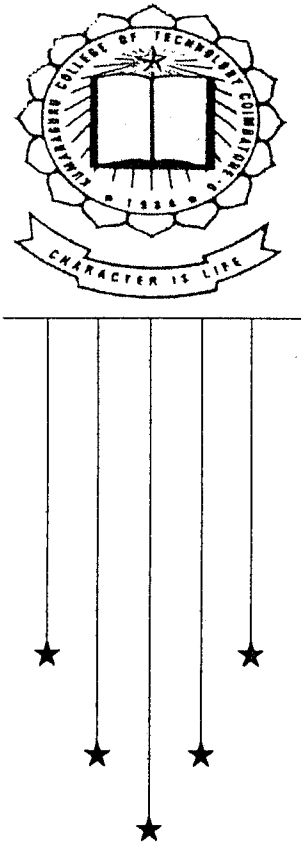


# 80C320 CPU DEBUGGER



April 2001

## PROJECT REPORT

*SUBMITTED BY*

AARTHI VENKATESHAN

KARTHICK . R

MARY SUSANA GEORGE

RAJESH KUMAR . N

SENTHIL KUMAR. R

*GUIDED BY*

Mr. S. GOVINDARAJU, M.E., MISTE.

Assistant Professor, ECE Dept.

IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS  
FOR THE AWARD OF THE DEGREE OF  
**BACHELOR OF ENGINEERING IN**  
**ELECTRONICS & COMMUNICATION ENGINEERING**  
OF THE BHARATHIAR UNIVERSITY  
COIMBATORE

*Department of Electronics & Communication Engineering*

**KUMARAGURU COLLEGE OF TECHNOLOGY**

Coimbatore-641 006.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING  
**KUMARAGURU COLLEGE OF TECHNOLOGY**  
COIMBATORE-641006.  
(Affiliated to Bharathiyar University)

**CERTIFICATE**

*This is to certify that the Project Report entitled  
80C320 CPU DEBUGGER  
has been submitted by*


---

*in partial fulfillment for the award of the degree of Bachelor of Engineering in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

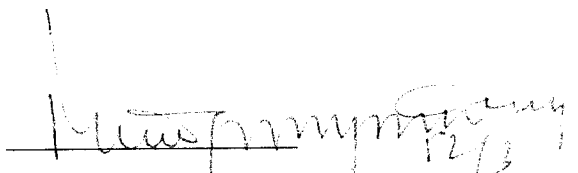
*of the Bharathiyar University, Coimbatore – 641 006  
during the academic year 2000-2001*

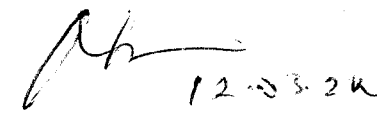
  
\_\_\_\_\_  
Faculty Guide

  
\_\_\_\_\_  
Head of the Department

*Certified that the Candidate was examined by us in the project work viva-voce  
Examination held on 12/3/2001 and the university Register No. was*

---

  
\_\_\_\_\_  
Internal Examiner

  
\_\_\_\_\_  
External Examiner

# **PREMIER**

## **CERTIFICATE**

This is to certify that the following students of KUMARAGURU COLLEGE OF TECHNOLOGY, Coimbatore, of branch Electronics & Communication Engineering

Aarthi Venkateshan  
R Karthick  
R Senthil Kumar  
N Rajeshkumar  
Mary Susana George

had undertaken their project entitled "80C320 CPU DEBUGGER", from June 2000 to March 2001 at our Industry and have successfully completed it.

Their performance during that period was found to be good. We wish them all success.

Place : Coimbatore

Date : March 8, 2001

Industry Seal

Premier Polytronics Limited  
TRICHY ROAD  
SINGANAILLUR POST  
COIMBATORE-641 005.

  
**V SRINIVASAN**  
Manager - R&D

  
**K K VENKATARAMAN**  
Vice President - R&D

**premier polytronics ltd.**

304, Trichy Road • Singanailur • Coimbatore - 641 005 • India.

Tel : 0422 - 573548 • Fax : 0422 - 573651 • www.premier-1.com • Email : mail@premier-1.com

Important : All information contained herein is confidential and not be disclosed to parties not specified herein



***“ The secret of success is to put your best forward, nurtured at every step by the great men who give you the sense of direction”***

We thank our Principal **Dr.K.K. Padmanabhan** B.Sc(Engg), M.Tech, Ph.d and the Management for the facilities provided in the college to accomplish this project.

We place on record our deep sense of gratitude to our Head of the Department **Prof. Muthuraman Ramasamy**, M.E, MISTE, MIE, C.ENG(I), MBMESI, MIEEEE(USA) for his constant encouragement and support throughout the project.

We are greatly indebted to our guide **Prof. S. Govindaraju**, M.E, M.I.S.T.E for his invaluable advice and gentle reminders, that really motivated us through many a tough encounter, only to help us reach our goal.

We would be failing in our duty if we do not express our bountiful thanks to **Mr. S. N. Ramachandran**, Director, Premier Polytronics Limited who opened the doors of opportunity and paved the way for our project.

We profoundly thank the Vice President, Research and Development **Mr.K.K.Venkataraman** and Manager, Research and Development **Mr. V. Srinivasan** who helped us carry out this project meticulously within the stipulated time.

We profusely thank **Mr. E. Saranavanakumar** for fine-tuning our technical potential and helping us bridge our theoretical and practical knowledge, in spite of his busy schedule.

We are grateful to **Miss. Amirtham** for giving us hours of beneficial training during this project.

We express our heartfelt thanks to **Mr. Vijaykumar**, **Mrs. Geethanjali** and **Mrs. Rajeswari** who helped us in the completion of our project.

We thank all the staff of Premier Polytronics Limited and the Electronics and Communication Engineering Department who was a source of strength in this project.

In the world of Textile Industry today, Automation is the key to tomorrow's technology. This Evergreen industry with all the blooming ventures has many an advanced machines. The PREMIER HFT TESTER developed by Premier Polytronics Limited; Coimbatore is used to test the various fibers of cotton. A method of quality assurance is extended to reduce the time span and fatigue of workers, as time and energy are two valuable resources of mankind.

The firmware Project titled "**80C320 CPU DEBUGGER**" has two major sections namely the Hardware and the Software divisions. The aim of this project is to test the Controller board of the HFT tester and trace the defects if any. The status of the CPU board is indicated by means of LED's in the front panel of the jig. The 80C320 controller used is the brain of this machine; its meticulous design includes a number of peripherals of which every parameter needs to be tested for the proper functionality of the machine.

On the Hardware front, three printed Circuit Boards have been fabricated besides the Front Panel wherein the control switches and LED's indicate the status of the controller board. On the Software front, coding has been written in the assembly language of 80C320.

The Test Jig is a user-friendly interface that has found real-time application in the company as the versatile CPU board is used in many of their products.

# ***INDEX***

Page No.

Acknowledgement

Synopsis

Contents

<b>1. Introduction</b>	<b>1</b>
1.1 Company Profile	
1.2 HFT 9000 Machine	
1.3 Project Description	
<b>2. Hardware</b>	<b>6</b>
2.1 Test Jig Overview	
2.2 An insight into the Test Jig	
2.2.1 Power Supply PCB	
2.2.2 Interface PCB	
2.2.3 Display PCB	
2.3 Fabrication of PCB's	
<b>3. Software</b>	<b>22</b>
3.1 Introduction to Assembly Programming	
3.2 Details of Testing Process	
3.3 Algorithm	
3.4 Flowchart	
3.5 Coding	
<b>4. Operating Instructions</b>	<b>74</b>
4.1 Test Procedure	
4.2 Handout	
<b>5. Debugging</b>	<b>82</b>
<b>6. Limitations of the System</b>	<b>83</b>
<b>7. Scope for Expansion</b>	<b>84</b>
<b>8. Conclusion</b>	<b>85</b>

Bibliography

Appendix A

Premier Polytronics Limited is a part of the Premier Mill group. The company was started in 1983 with technical and financial collaboration from Zellweger Uster. From 1996, Premier has been independently manufacturing and marketing worldwide.

Premier's complete range of products includes Fibre testing, Yarn testing, Online monitoring, Winding and continues to manufacture splicers with Mesdan, Italy. The customers of Premier are about 1000 spinning mills currently. More than 30% of the sales turnover comes from exports.

This ISO 9001 certified company had received the R & D award for the years 96-97 and 97 -98. This company has CE certification for most of its products and has received the Government of India Department of Electronics award for excellence in Electronics. Besides this, Premier has received the EEPC award for export excellence and the FITEI special export award in 98-99.

Any product of Premier Polytronics Limited has a seal of quality and is a symbol of consistent competence. Premier is a company surging ahead in tune with tomorrow's technology offering the best in the field of Textiles.

The PREMIER HFT TESTER is used to test the quality of various types of fibers of cotton. The HFT is the acronym to high volume fiber testing. The machine is semi automatic since the sample and the tested cotton has to be inserted and removed manually. All the parameters are measured indirectly.

### **Parameters to be measured:**

- Length of the fiber
- Strength of the fiber
- Micronaire of the fiber
- Colour of the fibre

### **Length of the fiber:**

The length of the fiber is measured using a separate module with the aid of OPTICAL MEASUREMENT. The optical module consists of a modulator and a demodulator. The modulator modulates the supply frequency in a suitable range. We are going for modulation and corresponding demodulation to avoid external light interference. A stepper motor is used with a step size of 1.8 degrees / pulse. The length is obtained from the number of steps moved by the stepper motor.

### **Strength of the fiber:**

The strength of the fiber is measured using a FORCE TRANSDUCER. The force transducer is of F-type. The force transducer consists of JAWS used for clamping the sample cotton. The jaw consists of



two parallel plates out of which one plate is fixed and the other plate is movable. The strength for breaking the cotton is converted into the equivalent difference current. This difference current is converted in to an equivalent voltage.

### **Micronaire of the fiber:**

In order to make the micronaire measurement, we place a known weight of cotton inside the system. A burst of air is supplied into the system where an air transducer is placed. This difference in air pressure is amplified and converted into an equivalent voltage from which the micronaire is calculated.

### **About the CPU board:**

The 80C320 controller board is that which controls and coordinates the functions of this ELECTROMECHANICAL machine. The machine is called as an electromechanical machine because the electronic board controls the movements of the pneumatic parts.

The Digital switches monitor the pneumatic part's movement and solenoid coils. The switch is ON when an input is given and OFF when an input is not given. The two inputs which the controller needs is the digital IN and the state of the digital switches. The supply given to the coil is called as digital OUT.

The pneumatic controls are established with the help of two stepper motors. Stepper motor-1 controls the direction of motion (*forward or backward*). Stepper motor-2 controls the pulse. 24v DC drives the stepper motor. In order to avoid back EMF in the circuit we go for Drag and Drivers.

Drag and Driver circuit is an important circuit, which is similar to a chopper, or a slipper circuit. The function of the dragon driver is to energize

the coil of the motor. With the toggling of the trigger switches, we can sense the movements of the pneumatic movements.

To have a user-friendly communication with the machine, visual basic 6 is used as the front-end tool. The computer sends the input serial data to the controller. In response, the controller sends the control signals to various parts of the system.

A Fibro - sampler is used to check if the required amount of cotton in order to check for the size, length and weight.

### **Sequence of operation for measuring length:**

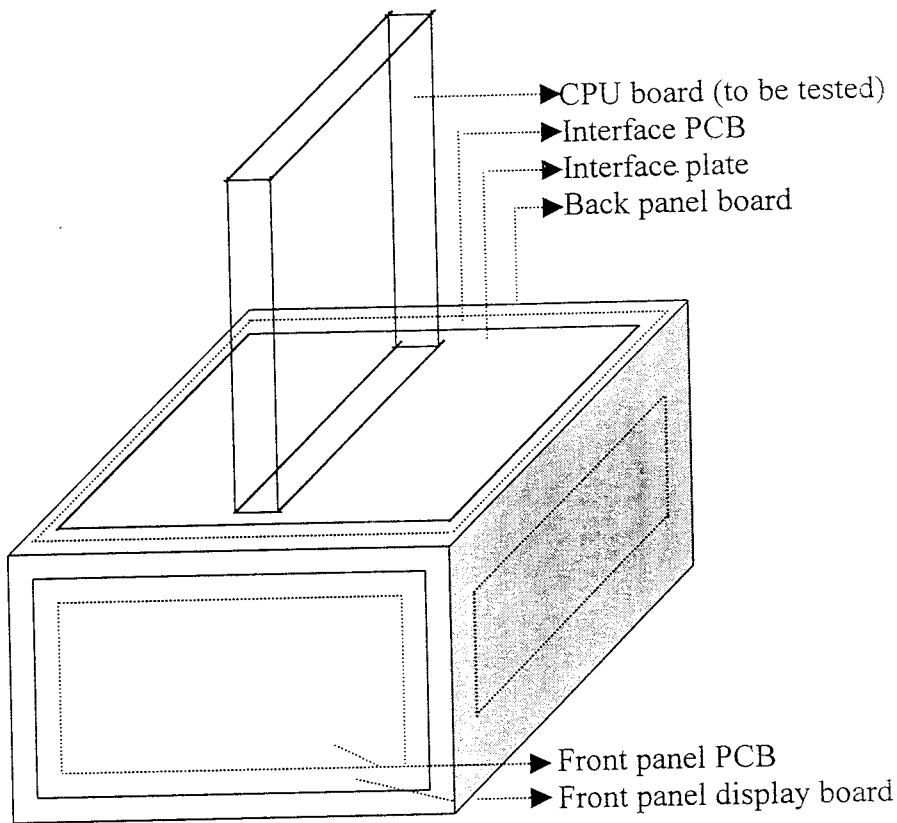
From the fibro sampler the sample cotton is placed in the machine on the pulse switch .Now the pulse switch is activated and is in the ON condition and its status is sent to an inductive sensor. Now again the inductive sensor sends a pulse to check whether the pulse switch is till in ON condition. If the condition is true, it causes the toggling of the switches in the optical module and hence the length is measured and displayed in the computer screen.

Our project deals with the testing of the CPU board of the Premier HFT Tester, which has been developed at Premier Polytronics Limited, Coimbatore. The various parameters to be tested are done by means of our Test Jig, which by large means reduces the time involved in debugging and facilitates easy tracking of errors.

The CPU board, which is to be tested, has a number of peripherals, the functionality of which is under test. A firmware project - the Test Jig can be divided overall into two parts – Hardware and Software. The Hardware side of the project includes the building of three Printed Circuit Boards namely the Power Supply Board, Interface board and the Display Board. The Front panel Display board and the Back Panel board are also present besides the PCB's. The Software side includes the assembly language programming which is dealt with in detail later. The wiring forms the bridge between the hardware and the software

The testing of the CPU board is made in a user-friendly manner, and the testing of the modules can be done in a simple yet effective manner. The errors may be identified and the needful may be done.

Area to be tested	Test Methods
1. 81C55	Set in Dip switches and the outputs are given to LED's.
2. Serial Communication	To transmit a fixed message in MAX 232 (U8), write a message in the program and receive the same message
3. RAM	Write data on NVRAM, retrieve it and display it in PC.
4. A/D Converter 5. D/A Converter 6. Multiplexer	Some Analog inputs are given, one of them is elected using a multiplexer In order to digitize we Pass it through a ADC and use a DAC to convert the digital signal. thus when the DAC outputs are Correct, all the above Peripherals are tested



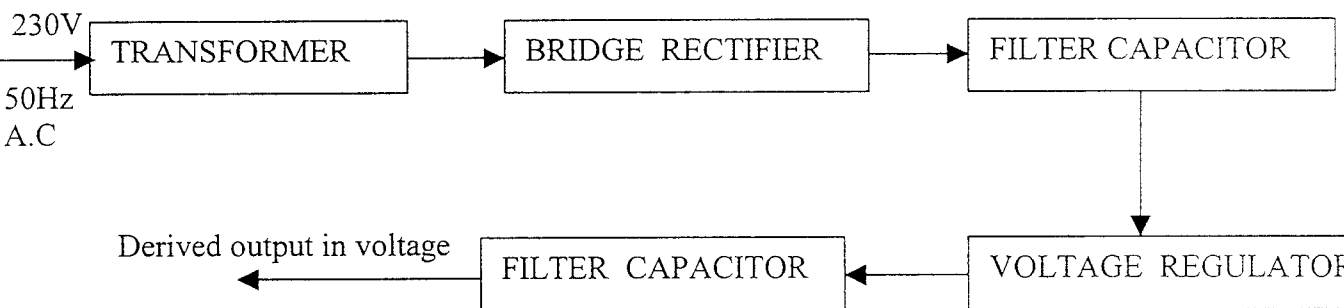
Test Jig

The PCB's that were fabricated are namely the Power supply board, Interface Board and Display Board. The power supply board, which provides the supply for the entire Test Jig, a transformer is present inside the Jig. The Interface PCB is placed parallel to the top plate of the Test Jig, so that the CPU board to be tested can be mounted on the jig through the J1 and J2 Euro connectors. The side of the Jig consists the power supply board. The interface board forms the junction of all the signals. The signals from the CPU board come to the interface board and then go to the display board. The display board consists of all the input and output signals. The display PCB lies behind the Front panel, which consists of all the control switches and output LED's.

## Required output from power supply:

VOLTAGE	CURRENT
+5V	1A
+15V	0.5A
-15V	0.5A
+24V	0.5A

## Block diagram of power supply:



## Description of power supply circuits:

The blocks used in the circuit are

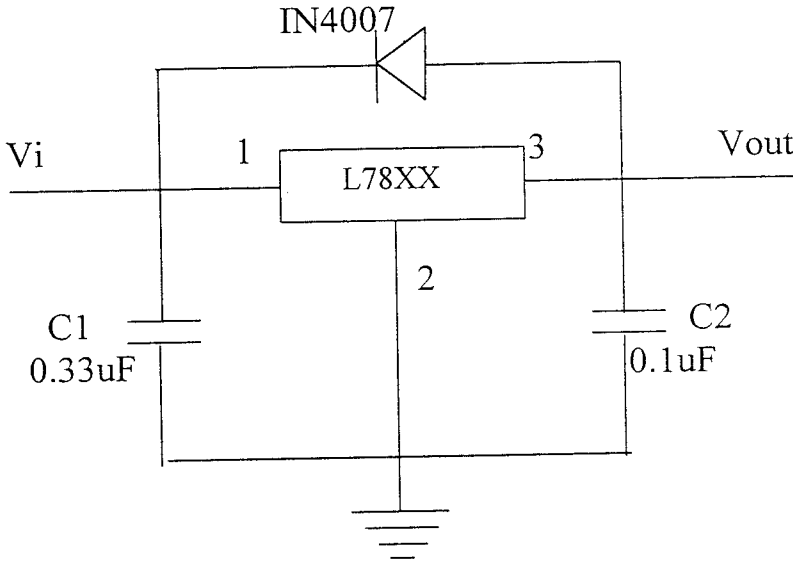
- Transformer
- Bridge
- Filter capacitor
- Voltage regulator

The transformer used is a step-down transformer. A bridge rectifier is used followed by a voltage regulator. For positive output voltages we use a positive voltage

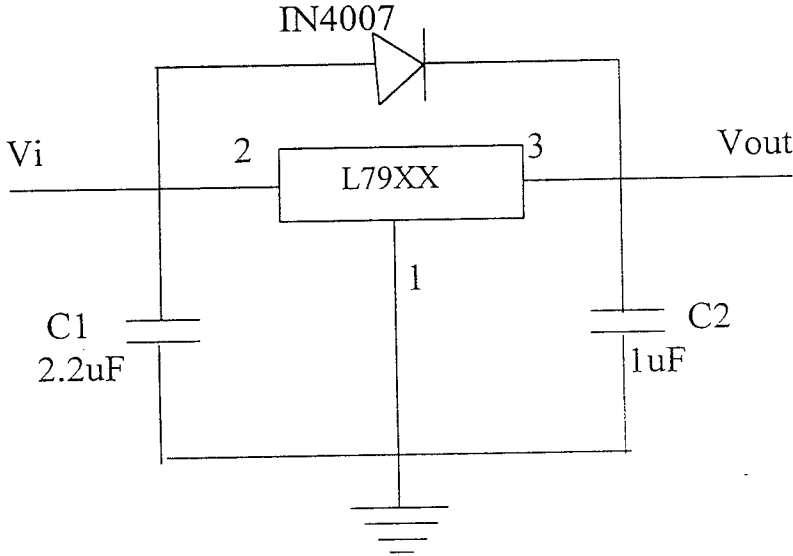
regulator in the 78 series and for a negative output voltage we use a negative output voltage regulator in the 79 series

**Application Circuit:**

**Positive voltage Regulator:**

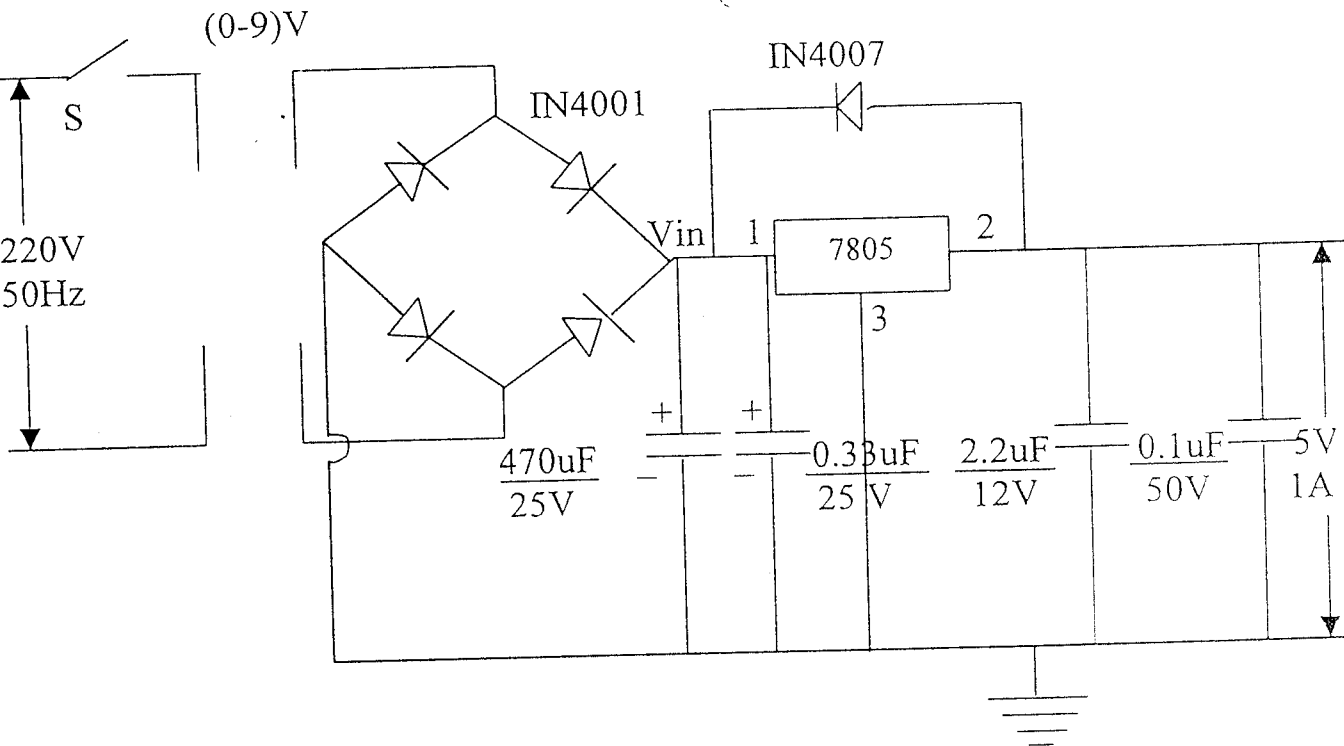


**Negative voltage regulator:**



The diode protects the device when the output voltages are higher than input during differential load conditions, during power on/off etc..

### Circuit diagram for an output of 5V/1A:



voltage range used for 5V/1A is (8-20)V

$$V_o = 5V$$

$V_{in}$  = a minimum of 8V

~10 V D.C

$$V_{dc} = 1.2 V * V_{ac}$$

$$= 10V$$

$$V_{ac} \sim 8.33V$$

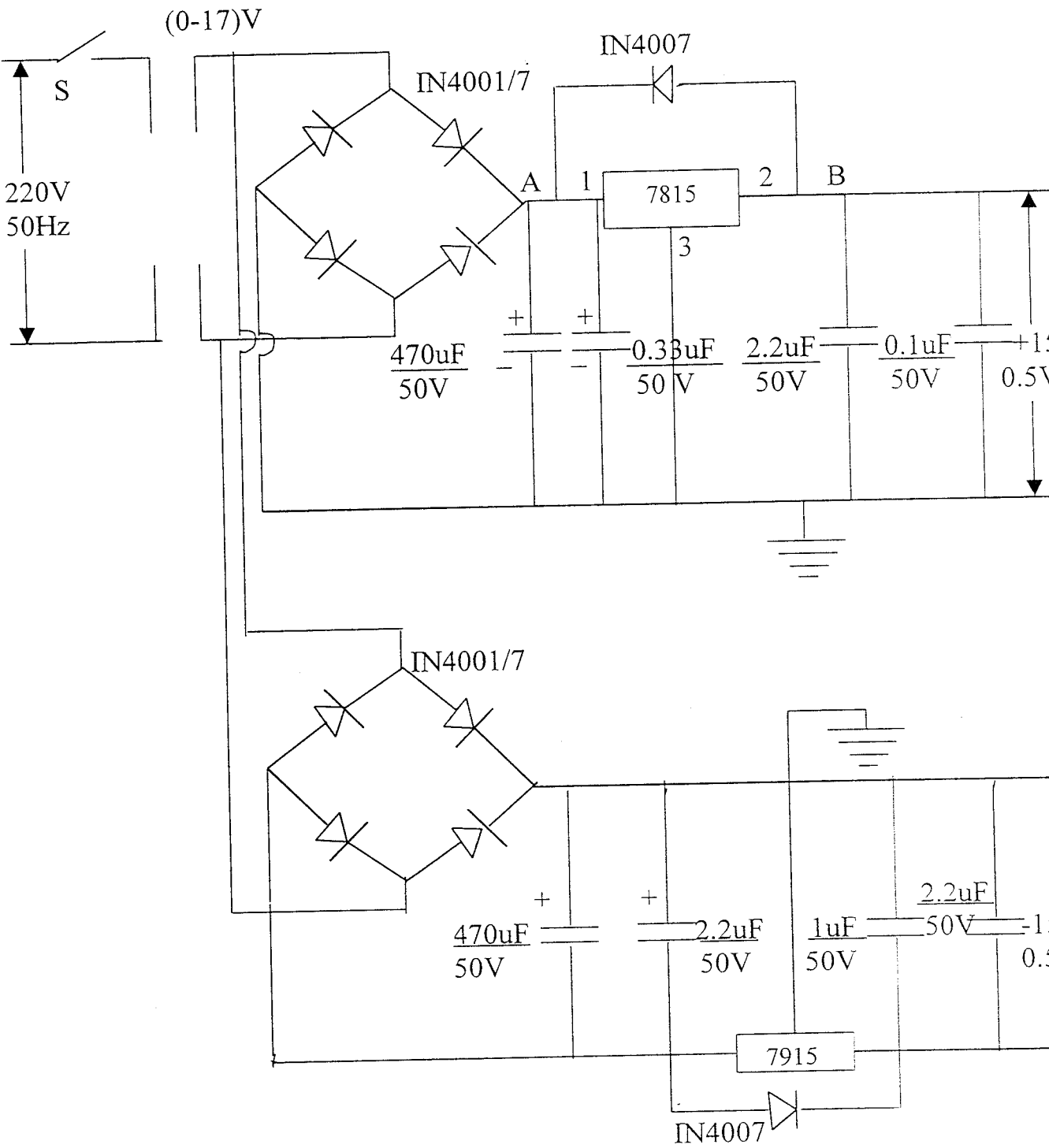
So we require (0-9)V transformer.

$$V_{in} = 11.7V$$

$$= 12V$$

# Circuit diagram for 15V and -15V output(0.5A):

Voltage range used: (18-30)V and (-18.5 to -30)V





Voltages at points A and B:

$$V_o = 15V$$

$$V_{in} = \text{a minimum of } 18V$$

$$\sim 20 \text{ V D.C}$$

$$V_{dc} = 1.2 \text{ V} * V_{ac}$$

$$= 10V$$

$$V_{ac} \sim 16.6V$$

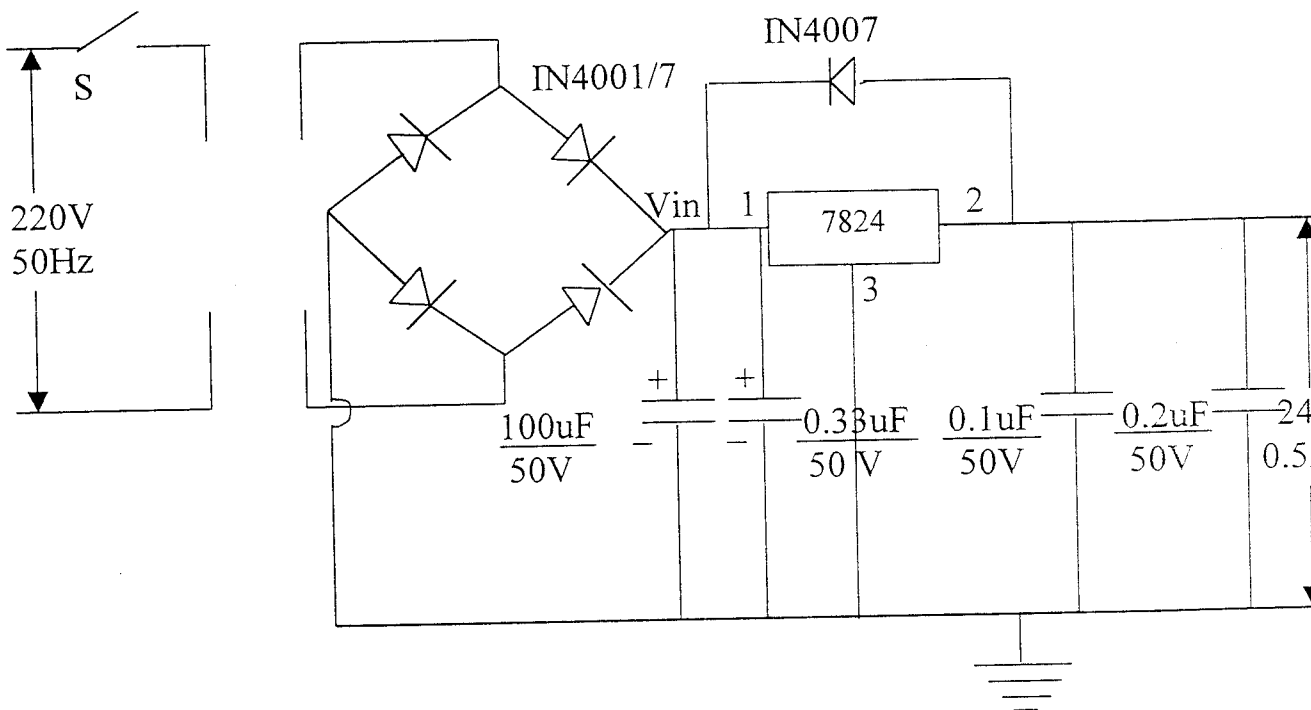
$$V_{ac} \sim 17V$$

So we require (0-17)V transformer.

$$V_{in} = 11.7V$$

$$= 12V$$

### Circuit diagram for an output of 24V/0.5A:



voltage range used for 24V/0.5A is (28-38)V

$$V_o = 24V$$

$$V_{in} = \text{a minimum of } 28V$$

$$\sim 30 \text{ V D.C}$$

$$V_{dc} = 1.2 \text{ V} * V_{ac}$$

$$= 26.66V$$

$$V_{ac} \sim 27V$$

So we require (0-27)V transformer.

## Printed circuit board:

For 7805(ie 5V) package To3 package

All the rest,+12V,+15V and -15V,we use the 220 package,

The input voltages provided to board , are AC voltages:

0-12V used for +5V

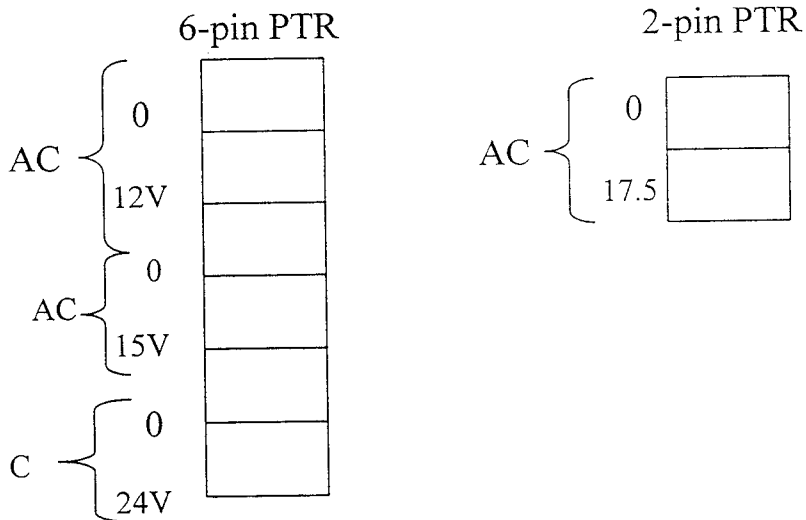
0-15V used for +15V

0-24V used for +24V

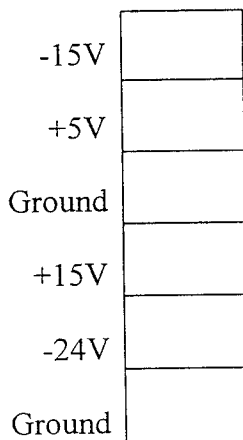
0-17V used for -15V

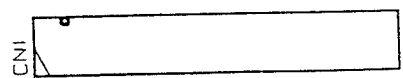
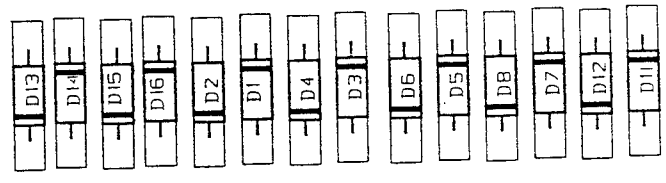
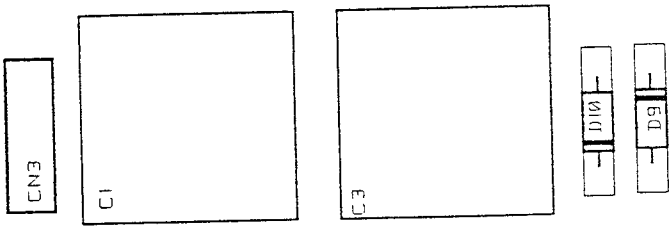
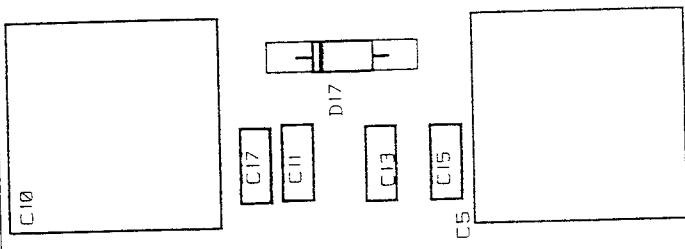
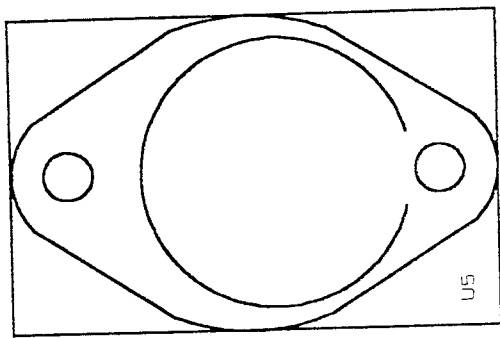
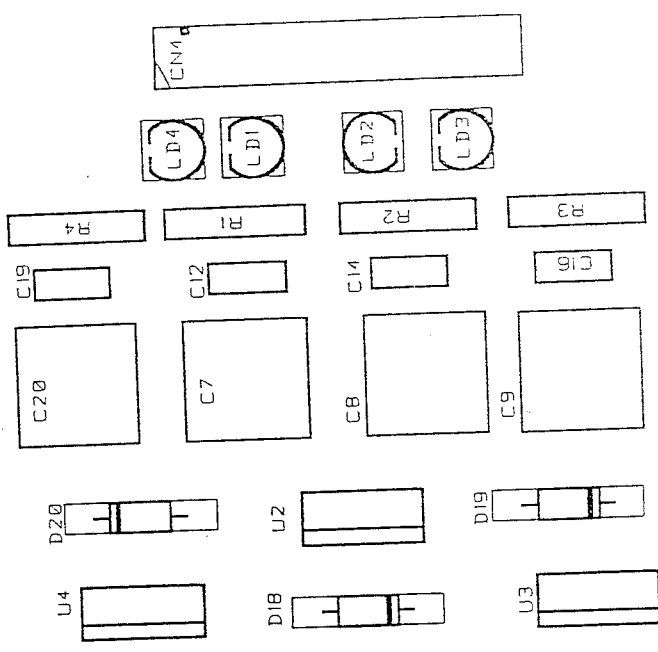
We may use a 6-pin PTR, individual 2-pin PTR's for the positive voltages. A separate 2-pin PTR is given to avoid confusion.

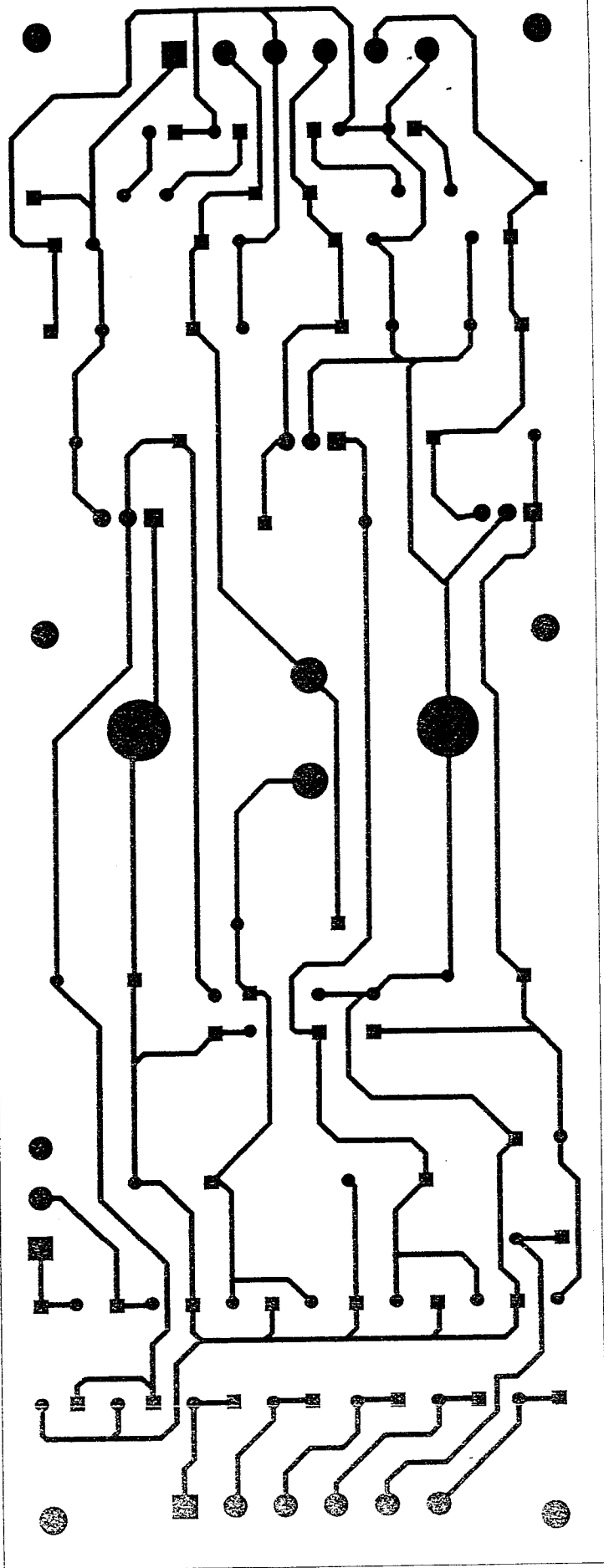
## Input side:



in the output, PTR's are connected, again a 6-pin PTR and voltages may be checked using a DMM.







# INTERFACE BOARD

As the name indicates this board forms an interface or interlink between the motherboard, which is to be tested and the display board where the outputs are indicated. The signals from the motherboard are received in the interface board and they are split up depending on their functionality.

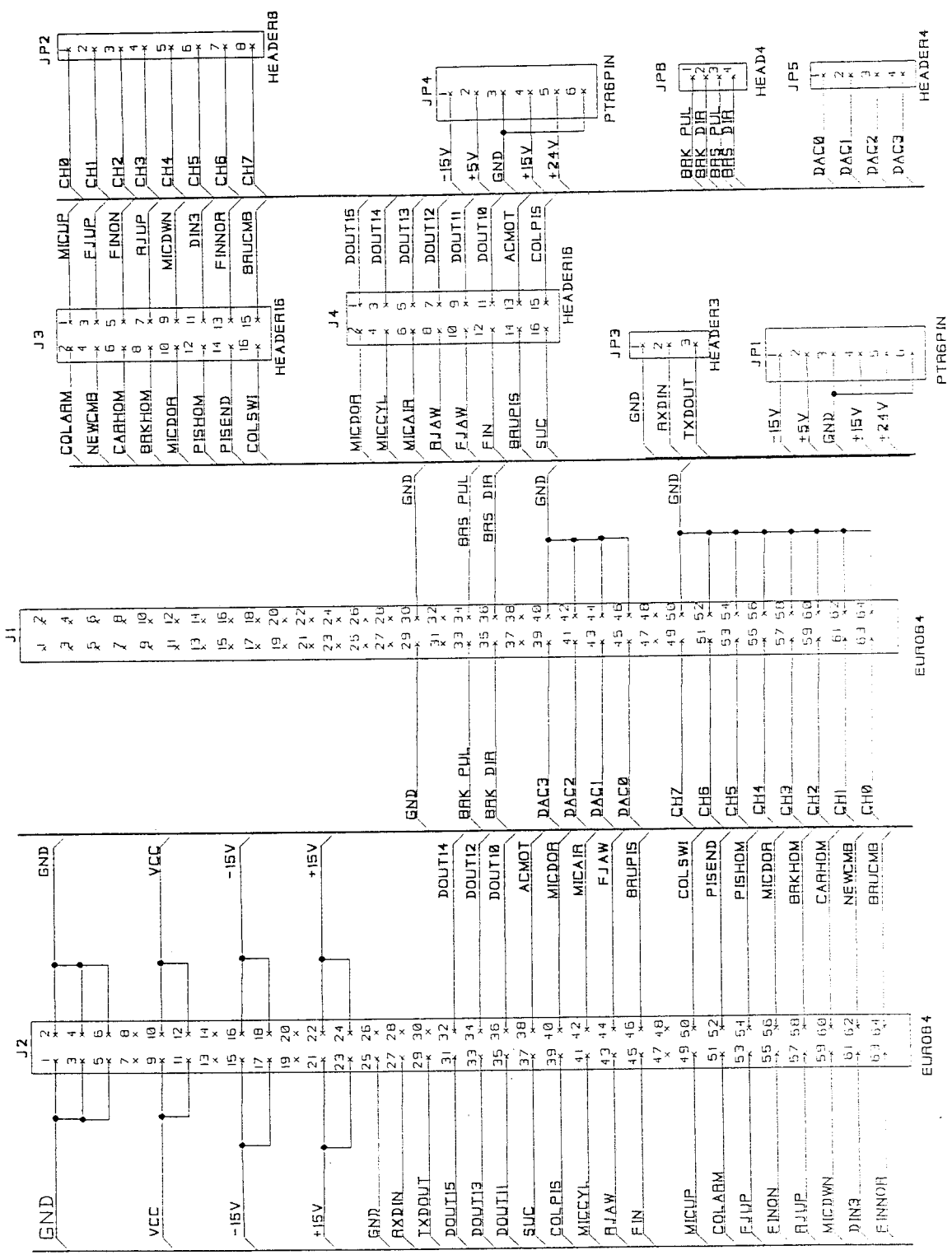
They are

- Signals going to the display PCB
- Signals going to the back panel
- Power supply signals

## DESCRIPTION OF THE COMPONENTS USED

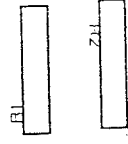
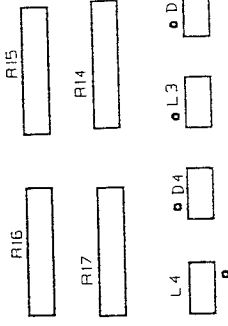
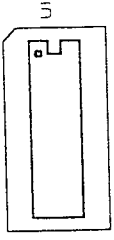
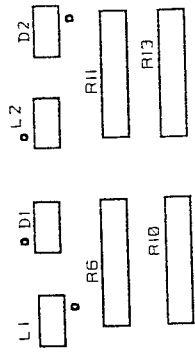
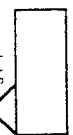
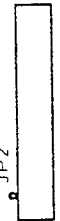
- A 6 pin PTR has all the power supply signals necessary for the jig (i.e.) +5 V, +15 V, -15 V, +24 V. The 24 V provision is left for further use.
- The 16 digital inputs and 16 digital output signals are divided using four 8-pin headers. Now these signals are connected to the front panel end components like switches and the LED's using external wiring.
- A 3 pin header which has the serial communication transmission signals like Txd and Rxd
- There is a 4 pin header which connects to the switches S17, S18, S19
- (to be obtained)
- A 6-pin header, which contains all the DAC, signals going to the display PCB for further comparison.
- A 6- pin header, which has external wiring with, the rotary switch in the front panel.

Thus all the signals are separated and used as required. The interface board has two euro-64 connectors upon which the 2- Euro connectors of the motherboard will sit. Thus the interface board provides total interface to the signals of the jig.

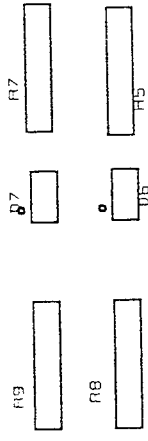
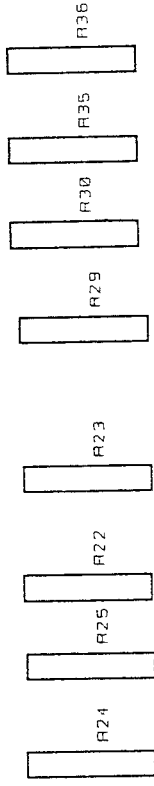
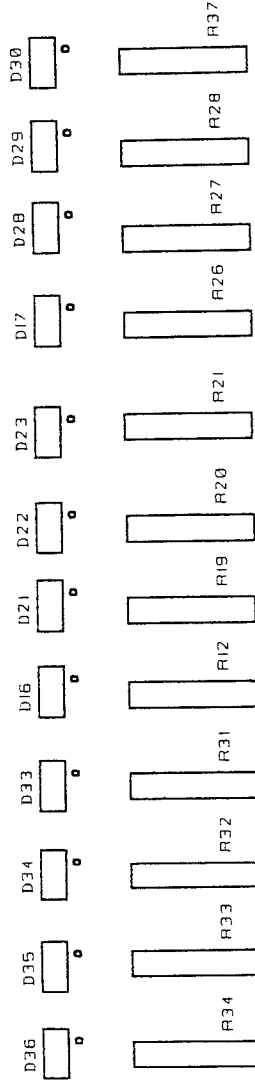


EURO64

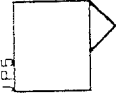
EURO64



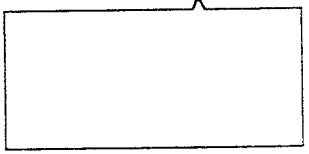
JP6



U2



JP5



CN2

## DISPLAY PCB

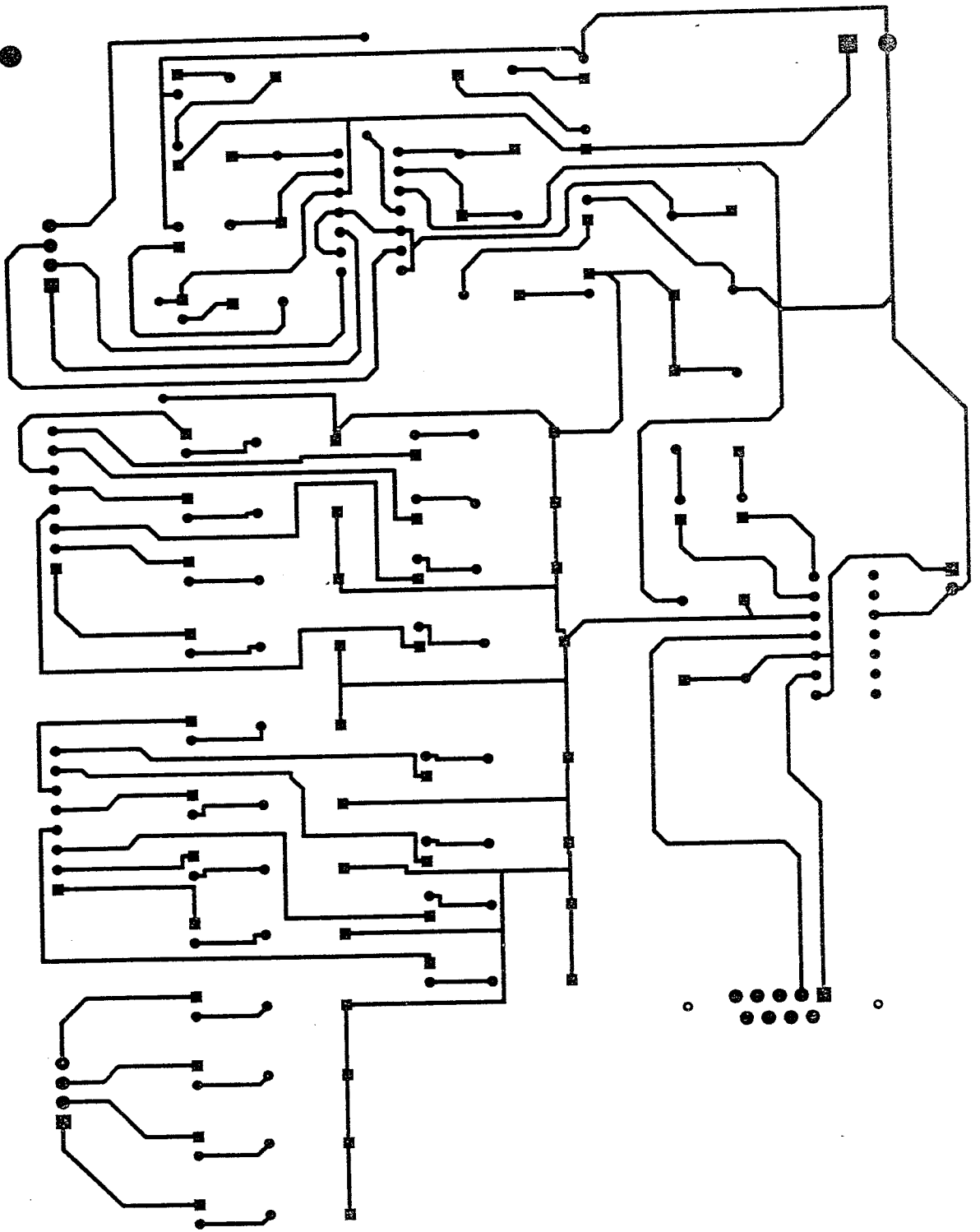
The display PCB contains the necessary display circuitry. The end components like the switches and the LED 's are placed on the front panel and the corresponding wiring is done from the display PCB to the front panel and from the front panel to the interface board.

In the display PCB we have the current limiting resistors necessary for the LED's. We have the lm339 comparator IC's to compare the DAC output voltage levels and the fixed voltage level of 2.29V

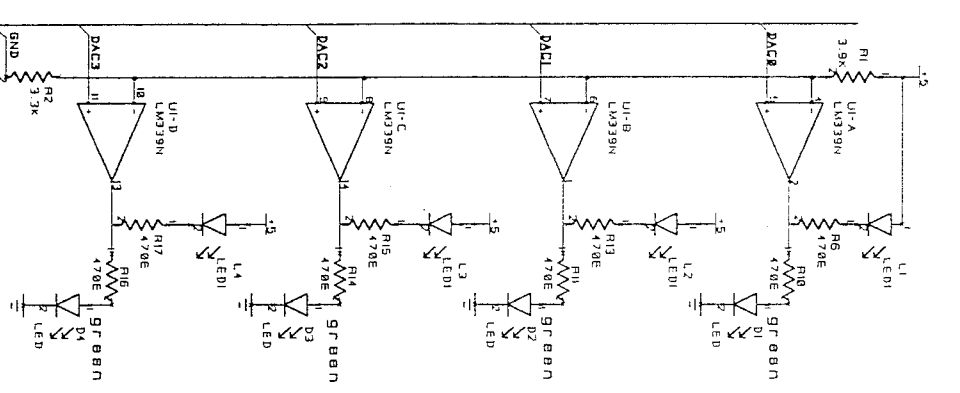
### Components list:

	Components	Quantity
<i>Interface board.</i>	Resistor network	2
	Relimate conn-8pin	5
	Relimate conn-4pin	3
	Relimate conn-3pin	1
	Relimate conn-2pin	1
	PTR connector-6pin	2
<i>Display board.</i>	Resistors-3.3k	1
	3.9k	1
	4.7k	2
	470E	30
	DTCON-9pin	1
	Relimate-2pin	2
	Relimate-4pin	1
	LED	30
	LM339	2
	PTR-2pin	1
Relimate-8pin	2	

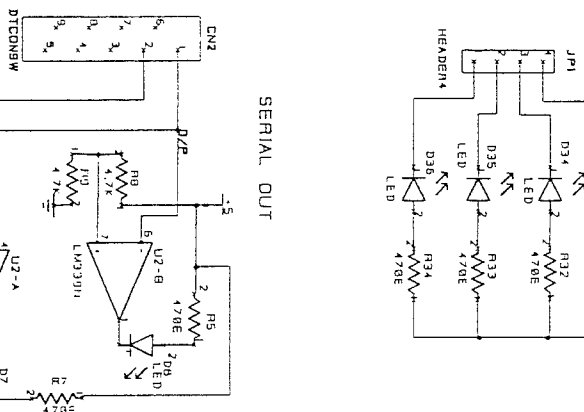




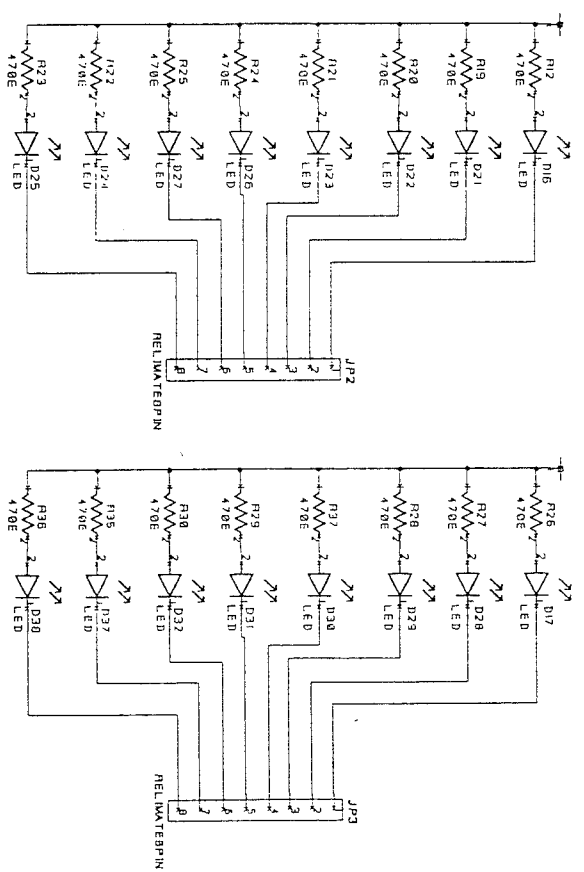
ANALOG OUTPUT



Digital outputs



Digital outputs



DRAWN	SIZE	FSCM NO.	DATE
ISSUED	SCALE		
			SHEET

The process of making of the PCB involves the following steps namely

- Drawing the schematic
- Routing the PCB
- Obtaining the legend
- Plotting
- Sketching the routes using copper ink
- Etching
- Alcohol Treatment
- Assembling of the Components

### **Drawing of the Schematic:**

The schematic is drawn using software called the Cadstar. This software has a library, which contains almost all the electronic components, which are necessary for the PCB. The electronic components are also classified according to their ratings. That is, for example in the case of resistor, the various range of resistors like 1K, 10K, 100K etc are present which can be made use according to the necessity. Thus using this library we select the components required and draw the schematic.

### **Routing the PCB:**

Now the schematic diagram has to be routed by using the option called Auto Routing. In order to fulfill our circuit requirements some changes are made in the auto routed PCB manually and finally we get the legend ready.

## **Plotting of the PCB:**

From the legend obtained the points where the components are to be inserted are plotted using the plotter. Then as the next step, the copper coated side of the PCB is filed mildly using Eversheet.

## **Sketching the Routes using Copper ink:**

A special type of ink is used to sketch the routes. The above process is done manually. Using photo-filming technique can also do this. There is not being any intersection of the lines, have a start in used to avoid the intersection. .

## **Etching:**

The process of etching is done using ferric chloride solution. The PCB is placed in a tray containing the above solution. The tray is shaken slightly so that the ferric chloride solution will react with copper. The copper chloride obtained as a result dissolves in the water and is separated. Thus the routes (where the ink is present) above the copper stay making the routes very clear.

## **Alcohol Treatment:**

The PCB obtained in the above step is cleaned using Ether solution. Thus the board is now ready for assembling.

## **Assembling of the components:**

Now the components are assembled on the board and they are soldered. Now the board is ready.

The central idea of the working of the project is that the EPROM in the CPU board is removed and replaced by a test EPROM. The test EPROM overrides the entire board. The software necessary for the controlling the micro controller and other hardware is loaded into this EPROM.

Before loading or in fact programming the EPROM, the assembly language codes are written using the assembler. The assembler that we have used here in our project is called ASM A.51. Intel Corporation developed this assembler for the Dallas make micro controller chip 80C320. The software, which is written, is loaded in the assembler and finally compiled. Now this particular file is converted into hex file and is used to program the test EPROM. Now our software sits on the board. The assembler Asm.a51 is very easy and comfortable to use. It runs on DOS.

The instruction set of 80C320 shares a lot in common with the 8051 family. There are 111 instructions. They are based on the following categories:

- Arithmetic Instructions - 24
- Logical Instructions - 25
- Data Transfer Instructions - 28
- Bit Manipulator Instructions - 12
- Program Branching Instructions - 22

## **Features of the Instruction Set:**

Instructions are faster than the original 8051. The numerical average of all op codes is approximately a 2.5 to 1 speed improvement. Speed sensitive applications would make the most use of instructions that are three times faster. Another important feature is the dual data pointer feature. It allows the user to eliminate wasted instructions when moving a block of memory.

## **AREAS TO BE TESTED :**

- DIGITAL MODULE
- ANALOG MODULE
- MEMORY MODULES:
  - INTERNAL MEMORY MODULE
  - EXTERNAL MEMORY MODULE
- POWER ON SELF TEST

### **DIGITAL MODULE:**

In the digital module, we have two 8155 peripheral port interfaces. Each 8155 PPI consists of three ports namely port A, port B and port C. Out of which Port A and port B are 8 bit ports and Port C is a 6 bit port. We now use the two 8 bit ports for the purpose of checking the Digital module.

The logic behind checking the Digital lines is as follows

- 1) Configure port A and port B of one 8155 PPI as input port.
- 2) Configure port A and port B of the other 8155 PPI as output port.
- 3) The inputs are given to the ports through SWITCHES.
- 4) The outputs are displayed in the corresponding LED'S.
- 5) When one or more switches is closed, the voltage scan loop will detect the corresponding line or lines and send the outputs to the corresponding LED'S
- 6) The glow of the LED'S indicates that the corresponding lines are without any error.
- 7) If any LED doesn't glow it indicates that an error has occurred either in the LED or in the corresponding ports.

## ANALOG MODULE:

In the analog module of the original CPU system, we have an 8 to 1 multiplexer with 8 analog signal lines as inputs. The output line of the multiplexer is tied with the 12 bit bipolar (+10V to -10V) ADC. The output of the ADC is given to the 8 bit unipolar (0 to +5V) DAC.

The logic for the checking the ANALOG MODULE is as follows,

- 1) An analog voltage of 2.2Volts D.C is generated with a power supply voltage divider network and this voltage is supplied to the 8 analog input lines of multiplexer through an 8 way rotary switch so that only one input line is supplied at a particular time.
- 2) The line is supplied by the switch is sensed and the corresponding value for selecting the sensed line is given to the selection lines of multiplexer.
- 3) Hence the particular analog voltage line is selected and the signal is sent to the ADC. In the ADC, conversion is initiated by giving appropriate control signals and the control loop is made to wait till the end of conversion indication by specific signals.
- 4) Once we get the end of conversion signal, the digitized data is read from the ADC and is stored in convenient RAM locations, after the completion of the above process the control is given to the DAC.
- 5) In the DAC, the digitized data from the ADC is again converted to analog voltage.
- 6) The analog output from the DAC is compared again with 2.2Volts with +/-10% tolerance in an external comparator circuit using LM339, which drives green LED to the forward biased condition.

Hence, when the output voltage generated by the DAC crosses the specified threshold, the green LED glows showing that there is no error.

Incorporating appropriate conversion routines does the conversion of digitized data from 12 output lines of ADC to 8 input lines of DAC.

## **MEMORY MODULE:**

- 1) Memory locations from the first location are selected one by one sequentially. Data is written arbitrarily in the same sequence for all the locations.
- 2) Now, the data in the RAM locations are read back and verified for their correctness by comparing with the original data.
- 3) If it is found that the data in the memory location has been corrupted, we indicate that glowing the appropriate LED'S has damaged the RAM.

## **POWER ON SELF-TEST:**

A self-check routine is included in the system firmware. This routine is positioned at the starting of the main routine. Thus routine flashes all the LED'S in the display panel board to ensure the proper working



## **Digital Module**

U5: 8155 PPI (Programmable Peripheral Interface) with Port A, Port B, Port C as output ports.

U4: 8155 PPI (Programmable Peripheral Interface) with port A and port B as input ports and port C as output ports.

- Move the address of port A of U4 to data pointer.
- Read the data from the port A (i.e.,) input from first 8 switches (S1-S8) and debounce these values.
- Move the address of port A of U5 to the data pointer and move to debounced data to port A of U5.
- Now the address of port B of U4 is moved to the data pointer and the above mentioned process is repeated for port B.
- All the steps mentioned above are put inside a loop and this total scan loop is rotated in short intervals.

## **Memory Module**

### **(a) External Memory Module:**

- In the CPU PCB the externally interfaced memory ranges from 6000h to 7FFFh.
- The data pointer is made to point the starting location (i.e.,) 6000h.
- Sample data for writing into the RAM is stored in registers and is incremented, as the data pointer gets incremented.
- Data is moved from the register and is written into the location pointed by the data pointer.
- Now the accumulator is cleared and then the data from the written location is again moved to the accumulator and is compared with the original data previously written.

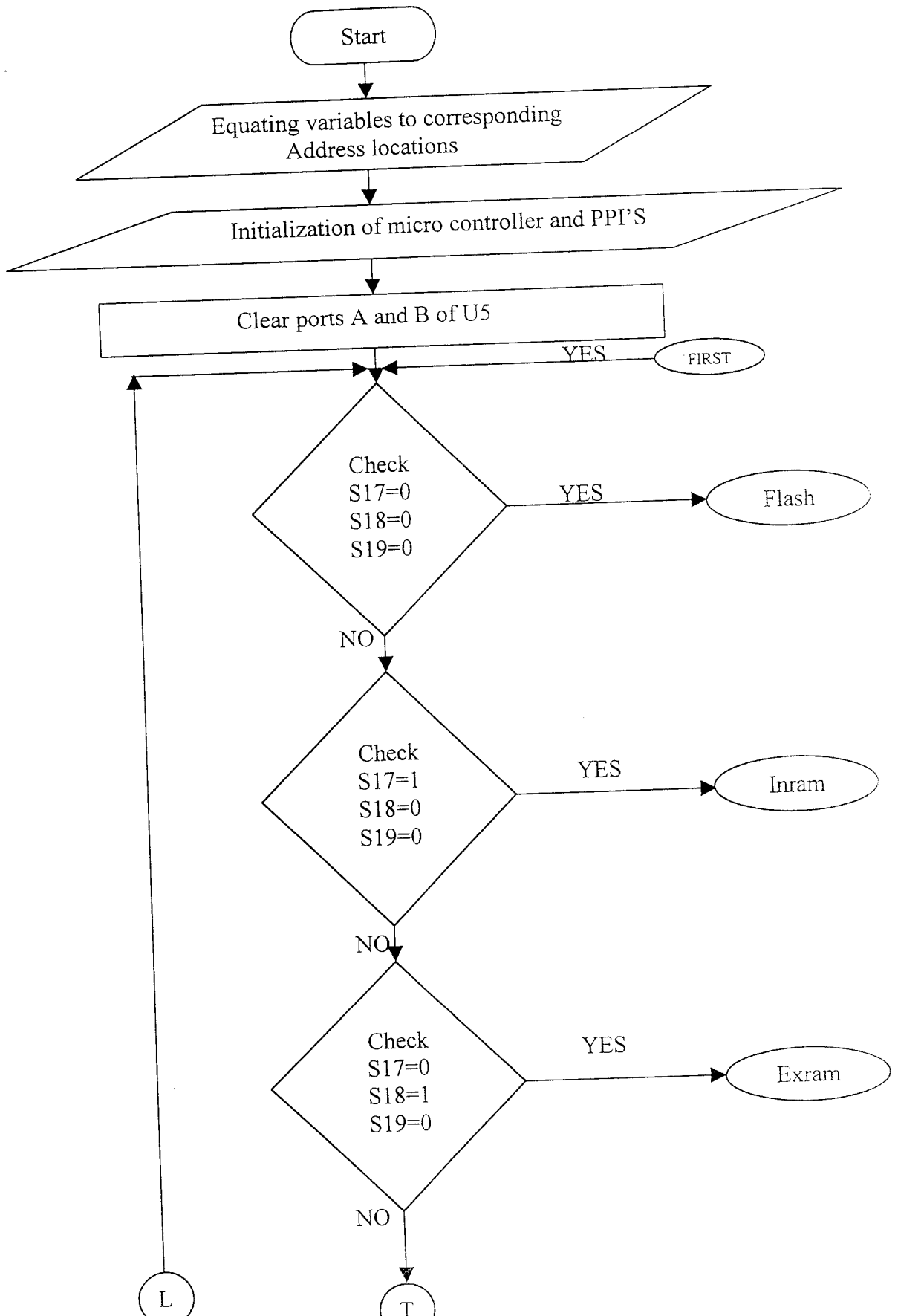
- If the data's match perfectly for all locations, the OK signal is displayed.
- If even one location looks damaged an error signal is displayed.

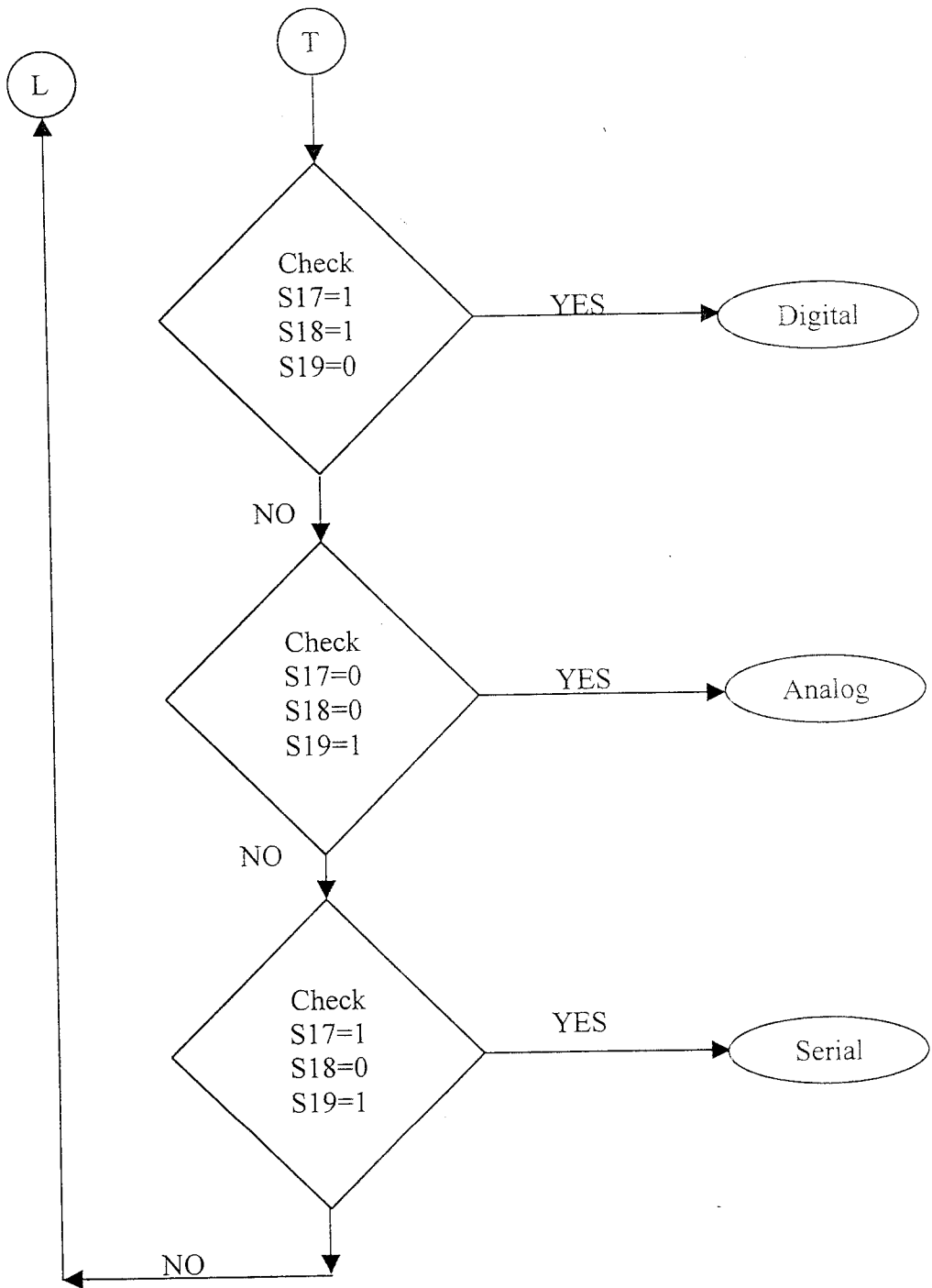
### **(b) Internal Memory Module**

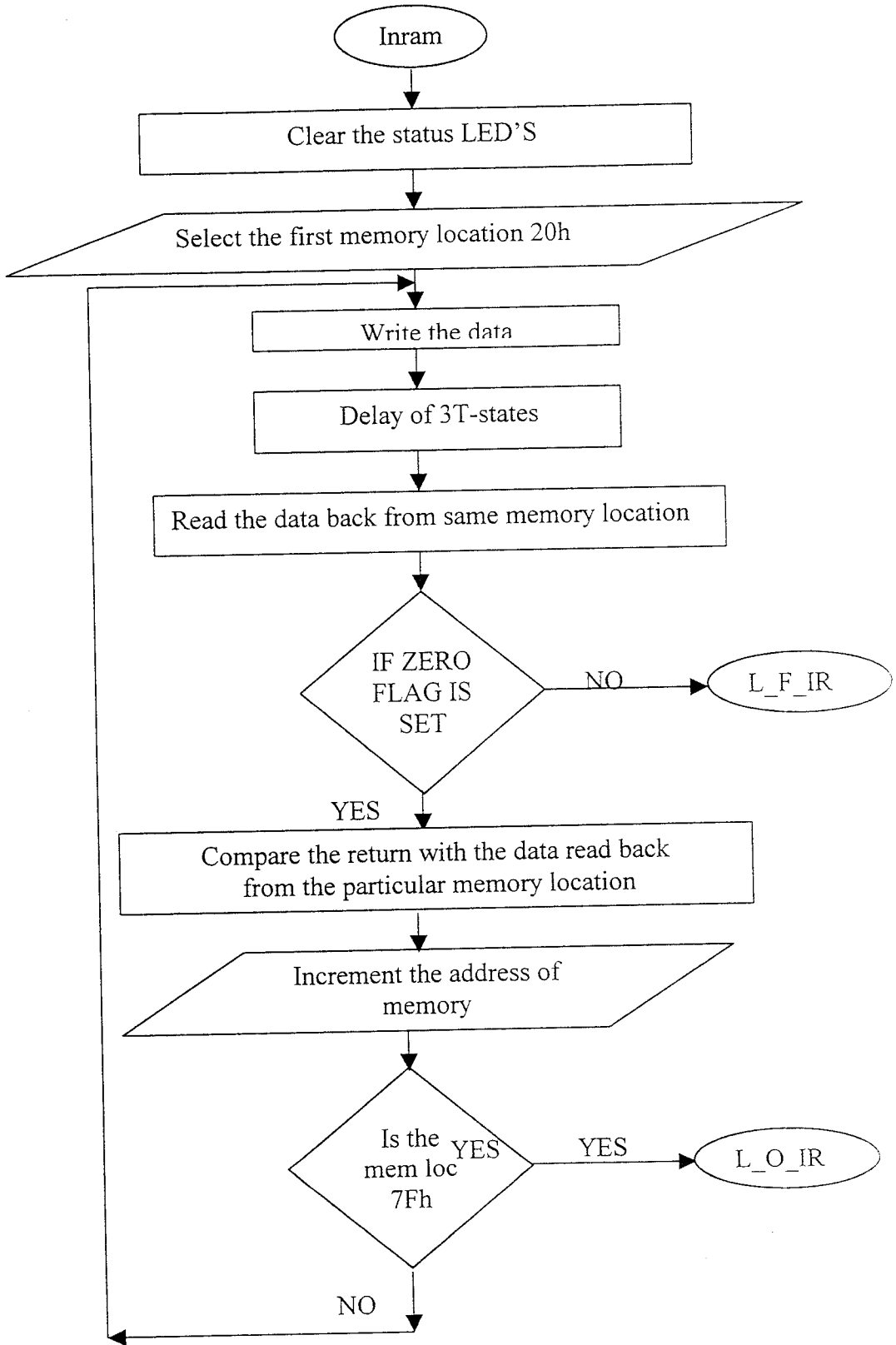
- Inside the micro controller 80c320, we have ON chip scratch pad RAM with user area from 20h to 7Fh.
- The same logic applied in the testing of external RAM is applied here.

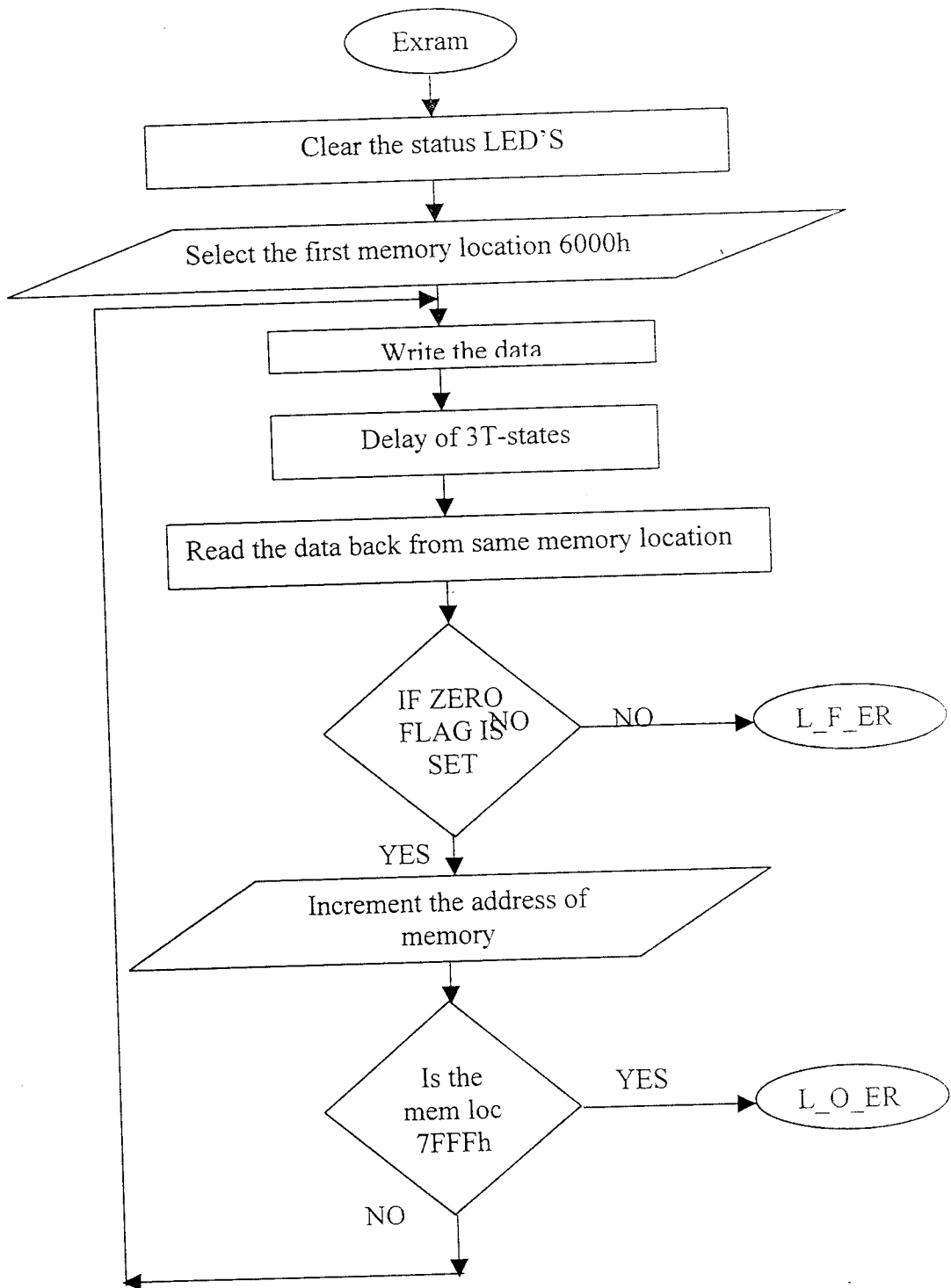
### **Analog Module**

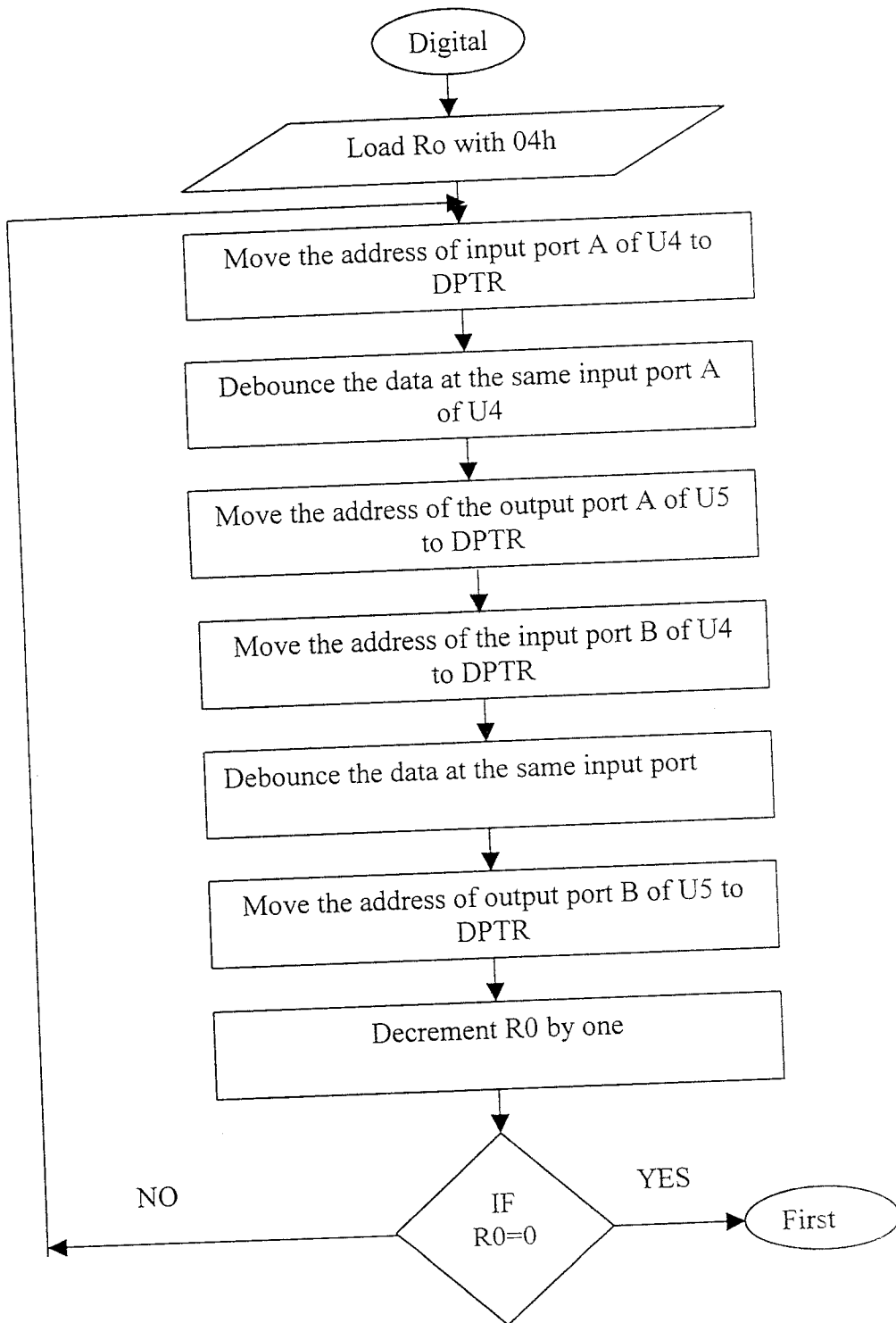
- An analog signal of 2.2V is given to an 8 way rotary switch, which supplies 8 inputs of the multiplexer (CH0-CH7). Hence at a particular point of time, only one analog channel (multiplexer input) is fed.
- The analog channel fed at a particular time is identified with the aid of digital switches S1-S8.
- Selection values are outed at port C of U4 into the selection lines of the multiplexer. Hence the supplied channel is selected and connected to the ADC.
- Initiate conversion in ADC and continuously check for End Of Conversion (EOC) signal and store the digitized data in separate memory location.
- The data stored in the memory location is compared with the corresponding values for it's upper and lower limits.
- If the conditions are satisfied on comparison, appropriate value is sent to the DAC in order to drive the green LED, which is the OK signal and the error signals are displayed.

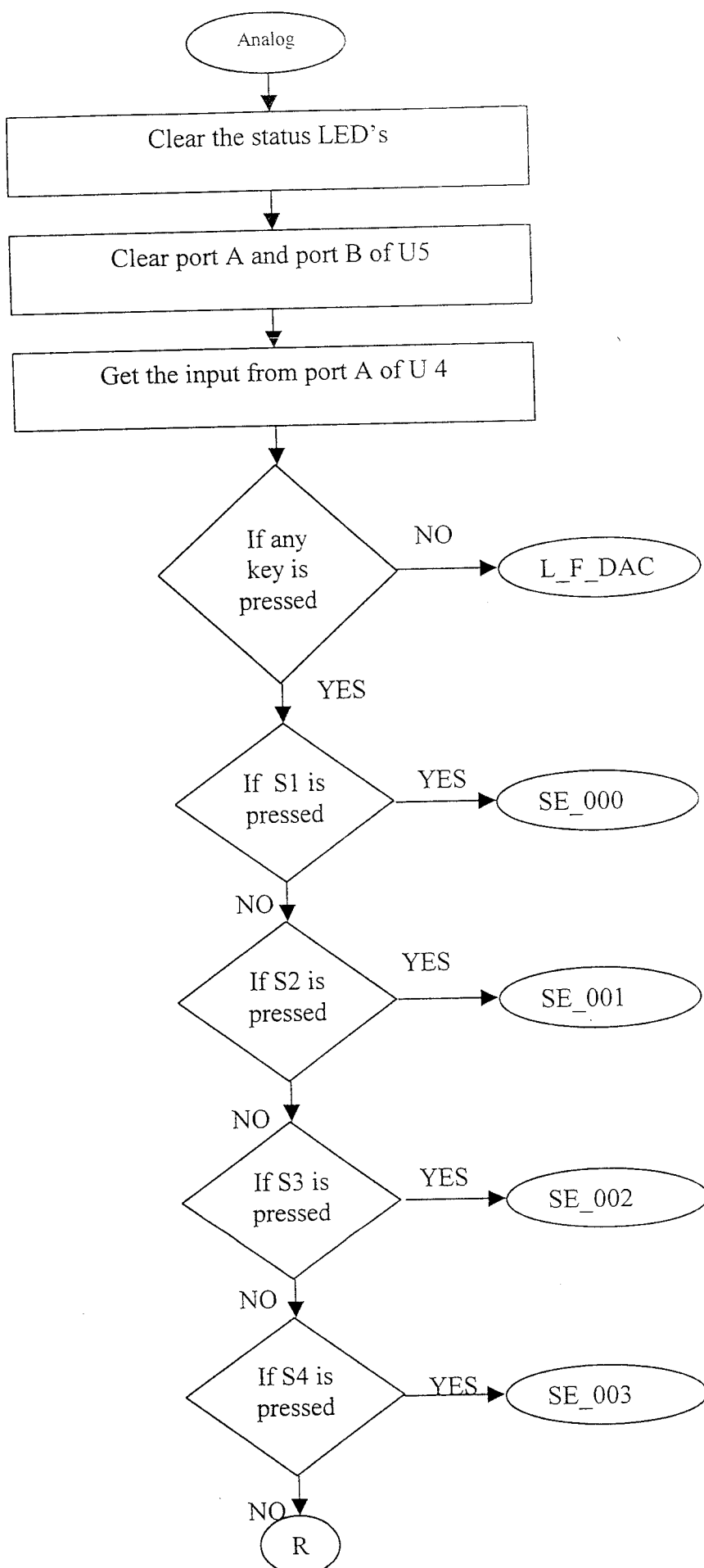




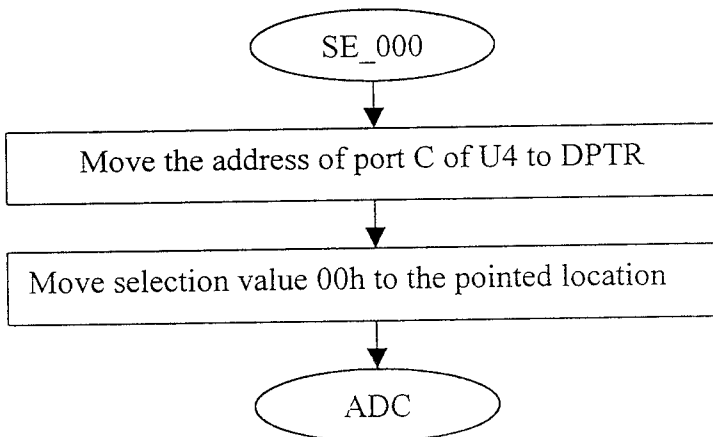
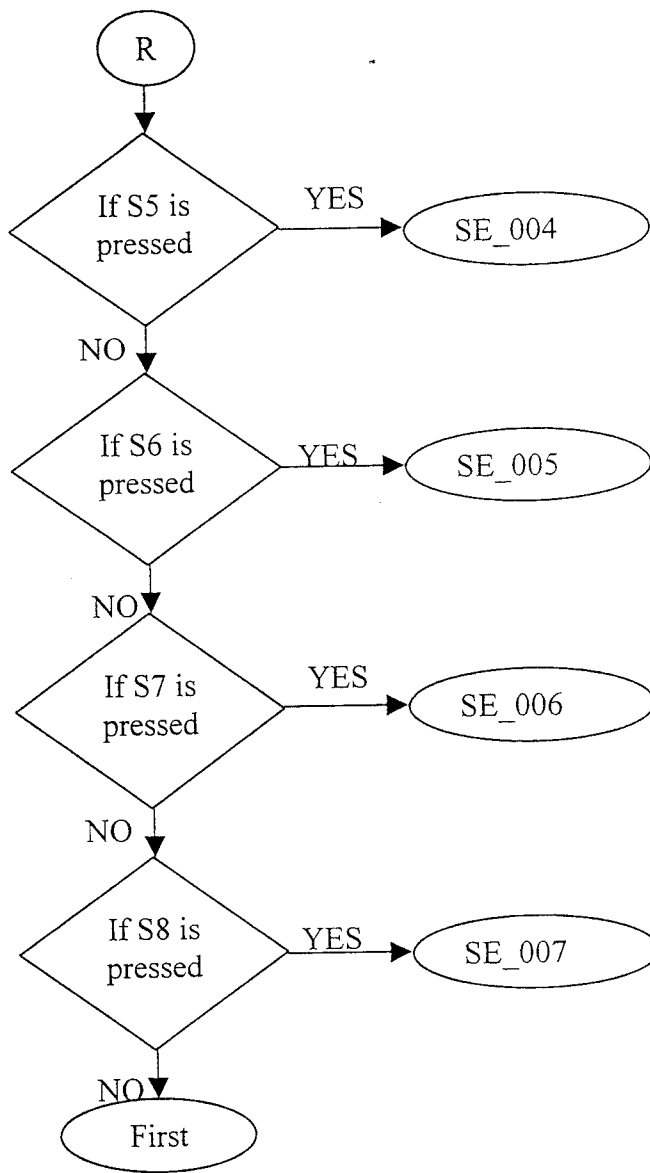


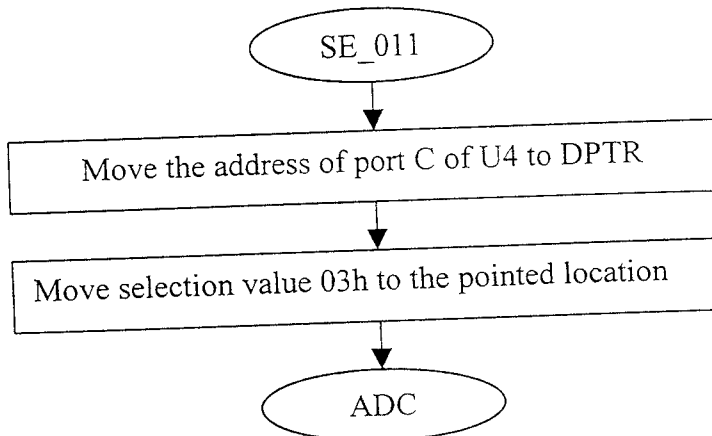
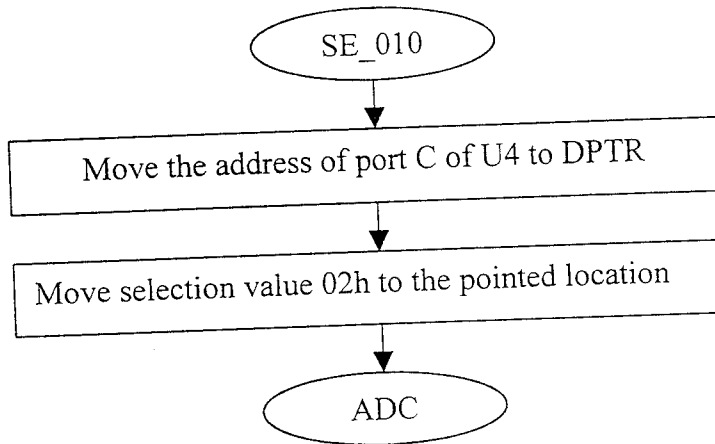
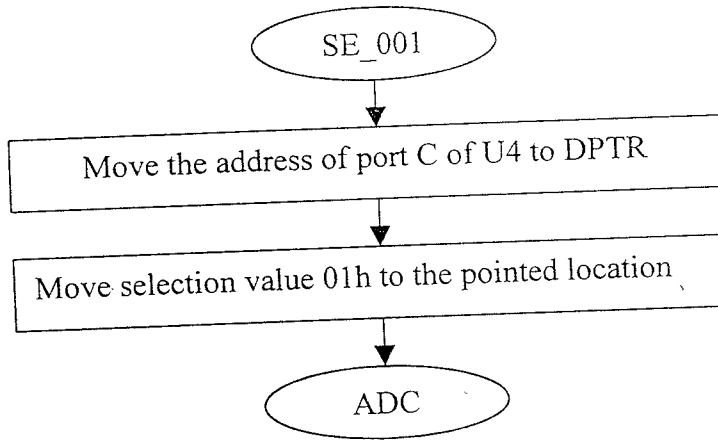


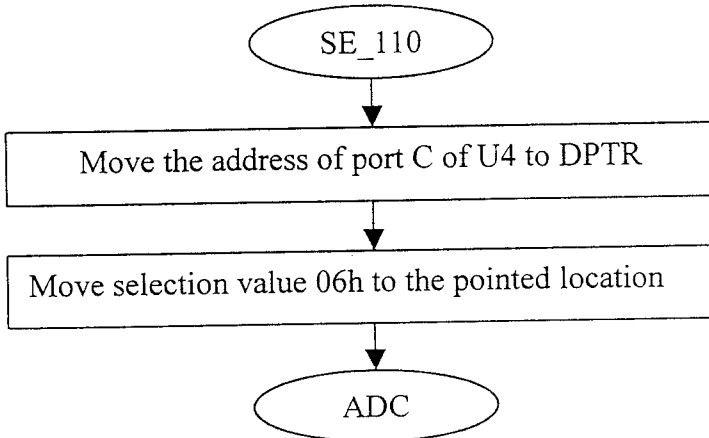
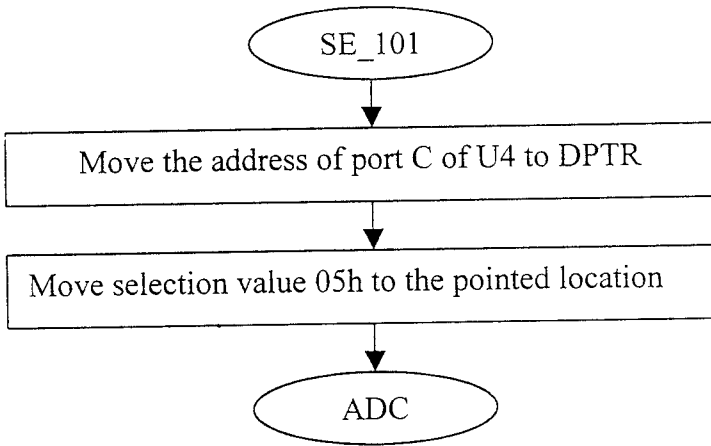
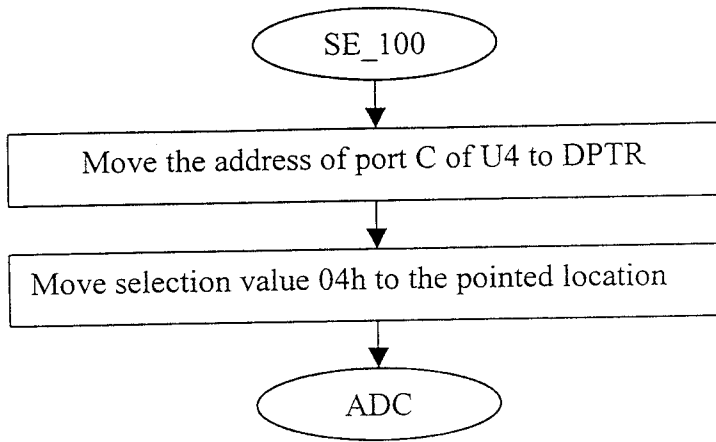


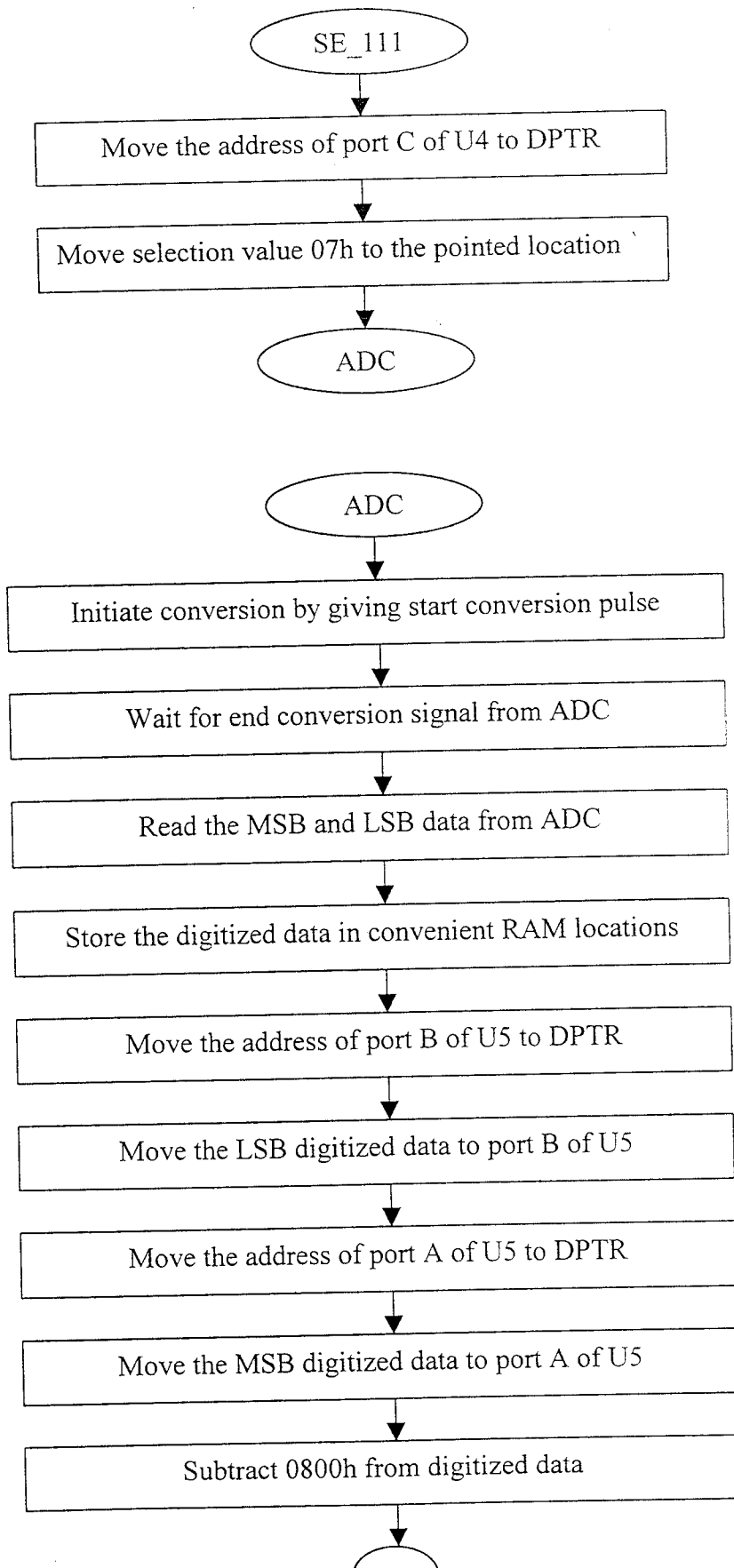


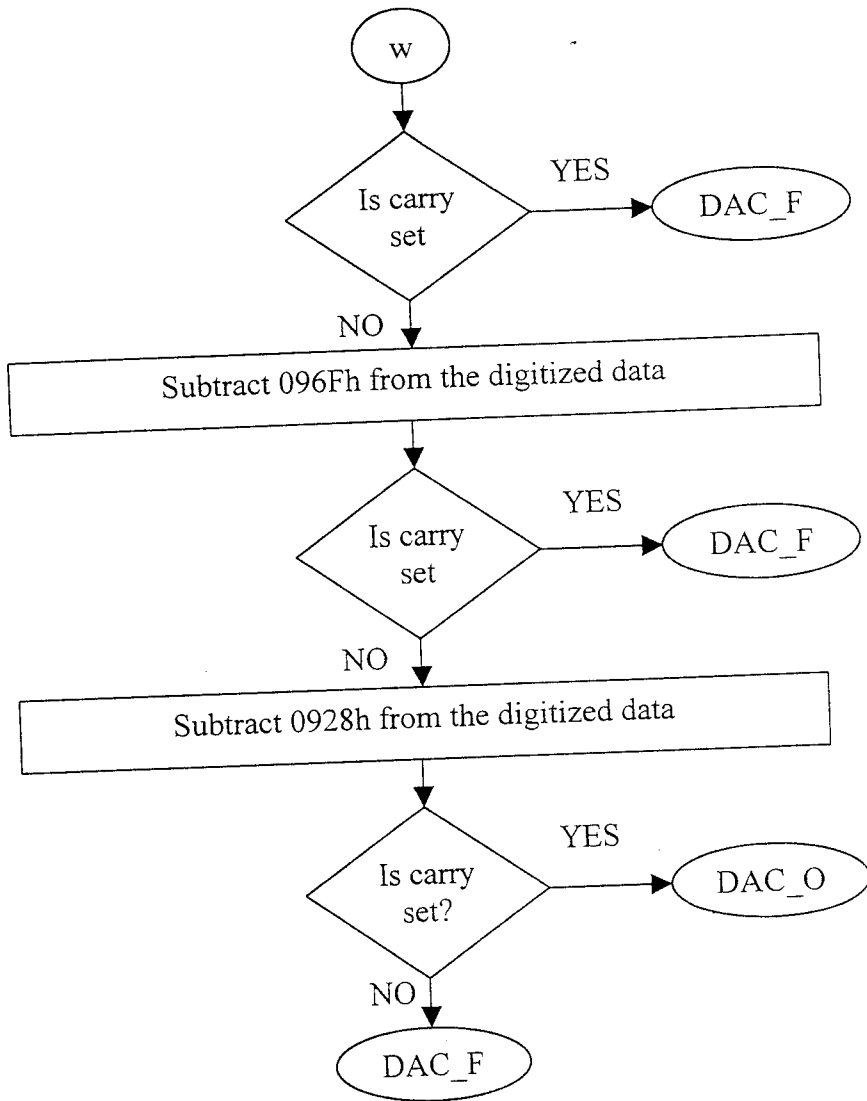


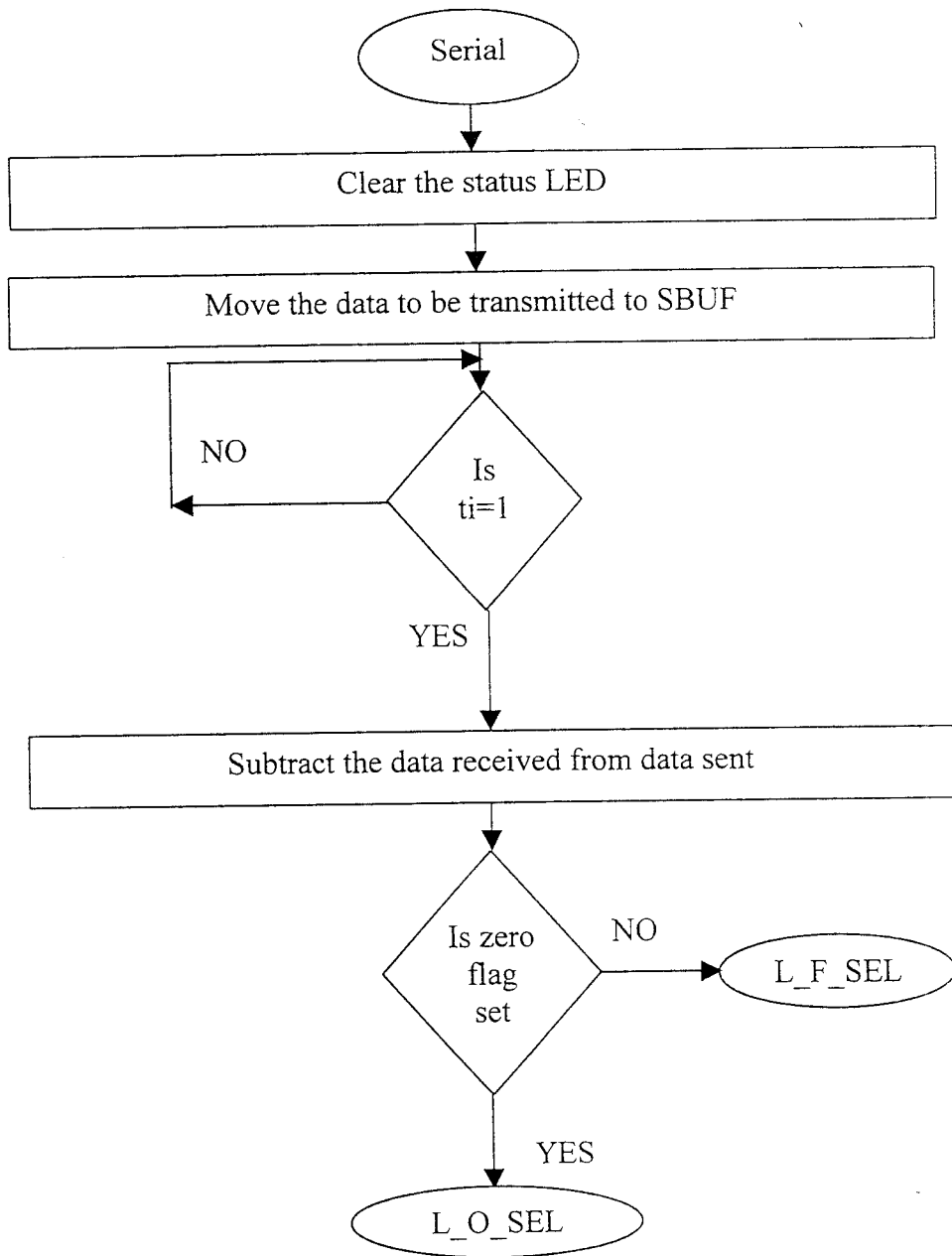












L\_O\_IR

Pairs of LED's glow alternatively

Check  
S17=1  
S18=0  
S19=0

NO

first

YES

L\_F\_IR

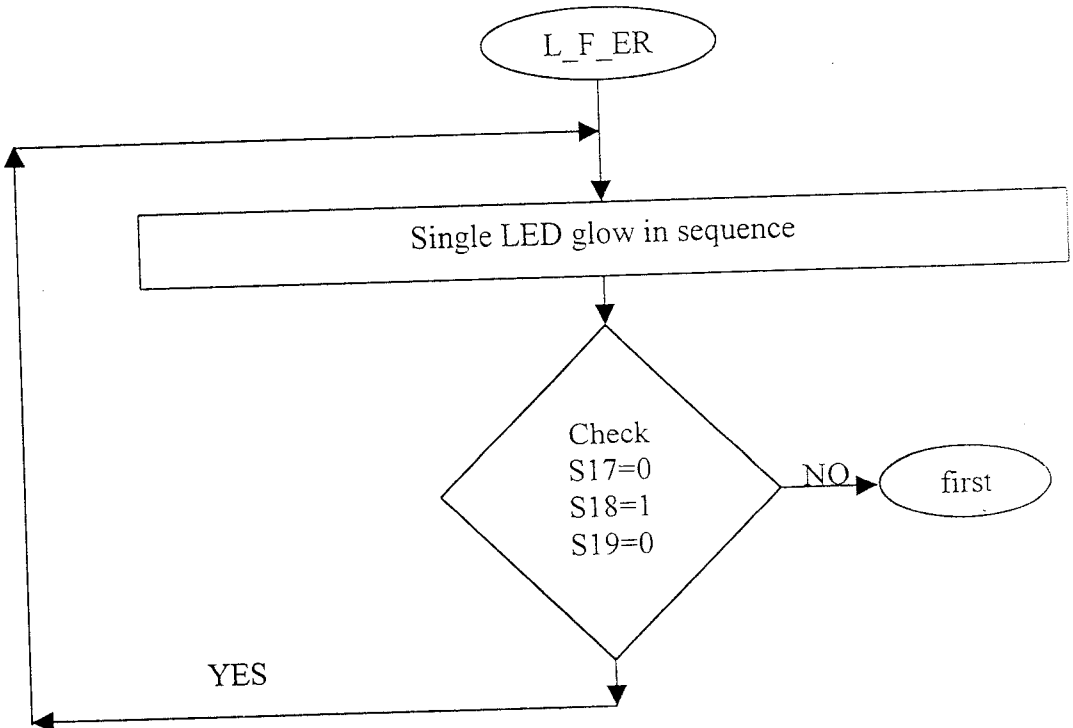
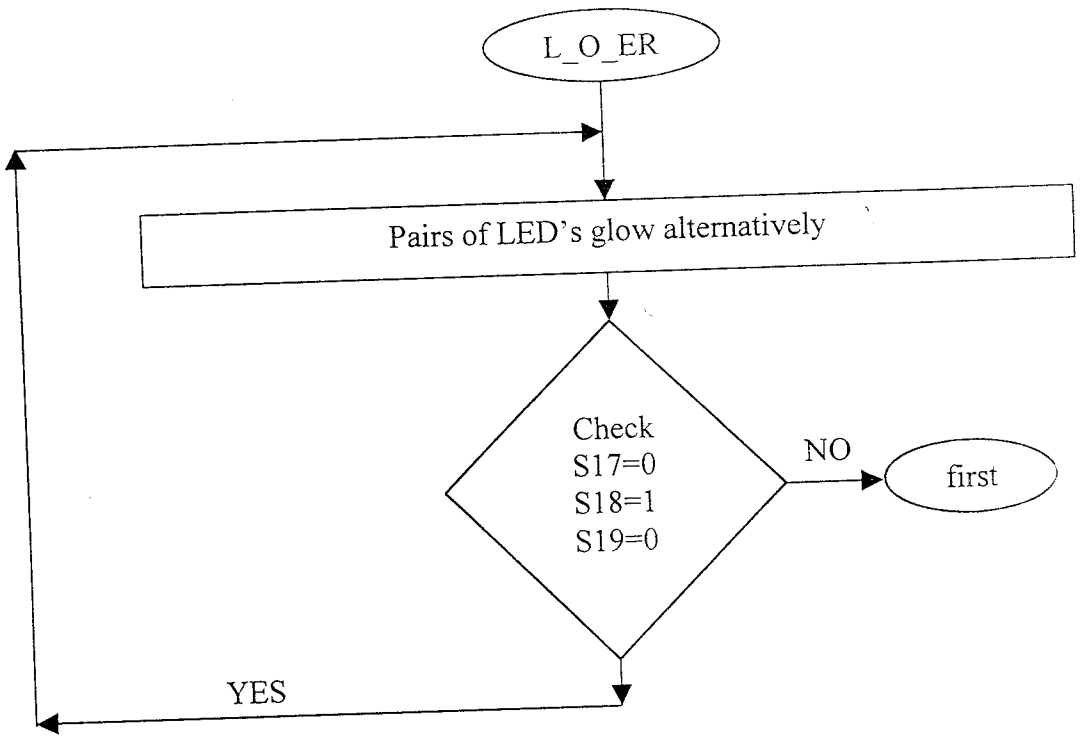
Single LED glow in sequence

Check  
S17=1  
S18=0  
S19=0

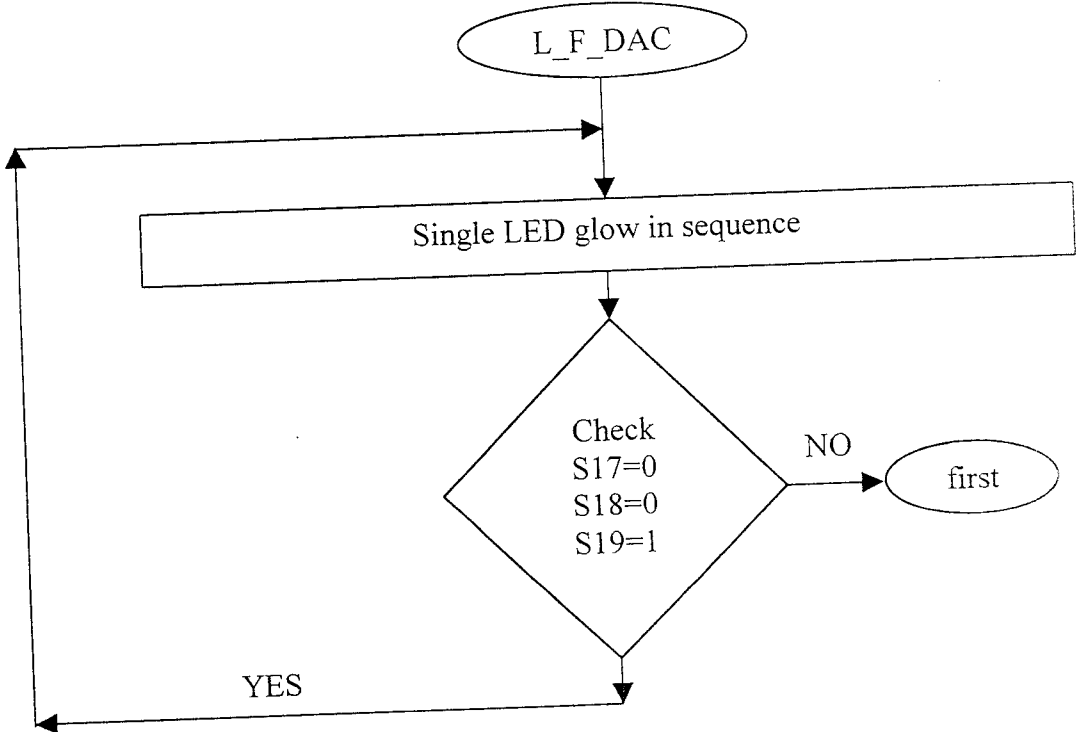
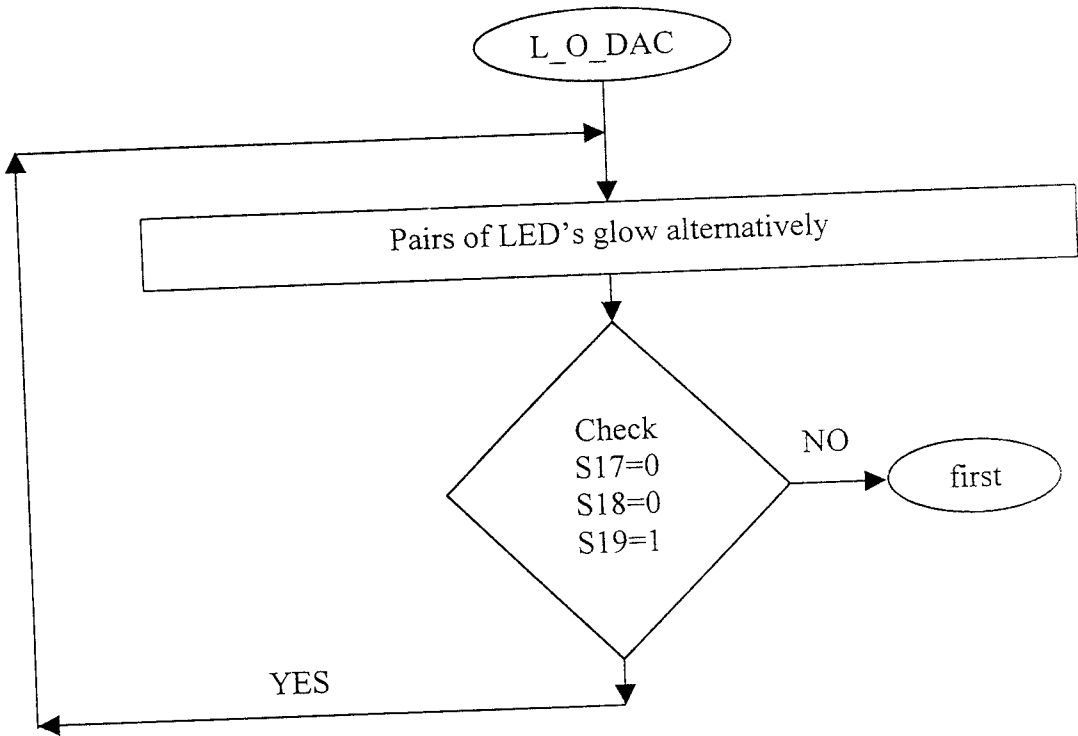
NO

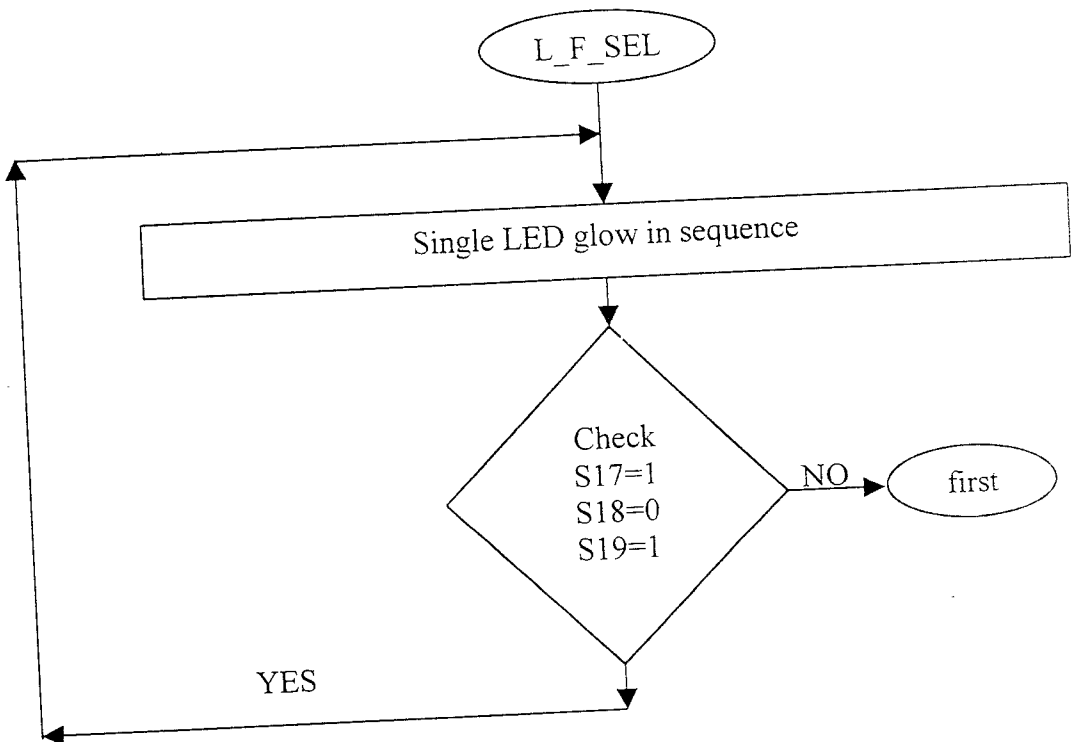
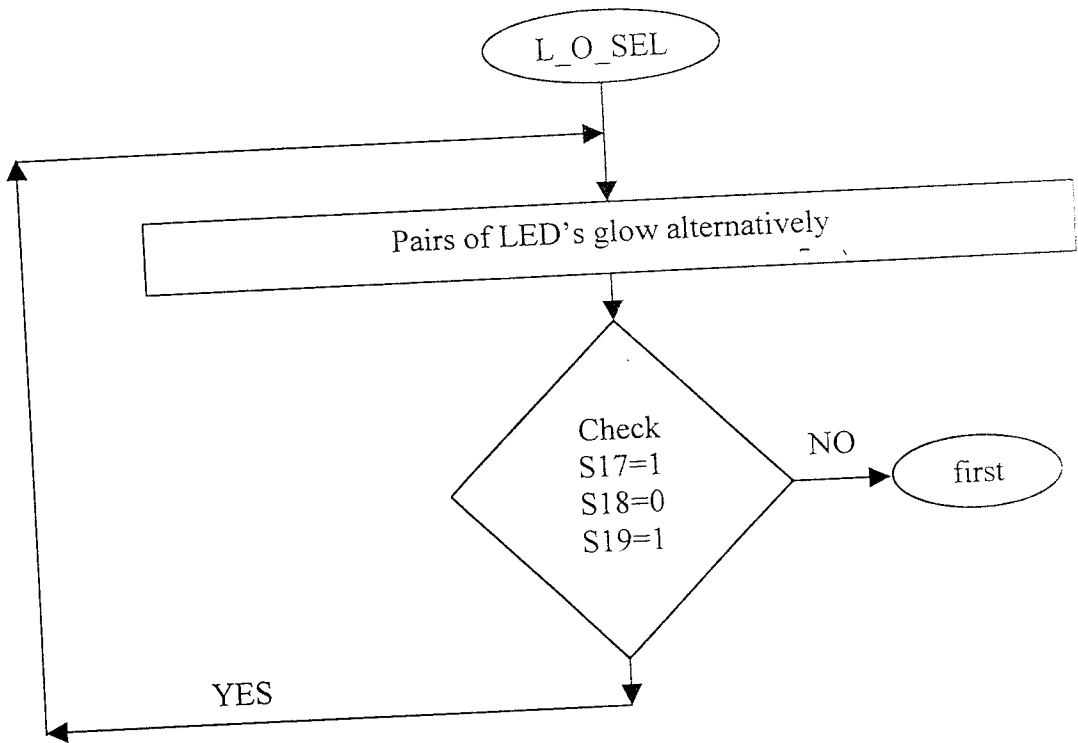
first

YES









```
*****
;*****
;
; HFT9000 TEST JIG PROGRAM
;*****
;*****
```

```
;Microcontroller          - 80C320
;Peripheral interface     - 8155
;RAM                      - 48Z08
;AD Converter             - AD 574
;DA Converter             - TLC 7226
;EPROM                   - 27C512
;*****
```

```
;Digital in ppi_u4
```

```
ppi_u4_cw      equ      0e000h    ;control word
ppi_u4_pa      equ      0e001h    ;port a
ppi_u4_pb      equ      0e002h    ;port b
ppi_u4_pc      equ      0e003h    ;port c
```

```
;Digital out ppi_u5
```

```
ppi_u5_cw      equ      0c000h    ;control word
ppi_u5_pa      equ      0c001h    ;port a
ppi_u5_pb      equ      0c002h    ;port b
```

```
; Internal RAM locations
```

```
intmed_a       equ      038h      ;
intmed_b       equ      039h      ;averaged LSB
intmed_c       equ      03ah      ;averaged MSB
count          equ      03bh      ;for taking average
```

```

data_hi      equ      03ch      ;MSB data
data_lo      equ      03dh      ;LSB data
ckcon       equ      08eh      ;c
conv_c       equ      020h      ;converted data of
intmed_c     equ      021h      ;converted data of
conv_b
intmed_b
db1          bit      20h.0      ;
db2          bit      20h.1      ;
db3          bit      20h.2      ;
db4          bit      20h.3      ;
db5          bit      20h.4      ;
db6          bit      20h.5      ;
db7          bit      20h.6      ;
db8          bit      20h.7      ;bit initializations
dc1          bit      21h.0      ;
dc2          bit      21h.1      ;
dc3          bit      21h.2      ;
dc4          bit      21h.3      ;
dc5          bit      21h.4      ;
dc6          bit      21h.5      ;
dc7          bit      21h.6      ;
dc8          bit      21h.7      ;

```

```

;ADC

```

```

strt_conv    equ      0a000h     ;start conversion for
ADC
lsb_rd       equ      0a001h     ;LSB data read

```

```

;DAC

```

```
dac_1      equ      04000h      ;first dac
dac_2      equ      04001h      ;second dac
dac_3      equ      04002h      ;third dac
dac_4      equ      04003h      ;fourth dac
```

\*\*\*\*\*

```
org      0h
ljmp     start
```

```
org      03h
reti
```

```
org      0bh
reti
```

```
org      013h
reti
```

```
org      1bh
reti
```

```
org      23h
lcall   ser_deci
reti
```

```
org      2bh
reti
```

```
org      03bh
reti
```

```
org      043h
reti
```

```
org    04bh
reti
```

```
org    053h
reti
```

```
org    05bh
reti
```

```
org    0100h
```

```
start:
```

```
mov    sp,#07fh
clr    ea
lcall  key_deb
lcall  init_80c320
lcall  init_u4_ppi
lcall  key_deb
lcall  init_u5_ppi
lcall  key_deb
lcall  key_deb
lcall  key_deb
lcall  delay_1
lcall  delay_1
lcall  clr_ports
lcall  key_deb
ljmp   main_loop
```

```
*****
```

init\_80c320:

mov tmod,#00100001b

mov ckcon,#017h ; #016h ;0001 0010 ; 0001

0110 b

mov scon,#01010000B

mov th1,#0e6H

mov tl1,#0e6H

setb tr1

clr ex1

;setb ea

setb es

ret

\*\*\*\*\*

init\_u4\_ppi:

mov A,#0ch ;port a-->i/p port b-->i/p port

c-->o/p

mov dptr,#ppi\_u4\_cw

movx @dptr,A

ret

init\_u5\_ppi:

mov A,#003h ;port a-->o/p port b-->o/p

mov dptr,#ppi\_u5\_cw

movx @dptr,A

ret

clr\_ports:

mov dptr,#ppi\_u5\_pa ;clears the port-a of u5

mov A,#0ffh

movx @dptr,A

mov dptr,#ppi\_u5\_pb ;clears the port-b of u5

```
movx @dptr,A
ret
```

```
;*****
se_001:      mov dptr,#ppi_u4_pc
```

```
      mov a,#001h
      movx @dptr,a
      ljmp adc_mod
```

```
;*****
se_010:      mov dptr,#ppi_u4_pc
```

```
      mov a,#002h
      movx @dptr,a
      ljmp adc_mod
```

```
;*****
se_011:      mov dptr,#ppi_u4_pc
```

```
      mov a,#003h
      movx @dptr,a
      ljmp adc_mod
```

```
;*****
se_100:      mov dptr,#ppi_u4_pc
```

```
      mov a,#004h
      movx @dptr,a
      ljmp adc_mod
```

```
;*****
se_101:      mov dptr,#ppi_u4_pc
```

```
      mov a,#005h
      movx @dptr,a
      ljmp adc_mod
```

```
;*****
se_110:      mov dptr,#ppi_u4_pc
```

```
      mov a,#006h
      movx @dptr,a
```



```
ljmp adc_mod
```

```
;*****
```

```
se_111:      mov dptr,#ppi_u4_pc
```

```
mov a,#007h
```

```
movx @dptr,a
```

```
ljmp adc_mod
```

```
;*****
```

```
se_000:
```

```
mov dptr,#ppi_u4_pc
```

```
mov a,#000h
```

```
movx @dptr,a
```

```
ljmp adc_mod
```

```
;*****
```

```
adc_mod:
```

```
mov intmed_a,#0h
```

```
mov intmed_b,#0h
```

```
mov intmed_c,#0h
```

```
mov count,#010h
```

```
temp:
```

```
mov data_hi,#0h
```

```
mov data_lo,#0h
```

```
clr a
```

```
mov dptr,#strt_conv
```

```
movx @dptr,a
```

```
mov data_hi,#0h
```

```
mov data_lo,#0h
```

```
nop
```

```
lcall stil_conv
```

```
mov dptr,#strt_conv
```

```
movx a,@dptr
swap a
mov data_hi,a
mov dptr,#lsb_rd
movx a,@dptr
swap a
clr c
```

```
addc a,intmed_a
mov intmed_a,a
mov a,data_hi
addc a,intmed_b
mov intmed_b,a
mov a,#0h
addc a,intmed_c
mov intmed_c,a
djnz count,temp
ljmp dac_mod
```

```
;*****
```

```
se_000xx: ljmp se_000
se_001xx: ljmp se_001
se_010xx: ljmp se_010
se_011xx: ljmp se_011
se_100xx: ljmp se_100
se_101xx: ljmp se_101
se_110xx: ljmp se_110
se_111xx: ljmp se_111
```

```
;*****
```

loop\_100:

setb pl.1

;anlog module check routine

setb pl.4

setb pl.5

start\_2:

lcall clr\_ports

lcall delay\_1

clr a

mov dptr,#ppi\_u4\_pa

movx a,@dptr

mov R4,a

lcall key\_deb

clr a

nop

movx a,@dptr

xrl a,R4

jnz start\_2

mov a,#0ffh

mov r0,a

xrl a,r4

jz se\_clear

mov a,#0feh

mov r0,a

xrl a,R4

jz se\_000xx

mov a,#0fdh

mov r0,a

xrl a,R4

jz se\_001xx

```
mov a,#0fbh
mov r0,a
xrl a,R4
jz se_010xx
```

```
mov a,#0f7h
mov r0,a
xrl a,R4
jz se_011xx
```

```
mov a,#0efh
mov r0,a
xrl a,R4
jz se_100xx
```

```
mov a,#0dfh
mov r0,a
xrl a,R4
jz se_101xx
```

```
mov a,#0bfh
mov r0,a
xrl a,R4
jz se_110xx
```

```
mov a,#07fh
mov r0,a
xrl a,R4
jz se_111xx
```

```
ljmp first
```

```
;*****
```

```
stil_conv:
```

```
    setb p3.2
```

```
    repeat:  nop
```

```
        jnb p3.2,repeat
```

```
    ret
```

```
;*****
```

```
loop_100x:
```

```
    ljmp loop_100
```

```
;*****
```

```
main_loop:
```

```
first:
```

```
    clr a
```

```
        mov a,90h
```

```
        anl a,#00dh
```

```
        xrl a,#000h
```

```
        jz  loop_000xx
```

```
;flashing the leds initially
```

```
    mov a,90h
```

```
        anl a,#00dh
```

```
        xrl a,#001h
```

```
        jz  loop_001
```

```
;internal RAM check routine
```

```
    mov a,90h
```

```
        anl a,#00dh
```

```
        xrl a,#004h
```

```
jz loop_010 ;external RAM check routine
```

```
mov a,90h
```

```
anl a,#00dh
```

```
xrl a,#005h
```

```
jz loop_011x ;digital module check routine
```

```
mov a,90h
```

```
anl a,#00dh
```

```
xrl a,#008h
```

```
jz loop_100x ;analog module check routine
```

```
mov a,90h
```

```
anl a,#00dh
```

```
xrl a,#009h
```

```
jz loop_101x ;serial comn check routine
```

```
ljmp first
```

```
*****  
;*****
```

```
se_clear:
```

```
lcall clr_ports
```

```
ljmp loc_fault_dac
```

```
*****  
;*****
```

```
loop_101x:
```

```
ljmp loop_101
```

```
*****  
;*****
```

```
loop_010:
```

```
setb p1.1 ;external RAM check routine
```

```
setb p1.4
```

```
setb p1.5
```

```
lcall delay_4
```

```
mov dptr,#06000h
mov R5,#000h      ;data
mov R6,#000h      ;count_lo
mov R7,#000h      ;count_hi
```

Lop1:

```
mov a,R5
movx @dptr,a
clr a
nop
nop
nop
movx a,@dptr
clr c
subb a,R5
jnz loc_fault_eramx
inc r5
inc dptr
mov r6,dpl
mov r7,dph
clr c
mov a,r6
subb a,#0ffh
mov a,r7
subb a,#07fh
jc  lop1
jmp loc_ok_eram
```

```
;*****
loop_000xx:  ljmp loop_000x
;*****
```

loop\_001:

```
setb pl.1 ;internal RAM check routine
setb pl.4
setb pl.5
lcall delay_4
mov r2,#0h ;COUNT
mov R1,#020h
mov R5,#000h ;data
```

j1:

```
mov a,r5
mov @R1,a
clr a
nop
mov a,@R1
clr c
subb a,R5
Jnz loc_fault_sram
inc R5
inc R1
inc r2
mov a,#07fh
clr c
subb a,R1
jnz j1
jmp loc_ok_sram
```

```
*****
```

```
start_2x: lcall clr_ports
```

```
ljump start_2
```

```
*****
```



```
mov dptr,#ppi_u4_pa
movx a,@dptr
xrl a,r0
jnz start_2x
ljmp loc_ok_dac
```

```
;*****
```

```
loc_fault_eramx:
```

```
    ljmp loc_fault_eram
```

```
;*****
```

```
loc_fault_dac:
```

```
    clr  p1.5
```

```
    setb p1.4
```

```
    setb p1.1
```

```
    lcall delay_1
```

```
    setb p1.5
```

```
    clr  p1.4
```

```
    setb p1.1
```

```
    lcall delay_1
```

```
    setb p1.5
```

```
    setb p1.4
```

```
    clr  p1.1
```

```
    lcall delay_1
```

```
    mov a,90h
```

```
    anl a,#00dh
```

```
    xrl a,#008h
```

```
    jnz firstx
```

```
    mov dptr,#ppi_u4_pa
```

```
    movx a,@dptr
```

```
    xrl a,r0
```

```
    jnz start_2x
```

```
loop_000x: ljmp loop_000
loop_011x: ljmp loop_011
```

```
;*****
firstx:   lcall clr_ports
          ljmp first
;*****
```

```
loc_ok_sram:
          lcall flash1
          mov a,90h
          anl a,#00dh
          xrl a,#001h
          jz loc_ok_sram
          ljmp first
```

```
;*****
loc_ok_eram:
          lcall flash1
          mov a,90h
          anl a,#00dh
          xrl a,#004h
          jz loc_ok_eram
          ljmp first
```

```
;*****
loc_ok_dac:
          lcall flash1
          mov a,90h
          anl a,#00dh
          xrl a,#008h
          jnz firstx
```

ljmp loc\_fault\_dac

\*\*\*\*\*

loc\_fault\_sram:

```
clr    pl.5
setb   pl.4
setb   pl.1
lcall  delay_1
setb   pl.5
clr    pl.4
setb   pl.1
lcall  delay_1
setb   pl.5
setb   pl.4
clr    pl.1
lcall  delay_1
mov    a,90h
anl    a,#00dh
xrl    a,#001h
jz     loc_fault_sram
ljmp   first
```

\*\*\*\*\*

loc\_fault\_eram:

```
clr    pl.5
setb   pl.4
setb   pl.1
lcall  delay_1
setb   pl.5
clr    pl.4
```

```

setb  p1.1
lcall delay_1
setb  p1.5
setb  p1.4
clr   p1.1
lcall delay_1
mov  a,90h
anl  a,#00dh
xrl  a,#004h
jz   loc_fault_eram
ljmp first

```

```

;*****

```

```

loop_000:                                     ;flashing the LEDs initially

```

```

    lcall flash
    ljmp first

```

```

;*****

```

```

loop_011:                                     ;digital module check

```

```

routine

```

```

    clr  p1.1
    clr  p1.4
    setb p1.5
    mov  R0,#004h

```

```

;count for scanning four times

```

```

start1:

```

```

    mov  dptr,#ppi_u4_pa
    movx a,@dptr
    mov  R4,a
    lcall key_deb
    movx a,@dptr

```

```
xrl a,R4
jnz start1
mov a,R4
mov dptr,#ppi_u5_pa
movx @dptr,a
```

nxt\_port:

```
mov dptr,#ppi_u4_pb
movx a,@dptr
mov R4,a
lcall key_deb
movx a,@dptr
xrl a,R4
Jnz nxt_port
-----
mov dptr,#ppi_u5_pb
mov a,R4
movx @dptr,a
```

updt:

```
djnz r0,start1
ljmp first
```

;\*\*\*\*\*

flash:

```
setb p1.1
setb p1.4
setb p1.5
lcall delay_1
clr p1.1
clr p1.4
clr p1.5
lcall delay_1
ret
```

```
;*****
```

```
flash1:
```

```
    setb    p1.1
    clr     p1.4
    clr     p1.5
    lcall   delay_1
    clr     p1.1
    clr     p1.4
    setb    p1.5
    lcall   delay_1
    clr     p1.5
    clr     p1.1
    setb    p1.4
    lcall   delay_1
    ret
```

```
;*****
```

```
delay_1:
```

```
    mov R4,#007h
```

```
lp3:
```

```
    mov R3,#0ffh
```

```
lp2:
```

```
    mov R2,#0ffh
```

```
lp1:    nop
```

```
    djnz r2,lp1
```

```
    djnz r3,lp2
```

```
    djnz r4,lp3
```

```
    ret
```

```
;*****
```

```
delay_4:
```

```
    mov R4,#02dh           ;equals 45 in decimal
```

```
LP6:
```

```
mov R3,#0ffh
lp5:
mov R2,#0ffh
lp4:  nop
      djnz r2,lp4
      djnz r3,lp5
      djnz r4,lp6
      ret
```

```
;*****
```

```
key_deb:
```

```
      mov     r1,#01h
loop3:
      mov     r2,#0ffh
loop2:
      mov     r3,#0ffh
loop1:
      djnz   r3,loop1
      djnz   r2,loop2
      djnz   r1,loop3
      ret
```

```
;*****
```

```
dac_mod:
```

```
      ljmp  out_8155
dac1:  clr  c
      mov  a,intmed_b
      subb a,#00h
      mov  a,intmed_c
      subb a,#08h
```

```

    jnc comp_a
    ljmp dac_fault
;*****

comp_a:
    clr c
    mov a,intmed_b
    subb a,#06fh
    mov a,intmed_c
    subb a,#009h
    jnc comp_b
    ljmp dac_fault
;*****

comp_b:
    clr c
    mov a,intmed_b
    subb a,#028h
    mov a,intmed_c
    subb a,#00ah
    jc dac_ok
    ljmp dac_fault
;*****

dac_fault:
    ljmp loc_fault_dac

```

```

;;      mov a,#0ffh
;;      mov dptr,#dac_1
;;      movx @dptr,a
;;      mov dptr,#dac_2
;;      movx @dptr,a
;;      mov dptr,#dac_3

```



```
;;      movx @dptr,a
;;      mov dptr,#dac_4
;;      movx @dptr,a
;;      lcall delay_4
;;      ljmp first
;*****
```

```
dac_ok:
    ljmp loc_ok_dac
```

```
;;      mov a,#0h
;;      mov dptr,#dac_1
;;      movx @dptr,a
;;      mov dptr,#dac_2
;;      movx @dptr,a
;;      mov dptr,#dac_3
;;      movx @dptr,a
;;      mov dptr,#dac_4
;;      movx @dptr,a
;;      lcall delay_4
;;      ljmp first
```

;\*\*\*\*\*

```
out_8155:
    mov a,intmed_c
    cpl a
    mov conv_c,#000h
    clr c
    rlc a
    jc b1
    clr db1
```

```
b1x:    rlc a
        jc b2
        clr db2

b2x:    rlc a
        jc b3
        clr db3

b3x:    rlc a
        jc b4
        clr db4

b4x:    rlc a
        jc b5
        clr db5

b5x:    rlc a
        jc b6
        clr db6

b6x:    rlc a
        jc b7
        clr db7

b7x:    rlc a
        jc b8
        clr db8

b8x:    mov a,intmed_b
        cpl a
        mov conv_b,#000h
        clr c
        rlc a
        jc c1
        clr dc1

clx:    rlc a
        jc c2
        clr dc2
```

```
c2x:    rlc a
        jc  c3
        clr dc3
```

```
c3x:    rlc a
        jc  c4
        clr dc4
```

```
c4x:    rlc a
        jc  c5
        clr dc5
```

```
c5x:    rlc a
        jc  c6
        clr dc6
```

```
c6x:    rlc a
        jc  c7
        clr dc7
```

```
c7x:    rlc a
        jc  c8
        clr dc8
```

```
c8x: mov dptr,#ppi_u5_pa
      mov a,conv_c
      movx @dptr,a
```

```
      mov dptr,#ppi_u5_pb
      mov a,conv_b
      movx @dptr,a
      lcall delay_1
      lcall delay_1
      ljmp dacl
```

```
;*****
```

b1: setb db1  
ljmp b1x

b2: setb db2  
ljmp b2x

b3: setb db3  
ljmp b3x

b4: setb db4  
ljmp b4x

b5: setb db5  
ljmp b5x

b6: setb db6  
ljmp b6x

b7: setb db7  
ljmp b7x

b8: setb db8  
ljmp b8x

c1: setb dc1  
ljmp c1x

c2: setb dc2  
ljmp c2x

c3: setb dc3  
ljmp c3x

```
c4:  setb dc4
     ljmp c4x
```

```
c5:  setb dc5
     ljmp c5x
```

```
c6:  setb dc6
     ljmp c6x
```

```
c7:  setb dc7
     ljmp c7x
```

```
c8:  setb dc8
     ljmp c8x
```

```
;*****
```

```
loop_101:
        setb p1.1           ;serial routine checking
        setb p1.4
        setb p1.5
        lcall delay_1
        lcall delay_1
        mov  data_hi,#0
        mov  r1,#0fh       ;COUNT
```

```
s1:
        mov  data_hi,#0
        lcall send_char
;lcall delay_s
        ;mov  a,data_hi
        ;clr  c
```

```

    ;subb  a,r1
    ;jz    slcon
    ;ljmp  loc_fault_serial
;slcon:
    setb  pl.1
    lcall delay_1
    lcall delay_1
    djnz  r1,s1
    ljmp  loc_ok_serial
;*****
send_char:
    mov   sbuf,r1
wait:
    jnb   ti,wait
    clr   ti
    clr   pl.1
    lcall delay_1
    lcall delay_1
    ret
;*****
serial:
;***
    clr   ri
    push  acc
    mov   a,sbuf
    mov   data_hi,a
    pop   acc
    ret
;*****

```

ser\_deci:

;\*\*\*

jnb ri,ser\_rtn\_i

ljmp serial

ser\_rtn\_i:

clr ti

clr ri

ret

;\*\*\*\*\*

delay\_s :

mov r2,#0ffh

s2:

nop

djnz r2,s2

ret

;\*\*\*\*\*

loc\_fault\_serial:

clr p1.5

setb p1.4

setb p1.1

lcall delay\_1

setb p1.5

clr p1.4

setb p1.1

lcall delay\_1

setb p1.5

setb p1.4

clr p1.1

lcall delay\_1

mov a,90h

```
anl a,#00dh
xrl a,#009h
jz loc_fault_serial
ljmp first
```

\*\*\*\*\*

```
loc_ok_serial:
```

```
lcall flash1
mov a,90h
anl a,#00dh
xrl a,#009h
jz loc_ok_serial
ljmp first
```

\*\*\*\*\*

```
end
```



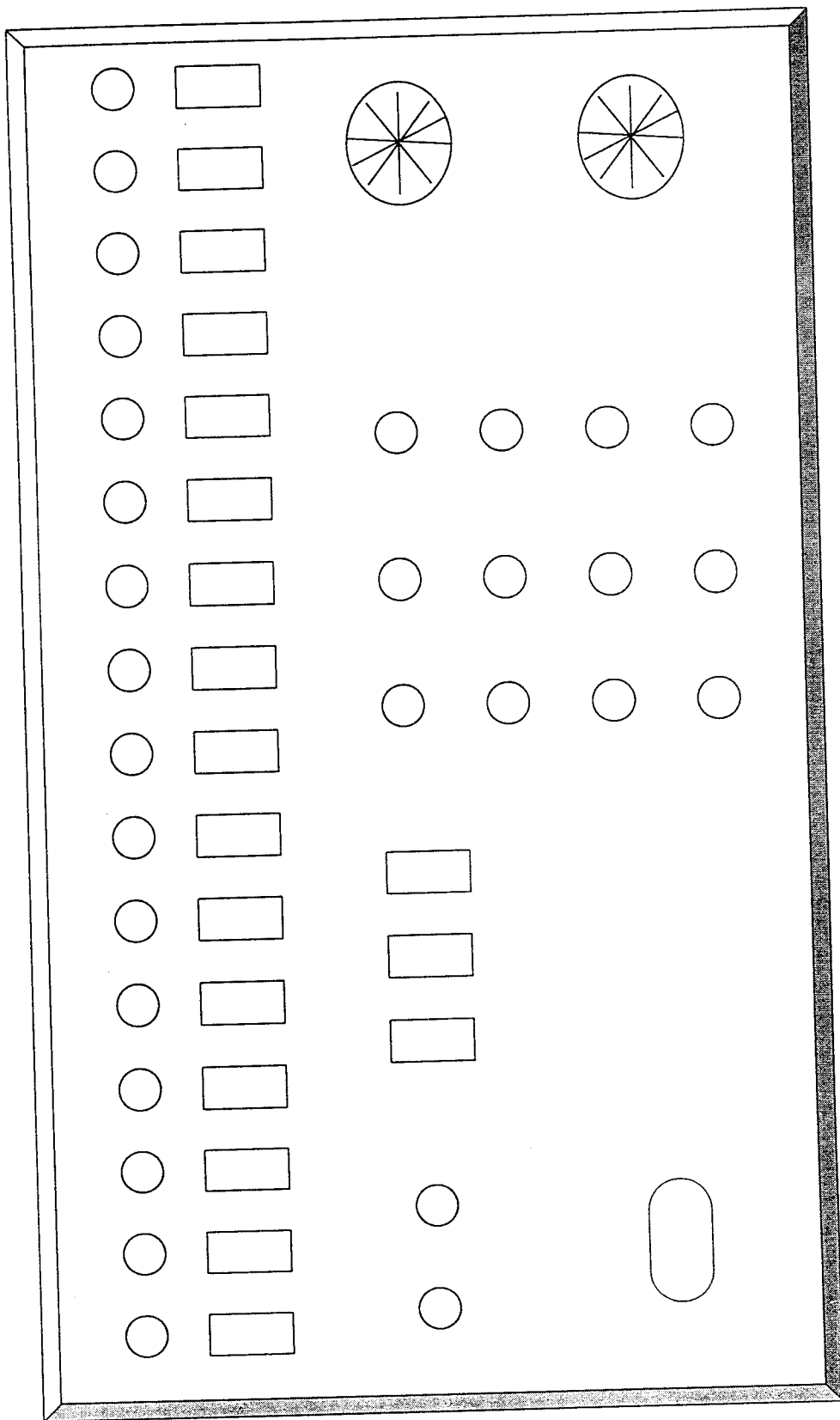
The Central Processing Unit board, which is to be tested using the Test Jig, is the controller of the machine and has a number of peripherals. Each and every line of the PCB has to be tested for its functionality by using the test jig. Different testing methods have been designed and carried out with the help of the specialized hardware and software.

The different modules to be tested are

- Digital module
- Analog Module
- External RAM
- Internal RAM
- Serial Communication

Now our aim is that the operator who is testing the CPU board need not have a deep Knowledge about the board and the peripherals in it but still should be able to operate The test jig. The operator other than comprehending the procedure to operate the jig, should also be able to report which peripheral on the CPU board is at fault when the error is reported on the front panel.

**The detailed view of the front panel is shown in the figure**



## Description of the Display Panel :

- The display panel has 16 Light Emitting Diodes (LED's) on the first row and is named L1 – L16. This representation is for the Digital Outputs.
- The second row has 16 corresponding switches named S1 - S16. This representation is for the Digital Inputs.
- Three switches S17, S18 and S19 are provided. These switches are selection switches, which indicate the module or the peripheral to be tested.
- The two LED's Txd and Rxd, which are provided, indicate the end of transmission and the end of reception.
- We have a rotary switch in order to feed a channel among CH0 – CH7, in the analog module.
- We have a 4\*3 matrix of LED's. The LED's E1, E2 and E3 indicate the OK/ERROR condition while the peripherals are being tested. The fourth LED in the first column is used to indicate the Power on or in other words the functioning of the jig.

**OK condition** – 2 LED's among E1, E2 and E3 glow at a time.

**ERROR condition** - E1, E2 and E3 glow alternately.

- The second and the third column of LED's are used to indicate the DAC outputs. The extra column of green LED's that are indicated is for the future expansion if any.
- A variable Potentiometer is provided in the display board below the variable switch for future development.
- At the back panel, which is provided at the back of the test jig, we have a chord that is to be connected to 230 V AC mains.
- We have the ON/OFF switch and a FUSE provision provided at the back panel.

## Steps to be followed while using the Test Jig:

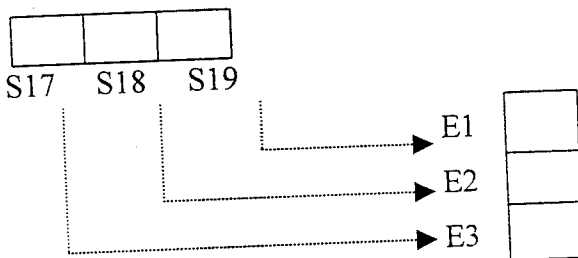
- The chord present in the back panel has to be connected to the 230 Volts AC supply and switched ON.
- Now the power supply LED is ON indicating that the jig has its supply voltage necessary.
- POST – Power On Self Test  
Now as soon as the main switch is ON, the 16 LED's L1 – L16 are checked for their functionality. This is a default routine that has been included in the software. They glow for a second indicating they are OK and in working condition. If in case, any one of the LED's do not glow during the POST, it is an indication that the LED is not Ok by itself and there is no problem with the functionality of the jig. The LED has to be replaced to obtain the result.
- Now after the POST, initially when the S17, S18 and S19 are in the combination 0-0-0, we have the flashing of the LED's. The flashing of the LED's E1, E2 and E3 shows that no module is selected. Again in this case too, we may use this flashing as a means to check for the working of the three LED's, if one of the LED's does not glow while the combination 0-0-0 is selected, then that indicates fault of the LED and not in the functionality of the Jig.
- After the flash routine, we may now select the switches according to the module to be tested. The combinations are indicated below:

S17	S18	S19	Module
0	0	0	Flashing
1	0	0	Internal RAM
0	1	0	External RAM
1	1	0	Digital
0	0	1	Analog
1	0	1	Serial

The various modules to be tested are indicated above. The procedure for testing of the various modules is given henceforth.

➤ **Digital Module :**

1. We first select the combination 1-1-0 in the switches S17, S18 and S19 in order to select the digital testing module.
2. Now the E1, E2 and E3 indicate the selected combination. The orientation of the LED's and the switches can be shown using the figure below:



3. Now that we have selected the digital combination, we may give the Digital Inputs through the switches S1 – S16, and the corresponding outputs is shown in the LED's L1 – L16. If a corresponding LED does not glow when the switch is turned on,

it indicated an error in either the Digital input or the Digital output module. The error can be identified.

➤ **Internal RAM:**

1. The combination for the switches S17, S18 and S19 1-0-0 is selected thereby  
Selecting the testing of the Internal RAM module.
2. Checking of each one of the memory locations does the testing of the Internal RAM, and if an error in a particular location is found, then immediately the check is terminated and the error is displayed. The system will abort from the loop.
3. The E1, E2 and E3 indicate the OK/ERROR condition.

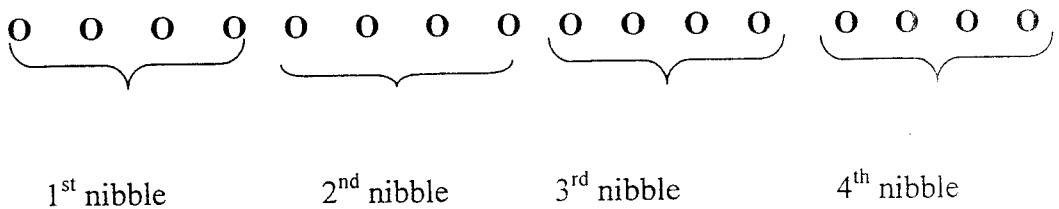
➤ **External RAM:**

1. The combination for the switches S17, S18 and S19 0-1-0 is selected thereby  
Selecting the testing of the External RAM module.
2. Checking of each one of the memory locations does the testing of the External RAM, and if an error in a particular location is found, then immediately the check is terminated and the error is displayed. The system will abort from the loop.
3. The LED's E1, E2 and E3 indicate the OK/ERROR condition.

➤ **Analog Module:**

1. The combination of S17, S18 and S19 switches 0-0-1 is selected thereby  
Selecting the testing of the Analog module.
2. Select the Channel 0 – Channel 7 in the Rotary switch provided.
3. The corresponding channel 0 to channel 7 should be selected in the switches S1 – S8. It should be noted that the selection of the channel in the Rotary switch and the selection of the channel in S1-S8 switches should be done simultaneously, if either one is not selected at a point of time or if there is a delay in the selection of either switches, the LED's E1, E2 and E3 will indicate an ERROR condition until the

switches are selected. Care should be taken that the selection of switches is done attentively in order to avoid the error shown in the LED's E1, E2 and E3 mistaken for the error in the CPU board. During the testing of the Analog module, we display the digitized 12-bit value that is the output of the ADC in the 16 LED's that are provided in the Display board. We have split the representation of the digitized value into 4, each representing one digit between the 16 LED'S. So in the display that you see in L1 – L16 using the 8421 weighted codes, we may decode the value.



4. During the testing of the channels 0 –7, it is not necessary to RESET to the Initial condition in order to test a subsequent channel. The software written Is compatible to test the channels online without resetting each time.

*HANDOUT*

---



## 80C320 CPU DEBUGGER

Default check : Power On Self Test (POST)

Testing Procedure :

- Select the switches S17, S18 and S19 to test required module

S17	S18	S19	Module
0	0	0	Flash
1	0	0	Internal RAM
0	1	0	External RAM
1	1	0	Digital
0	0	1	Analog
1	0	1	Serial Comm

- For all routines status indicated by E1, E2 and E3 LED's

OK condition - 2 LED's glow at a time

ERROR condition- LED's glow alternately one at a time

- **Digital Module** :
  1. Select switches S17, S18 and S19.
  2. Check for **Digital OUT** LED's **L1-L16** by Corresponding **Digital IN's** given through switches **S1-S16**.
- **Analog Module** :
  1. Select switches S17, S18 and S19.
  2. Select Rotary switch to required channel to be Tested ranging from **CH0-CH7**.
  3. Select the switches **S1-S8** for **CH0- CH7**.
  4. Check E1, E2 and E3 for Ok/Error condition.

During the checking of the Analog Module, we also display the corresponding Digitized value of the channel being tested which is displayed in the LED's **L1-L16**.

The Digitized values:

Analog values	Digitized values
1.7 volts	095C h
2.2 volts	09C0 h
2.7 volts	0A26 h

There were a number of hurdles we had to cross to complete this project. We learnt that in the practical field, we had to spend hours trying to identify the problem and then device means to solve it. A few of the significant ones we have mentioned below:

- During the testing of the Power Supply board we fabricated, we encountered a short between a screw in the heat sink and the To3 package. We were not getting the required voltage because of this, it took some time to identify the cause and then we placed a non-conducting material between the heat sinks and the IC's.
- Initially during the planning phase of the fabrication of PCB's, we started out as making the PCB's through Auto Routing and Using the Photo- Filming technique. It was taking a lot of time and then we thought of the alternative of drawing it manually and etching it.
- During the testing phase of our software, we noticed that our Digital Module was not working. None of the 16 outputs were reflecting in the LED's. We were constantly checking and modifying our software, but after a detailed study we found that there was a connection missing in the CPU board that they had given us to test. When we included a short in the CPU board, the Digital module worked.
- During the wiring phase, we noticed that we had shorts in the circuits to a very large extent when we had completed around 10% of the wiring. We rechecked the entire connections and found no fault. We then noticed that we had not used a sleeve at the connecting points.
- While writing the software, the delay calculations that we had done were not practically getting implemented. For example if we have set a 1 ms delay, while testing the Jig we would not obtain this delay. So according to the practical requirement, we changed the delay by trial and error method.

- (1) The parallel checking of the modules is not supported. (i.e) not more than one of the defined modules can be checked simultaneously.
- (2) The analog module to be tested has to be represented by selecting a digital switch of port A.
- (3) In the serial communication module it is not possible to test both the transmission and reception of data without PC link. Hence the serial communication module can be completed only after the system is made PC enabled.
- (4) In each of the modules, only error conditions can be detected and the type of error cannot be spotted.

For example an error in digital module may be any of the following kind.

- Total port failure
- Any discontinuity in the PCB route lining
- Failure of the decoder to produce chip select signals so that the chip itself may not get the signals selected.
- In both the internal and external memory modules the damaged memory locations cannot be identified. This is because the memories check loop is aborted once a damaged location is encountered.
- The analog channel can be fed with a fixed voltage only and it cannot be varied.

The system can be interfaced with the PC by connecting the serial port of the test jig with any serial port of the PC (LPJ ports). The selection of different modules can be given as commands. For this affront end (GUI) with necessary options has to be designed. Hence as the commands are given, each command is interpreted into the signals with specific standards and transmitted serially out of the PC through the serial port to the jig. Therefore the jig firmware has to sense the given command and switch to that particular module.

Once the testing of the module is completed inside the jig, the results or the status of the peripherals tested has to be transmitted back to the PC.

Hence all the operations of the jig can be made PC enabled through the serial link. Once the jig system is PC enabled, the following features can be added.

- (1) Readable text messages about the status information can be displayed in the PC instead of LED indications.
- (2) During the checking of a module comprising a few peripheral IC's, the operation of each peripheral unit can be tracked and the online information / status about each unit can be displayed as the process dynamically goes on.
- (3) While checking of the memory module (both internal and external), the check loop is made to scan through the entire range of memory locations and at the end of the check routine, in case of error, a list of damaged locations may be displayed, so that, if possible the user may use the rest of the memory locations which are not damaged.
- (4) The analog module can be tested for different levels of analog voltage inputs.

The project “80C320 CPU Debugger” was a success. The working model of the Test Jig is being used at Premier Polytronics Limited, Coimbatore for real-time applications. The controller board for which the Test Jig has been designed is being used in many products.

This project covers the testing of all the peripherals of the Controller board with inputs provided through switches and the defects, if any, are displayed by means of LED's. This Test Jig has considerably reduced the time involved in tracking errors in the peripherals of the controller board for smooth operation of the machines.

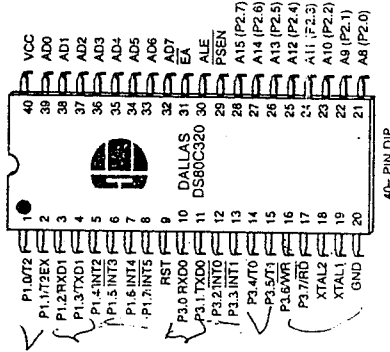
This project can be extended to have a PC interface making the system more user-friendly and effective. The end result of this PC link can be the online tracking of the PCB's in the various customer areas thereby facilitating online checking.

1. "High- Speed Micro controller Data Book", Dallas Semiconductors Corporation, 1995 edition.
2. "Linear Applications Data book", National Semiconductors Corporation, 2<sup>nd</sup> edition.
3. "Power Supply Circuits, Data Book", Texas Instruments, 1996 edition.
4. "Industry Standard Analog IC's", SGS Thompson, 1<sup>st</sup> Edition, June 1989.
5. "Micro controllers Data Book", Amtel Corporation, 3<sup>rd</sup> edition, May 1997.
6. "Embedded Applications, Volume 2", Intel Corporation, 1995 edition.
7. "Microprocessor and its Applications", Ramesh Goenkar , 1996 edition.
8. "Linear Integrated Circuits and its Applications", Roy Chowdhery, 1995 edition.

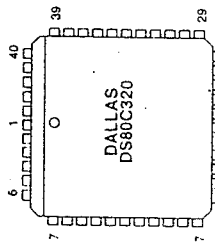
**FEATURES**

- 80C32—Compatible
- Pin-compatible
- Standard 8051 instruction set
- Four 8-bit I/O ports
- Three 16-bit timer/counters
- 256 bytes scratchpad RAM
- Multiplexed address/data bus
- Addresses 64KB ROM and 64KB RAM
- High-speed architecture
- 4 clocks/machine cycle (8032=12)
- Wasted cycles removed
- Runs DC to 33 MHz clock rates
- Single-cycle instruction in 121 ns
- Uses less power for equivalent work
- Dual data pointer
- Optional variable length MOVX to access fast slow RAM/peripherals
- High integration controller includes:
  - Power-fail reset
  - Programmable Watchdog timer
  - Early-warning power-fail interrupt
- Two full-duplex hardware serial ports
- 13 total interrupt sources with six external
- Available in 40-pin DIP, 44-pin PLCC and TOFP

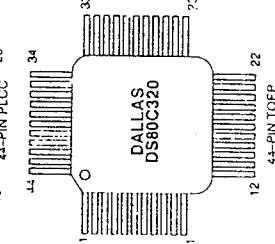
**PIN ASSIGNMENT**



40-PIN DIP



44-PIN PLCC



44-PIN TOFP

**DESCRIPTION**

The DS80C320 is a fast 80C31/80C32-compatible microcontroller. Wasted clock and memory cycles have been removed using a redesigned processor core. As a result, every 8051 instruction is executed between 1.5 and 3 times faster than the original for the same crystal speed. Typical applications will see a speed improvement of 2.5 times using the same code and same crystal. The DS80C320 offers a maximum crystal rate of 33 MHz, resulting in apparent execution speeds of 82.5 MHz (approximately 2.5X).

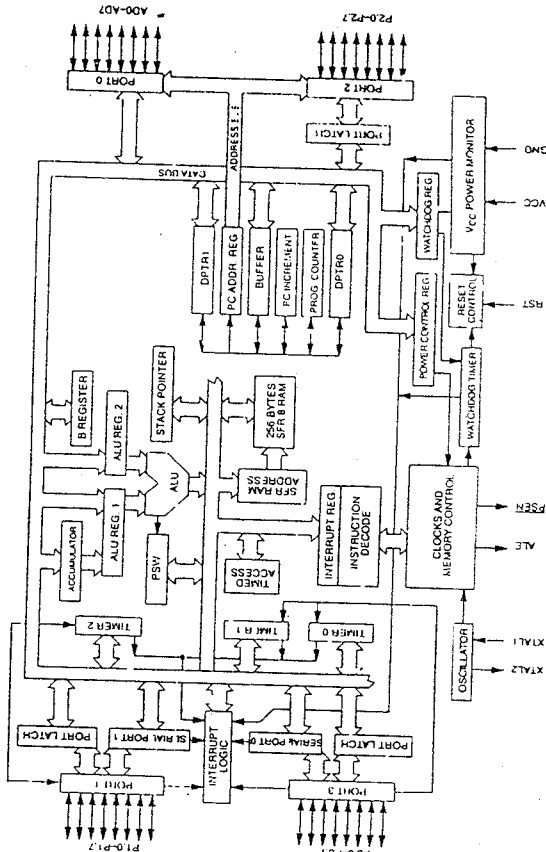
serial port, seven additional interrupts, programmable watchdog timer, power-fail interrupt; and reset. The DS80C320 also provides dual data pointers (DPTs) to speed block data memory moves. It can also adjust the speed of on-chip data memory access to between two and nine machine cycles for flexibility in selecting memory and peripherals.

The DS60C320 provides several extras in addition to greater speed. These include a second full hardware

**ORDERING INFORMATION**

PART NUMBER	PACKAGE	MAX CLOCK SPEED	TEMPERATURE RANGE
DS80C320-MCG	40-pin plastic DIP	25 MHz	0°C to +70°C
DS80C320-QCG	14-pin PLCC	25 MHz	0°C to +70°C
DS80C320-ECG	44-pin TOFP	25 MHz	0°C to +70°C
DS80C320-MNG	40-pin plastic DIP	25 MHz	-40°C to +85°C
DS80C320-QNG	44-pin PLCC	25 MHz	-40°C to +85°C
DS80C320-ENG	44-pin TOFP	25 MHz	-40°C to +85°C
DS80C320-MCL	40-pin plastic DIP	33 MHz	0°C to +70°C
DS80C320-QCL	44-pin PLCC	33 MHz	0°C to +70°C
DS80C320-ECL	44-pin TOFP	33 MHz	0°C to +70°C
DS80C320-MNL	40-pin plastic DIP	33 MHz	-40°C to +85°C
DS80C320-QNL	44-pin PLCC	33 MHz	-40°C to +85°C
DS80C320-ENL	44-pin TOFP	33 MHz	-40°C to +85°C

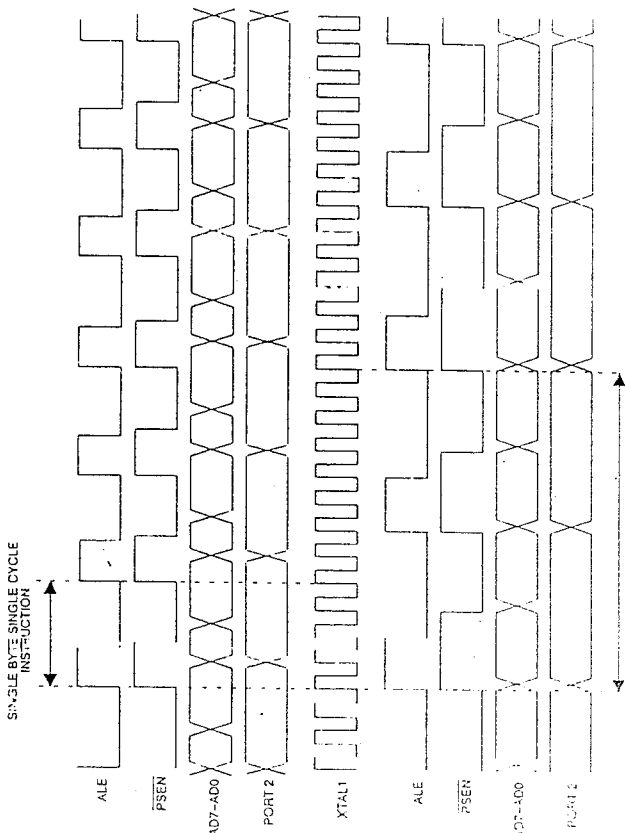
DS80C320 BLOCK DIAGRAM Figure 1



limiting diagrams are provided below in the electrical specifications.

COMPARATIVE TIMING OF THE DS80C320 AND 80C32 Figure 2

DS80C320 TIMING



STANDARD 80C32 TIMING

efficient design.

In this updated core, dummy memory cycles have been eliminated. In a conventional 80C32, machine cycles are generated by dividing the clock frequency by 12. In the DS80C320, the same machine cycle is performed in 4 clocks. Thus the fastest instruction, 1 machine cycle, is executed three times faster for the same crystal frequency. Note that these are identical instructions. A comparison of the timing differences is shown in Figure 2. The majority of instructions on the DS80C320 will see the full 3 to 1 speed improvement. Some instructions will get between 1.5 and 2.4 X improvement. Note that all instructions are faster than the original 80C32. Table 2 below shows a summary of the instruction set including the speed.

The numerical average of all opcodes is approximately a 2.5 to 1 speed improvement. Individual programs will be affected differently, depending on the actual instructions used. Speed sensitive applications would make the most use of instructions that are three times faster. However, the sheer number of 3 to 1 improved opcodes makes dramatic speed improvements likely for any code. When these architecture improvements are combined with 0.8  $\mu$ m CMOS, the result is a single cycle instruction execution in 160 ns. The Dual Data Pointer feature also allows the user to eliminate wasted instructions when moving blocks of memory.

INSTRUCTION SET SUMMARY

All instructions in the DS80C320 perform the same functions as their 80C32 counterparts. Their affect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute and relative number of clocks.

For absolute timing of real-time events, the timing of software loops will need to be calculated using the table

run at 4 clocks per increment cycle to take advantage of higher speed operation.

The relative time of two instructions might be different in the new architecture than it was previously. For example, in the original architecture, the "MOVX A, @DPTR" instruction and the "MOV direct, direct" instruction used two machine cycles or 24 oscillator cycles. Therefore, they required the same amount of time. In the DS80C320, the MOVX instruction can be done in two machine cycles or eight oscillator cycles but the "MOV direct, direct" uses three machine cycles or 12 oscillator cycles. While both are faster than their original counterparts, they now have different execution times from each other. This is because in most cases, the DS80C320 uses one cycle for each byte. The user concerned with precise program timing should examine the timing of each instruction for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV.

INSTRUCTION SET SUMMARY Table 2

Legends:

A	-	Accumulator
Rn	-	Register R7-R0
direct	-	Internal Register address
@Ri	-	Internal Register pointed-to by R0 or R1 (except MOVX)
rel	-	2's complement offset byte
bit	-	direct bit-address
#data	-	8-bit constant
#data 16	-	16-bit constant
addr 16	-	16-bit destination address
addr 11	-	11-bit destination address



12	ANL C, bit	2	8	2	8
	ORL C, bit	1	4	2	8
	ORL C, bit	2	8	2	8
	ORL C, bit	1	4	2	8
	MOV C, bit	2	8	2	8
	MOV bit, C	2	8	2	8
	Program Branching				
	Instructions:				
	ACALL addr 11	2	12		
	LCALL addr 16	3	16		
	RET	1	16		
	RETI	1	16		
	AJMP addr 11	2	12		
	LJMP addr 16	3	16		
	SJMP rel	2	12		
	JMP @A+DPTR	1	12		
	JZ rel	2	12		
	JNZ rel	2	12		
	DJNZ Rn, rel	2	12		
	DJNZ direct, rel	3	16		

2	INC Rn	1	4		
1	INC direct	2	8		
2	INC @Ri	1	4		
1	INC DPTH	1	12		
2	DEC A	1	4		
1	DEC Rn	1	4		
2	DEC direct	2	8		
1	DEC CF:	1	4		
2	MUL AB	1	20		
1	DIV AB	1	20		
2	DIV A	1	4		
	Logical Instructions:				
1	RL A, Rn	1	4		
2	RL A, direct	2	8		
1	RL A, @Ri	1	4		
2	RL A, #data	2	8		
3	RL direct, A	2	8		
2	RL direct, #data	3	12		
1	CLR A, Rn	1	4		
2	CLR A, direct	2	8		
1	RL A, @Ri	1	4		
2	RL A, #data	2	8		
2	RL direct, A	2	8		
3	RL direct, #data	3	12		
	Data Transfer Instructions:				
1	MOV A, Rn	1	4		
2	MOV A, direct	2	8		
1	MOV A, @Ri	1	4		
2	MOV A, #data	2	8-36*		
1	MOV Rn, A	1	4		
2	MOV Rn, direct	2	8-36*		
3	MOV Rn, DPTR, A	3	8-36*		
2	PUSH direct	2	8		
3	POP direct	3	8		
2	XCH A, Rn	2	8		
3	XCH A, direct	3	8		
2	XCH A, @Ri	2	8		
3	XCHD A, @Ri	3	8		
1	V @Ri, A	1	4		
2	V @Ri, direct	2	4		
3	V @Ri, #data	3	12		
3	V DPTR, #data 16	3	12		

The table above shows the speed for each class of instruction. Note that many of the instructions have multiple opcodes. There are 255 opcodes for 111 instructions. Of the 255 opcodes, 159 are three times faster than the original 80C32. While a system that emphasizes those instructions will see the most improvement, the large total number that receive a 3 to 1 improvement, assure a dramatic speed increase for any system. The speed improvement summary is provided below.

SPEED ADVANTAGE SUMMARY	
# of opcodes	Speed Improvement
159	3.0 x
51	1.5 x
43	2.0 x
2	2.4 x
255	Average: 2.5

### MEMORY ACCESS

The DS80C320 contains no on-chip ROM and 256 bytes of scratchpad RAM. Off-chip memory is accessed using the multiplexed address/data bus on P0 and the MSB address on P2. A typical memory connection is shown in Figure 3. Timing diagrams are provided in the Electrical Specifications. Program memory (ROM) is accessed at a fixed rate determined by the crystal frequency and the actual instructions. As mentioned above, an instruction cycle requires four clocks. Data memory (RAM) is accessed according to a variable speed MOVX instruction as described below.

3. I/O devices or registers
4. Control logic
5. Chip Select logic
6. Bidirectional data bus
7. Handshake signals and Interrupt logic

A programmable I/O device is programmed by writing a specific word, called the control word, according to the internal logic; its status can be verified by reading the status register. This I/O device can be expanded to include elements such as multiple I/O ports, counters, and parallel-to-serial registers. The programmable devices used in the Intel SKD-85 system—the 8155/8156, the 8355 (or 8755), and the 8279—are described in detail in the next sections.

## 14.2 THE 8155/8156 AND 8355/8755 MULTIPURPOSE PROGRAMMABLE DEVICES

The 8155 and the 8355 are two multipurpose programmable devices specifically designed to be compatible with the 8085 microprocessor. The ALE,  $\overline{\text{IO/M}}$ ,  $\overline{\text{RD}}$ , and  $\overline{\text{WR}}$  signals from the 8085 can be connected directly to these devices; this eliminates the need for external demultiplexing of the low-order bus  $\text{AD}_7\text{--AD}_0$  and generation of the control signals such as  $\overline{\text{MEMR}}$ ,  $\overline{\text{MEMW}}$ ,  $\overline{\text{IOR}}$ , and  $\overline{\text{IOW}}$ .

The 8155 includes 256 bytes of R/W memory, three I/O ports, and a timer. The 8156 is identical with the 8155, except that the 8156 requires Chip Enable (CE) active high. The 8355 includes 2K of ROM and two I/O ports. The 8755 is similar to the 8355, except that the 8755 is EPROM. The programmable I/O sections of these devices are illustrated in the following sections.

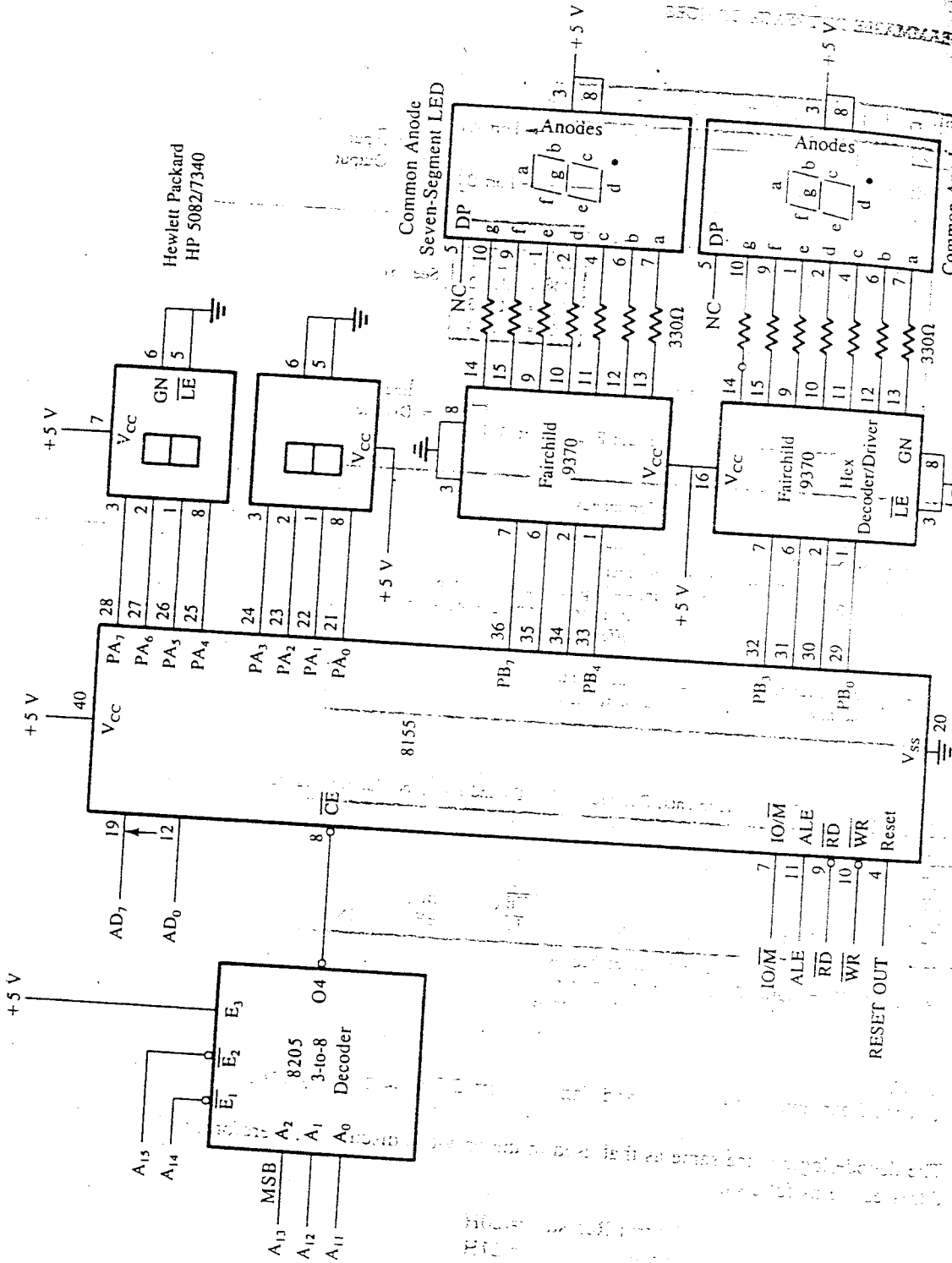
### 14.2.1 THE 8155/8156 PROGRAMMABLE I/O PORTS AND TIMER

The 8155/8156 is a device with two sections: the first is 256 bytes of R/W memory, and the second is a programmable I/O. Functionally, these two sections can be viewed as two independent chips. The I/O section includes two 8-bit parallel I/O ports (A and B), one 6-bit port (C), and a timer (Figure 14.5). All the ports can be configured simply as input/output ports. Ports A and B also can be programmed in the handshake mode, each port using three signals as handshake signals from port C. The timer is a 14-bit down-counter and has four modes. Pins PA, PB, and PC, shown in Figure 14.5, correspond to ports A, B, and C.

#### CONTROL LOGIC

The control logic of the 8155 is specifically designed to eliminate the need for externally demultiplexing lines  $\text{AD}_7\text{--AD}_0$  and generating separate control signals for memory and I/O. Figure 14.5 shows five control signals; all except the Chip Enable ( $\overline{\text{CE}}$ ) are input signals directly generated by the 8085.

- $\overline{\text{CE}}$ —Chip Enable: This is a master Chip Select signal connected to the decoded high-order bus.



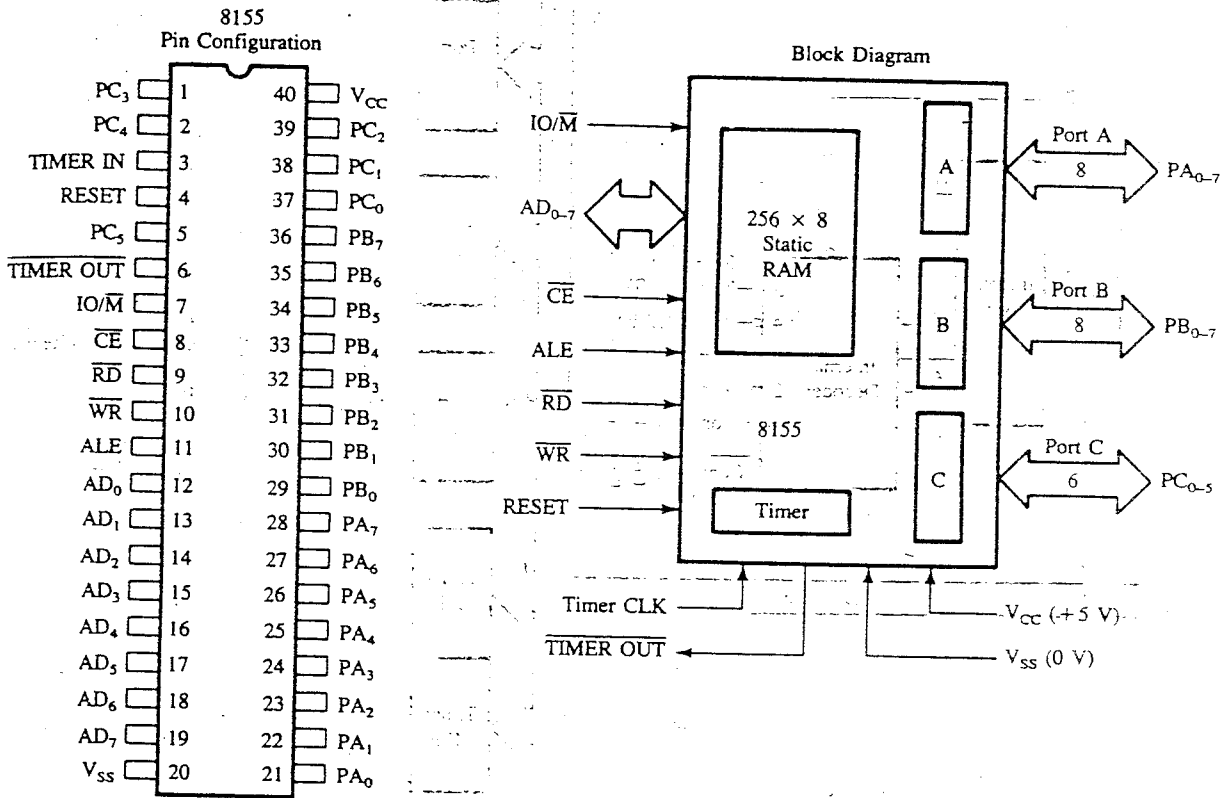


FIGURE 14.5

8155 Pin Configuration and Block Diagram

SOURCE: Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), p. 1-31.

- IO/M—When this signal is low, the memory section is selected, and when it is high, the I/O section (including timer) is selected.
- ALE—Address Latch Enable: This signal latches the low-order address AD<sub>7</sub>–AD<sub>0</sub>, CE, and IO/M into the chip.
- RD and WR—These are control signals to read from and write into the chip registers and memory.
- RESET—This is connected to RESET OUT of the 8085 and this resets the chip and initializes I/O ports as input.

**THE 8155 I/O PORTS**

The I/O section of the 8155 includes a control register, three I/O ports, and two registers for the timer (Figure 14.6). The expanded block diagram of the I/O section (Figure 14.6) represents a typical programmable I/O, as discussed in Section 14.12. In that section, two address lines plus the Chip Select logic were used to determine port addresses. The 8155 I/O section requires three address lines—AD<sub>2</sub> to AD<sub>0</sub> (A<sub>2</sub>–A<sub>0</sub> internally)—and the Chip

# Quad of Independently Functioning Comparators

## INTRODUCTION

The LM139/LM239/LM339 family of devices is a monolithic independently functioning comparators designed to meet the needs for a medium speed, TTL compatible comparator for industrial applications. Since no antisaturation are used on the output such as a Baker clamp or diode circuitry, the output leakage current in the OFF state is typically 0.5 nA. This makes the device ideal for applications where it is desired to switch a node to a voltage level while leaving it totally unaffected in the OFF state. Features include single supply, low voltage operation input common mode range from ground up to approximately one volt below  $V_{CC}$ . The output is an uncommitted so it may be used with a pull-up resistor and a output supply to give switching levels from any voltage up to 36V down to a VCE SAT above ground (approx. sinking currents up to 15 mA. In addition it may be connected as a single pole switch to ground, leaving the switched output in the OFF state. Power dissipation for comparators in the OFF state is typically 4 mW on a single 5V supply (1 mW/comparator).

## DESCRIPTION

The basic input stage of one of the four comparators of the LM139, Transistors  $Q_1$  through  $Q_4$  make a Darlington differential input stage with  $Q_3$  and  $Q_5$  give single-ended output from differential input in gain. Any differential input at  $Q_1$  and  $Q_4$  will cause  $Q_3$  to switch OFF or ON depending

on input signal polarity. It can easily be seen that operation with an input common mode voltage of ground is possible. With both inputs at ground potential, the emitters of  $Q_1$  and  $Q_4$  will be at one  $V_{BE}$  above ground and the emitters of  $Q_2$  and  $Q_3$  at 2  $V_{BE}$ . For switching action the bases of  $Q_3$  and  $Q_2$  need only go to one  $V_{BE}$  above ground and since  $Q_2$  and  $Q_3$  can operate with zero volts collector to base, enough voltage is present at a zero volt common mode input to insure comparator action. The bases should not be taken more than several hundred millivolts below ground, however, to prevent forward biasing a substrate diode which would stop all comparator action and possibly damage the device, if very large input currents were provided.

Figure 2 shows the comparator with the output stage added. Additional voltage gain is taken through  $Q_7$  and  $Q_8$  with the collector of  $Q_8$  left open to offer a wide variety of possible applications. The addition of a large pull-up resistor to the collector of  $Q_8$  to either  $+V_{CC}$  or any other supply up to 36V both increases the LM139 gain and makes possible output switching levels to match practically any application. Several outputs may be tied together to provide an OR'ing function or the pull-up resistor may be omitted entirely with the comparator then serving as a SPST switch to ground.

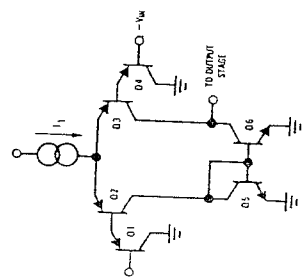


FIGURE 1. Basic LM139 Input Stage

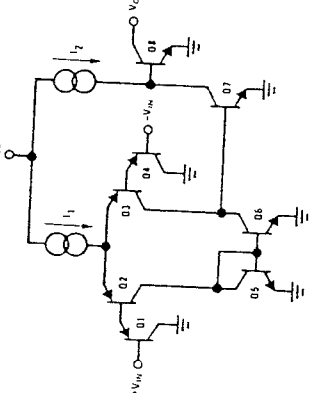


FIGURE 2. Basic LM139 Comparator

Output transistor  $Q_8$  will sink up to 15 mA before the output ON voltage rises above several hundred millivolts. The output current sink capability may be boosted by the addition of a discrete transistor at the output.

shown in Figure 3. Current sources  $I_3$  and  $I_4$  are added to help charge any parasitic capacitance at the emitters of  $Q_1$  and  $Q_4$  to improve the slew rate of the input stage. Diodes  $D_1$  and  $D_2$  are added to speed up the voltage swing at the emitters of  $Q_1$  and  $Q_2$  for large input voltage swings.

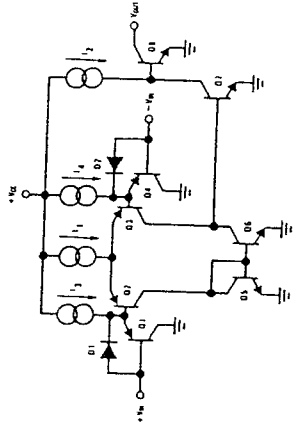


FIGURE 3. Complete LM139 Comparator Circuit

Biasing for current sources  $I_1$  through  $I_4$  is shown in Figure 4. When power is first applied to the circuit, current flows through the JFET  $T_{13}$  to bias up diode  $D_5$ . This biases transistor  $Q_{12}$  which turns ON transistors  $Q_9$  and  $Q_{10}$  by allowing a path to ground for their base and collector currents.

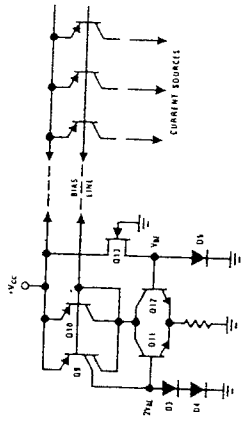


FIGURE 4. Current Source Biasing Circuit

Current from the left hand collector of  $Q_9$  flows through diodes  $D_3$  and  $D_4$  bringing up the base of  $Q_{11}$  to 2  $V_{BE}$  above ground and the emitters of  $Q_{11}$  and  $Q_{12}$  to one  $V_{BE}$ .  $Q_{12}$  will then turn OFF because its base emitter voltage goes to zero. This is the desired action because  $Q_9$  and  $Q_{10}$  are biased ON through  $Q_{11}$ ,  $D_3$  and  $D_4$  so  $Q_{12}$  is no longer needed. The "bias line" is now sitting at a  $V_{BE}$  below  $+V_{CC}$  which is the voltage needed to bias the remaining current sources in the LM139 which will have a constant bias regardless of  $+V_{CC}$  fluctuations. The upper input common mode voltage is  $V_{CC}$  minus the saturation voltage of the current sources (approximately 100 mV) minus the 2  $V_{BE}$  of the input devices  $Q_1$  and  $Q_2$  (or  $Q_3$  and  $Q_4$ ).

Figure 5 shows a basic comparator circuit for converting low level analog signals to a high level digital output. The output pull-up resistor should be chosen high enough so as to avoid excessive power dissipation yet low enough to supply enough drive to switch whatever load circuitry is used on the comparator output. Resistors  $R_1$  and  $R_2$  are used to set the input threshold trip voltage ( $V_{REF}$ ) at any value desired within the input common mode range of the comparator.

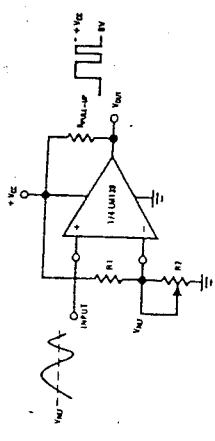


FIGURE 5. Basic Comparator Circuit

COMPARATORS WITH HYSTERESIS  
The circuit shown in Figure 5 suffers from one basic drawback in that if the input signal is a slowly varying low level signal, the comparator may be forced to stay within its linear region between the output high and low states for an undesirable length of time. If this happens, it runs the risk of oscillating since it is basically an uncompensated, high gain op amp. To prevent this, a small amount of positive feedback or hysteresis is added around the comparator. Figure 6

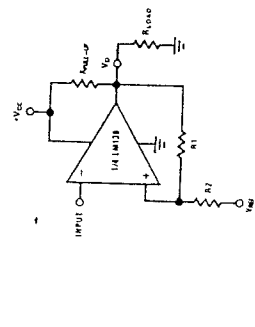


FIGURE 6. Comparator with Positive Feedback to Improve Switching Time

shows a comparator with a small amount of positive feedback. In order to insure proper comparator action, the components should be chosen as follows:

$$R_{PULL-UP} < R_{LOAD} \text{ and} \\ R_1 > R_{PULL-UP}$$

This will insure that the comparator will always switch fully up to  $+V_{CC}$  and not be pulled down by the load or feedback. The amount of feedback is chosen arbitrarily to insure proper switching with the particular type of input signal

TU/H/7385-6

TU/H/7385-5

TU/H/7385-3

TU/H/7385-4

TU/H/7385-2

TU/H/7385-1

reference input ( $V_{IN} > V_{REF}$ ). This will drive the output towards ground which in turn pulls  $V_{REF}$  down. Since  $V_{REF}$  is actually the noninverting input to the comparator, it too will drive the output towards ground as fast as possible switching time regardless of the input moves. If the input then travels down to the same procedure will occur only in the opposite direction ensuring that the output will be driven hard towards

hysteresis in the feedback loop of the comparator we use, however, than simply as an oscillation. It can be made to function as a Schmitt trigger with variable trigger points. A typical circuit is shown in Figure 7. Again, the hysteresis is achieved by shifting the voltage at the positive input when the output voltage changes state. This network requires only three resistors referenced to the positive supply  $+V_{CC}$  of the comparator. This can be modeled as a resistive divider,  $R_1$  between  $+V_{CC}$  and ground with the third resistor,  $R_2$ , fully connected to  $+V_{CC}$  or ground, paralleling it to  $R_1$ . To analyze this circuit, assume that the voltage,  $V_{IN}$ , at the inverting input is less than  $V_A$ . With the output will be high ( $V_O = +V_{CC}$ ). The upper voltage,  $V_{A1}$ , is defined by:

$$V_{A1} = \frac{+V_{CC} R_2}{(R_1 \parallel R_3) + R_2} \quad (1)$$

$$V_{A1} = \frac{+V_{CC} R_2 (R_1 + R_3)}{R_1 R_2 + R_1 R_3 + R_2 R_3} \quad (1)$$

$$V_{A2} = \frac{+V_{CC} R_2 \parallel R_3}{R_1 + R_2 \parallel R_3} \quad (2)$$

$$V_{A2} = \frac{+V_{CC} R_2 \parallel R_3}{R_1 + \frac{R_2 R_3}{R_2 + R_3}} \quad (2)$$

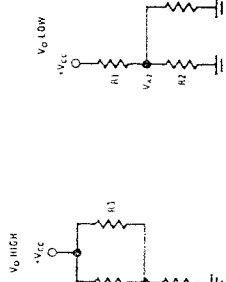
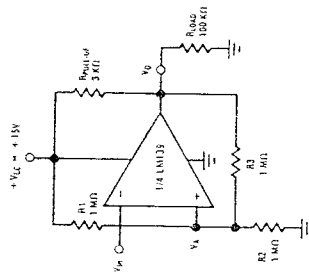


FIGURE 7. Inverting Comparator with Hysteresis

lower input trip voltage,  $V_{A2}$ , is now defined by:

$$V_{A2} = \frac{+V_{CC} R_2 \parallel R_3}{R_1 + R_2 \parallel R_3} \quad (3)$$

or

$$V_{A2} = \frac{+V_{CC} R_2 R_3}{R_1 R_2 + R_1 R_3 + R_2 R_3} \quad (3)$$

When the input voltage,  $V_{IN}$ , decreases to  $V_{A2}$  or lower, the output will again switch high. The total hysteresis,  $\Delta V_A$ , provided by this network is defined by:

$$\Delta V_A = V_{A1} - V_{A2} \quad (4)$$

or, subtracting equation 2 from equation 1

$$\Delta V_A = \frac{+V_{CC} R_1 R_2}{R_1 R_2 + R_1 R_3 + R_2 R_3} \quad (4)$$

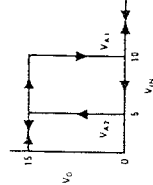
To insure that  $V_O$  will swing between  $+V_{CC}$  and ground, choose:

$$R_{PULL-UP} < R_{LOAD} \text{ and } R_3 > R_{PULL-UP} \quad (5)$$

Heavier loading on  $R_{PULL-UP}$  (i.e. smaller values of  $R_3$  or  $R_{LOAD}$ ) simply reduces the value of the maximum output voltage thereby reducing the amount of hysteresis by lowering the value of  $V_{A1}$ . For simplicity, we have assumed in the above equations that  $V_O$  high switches all the way up to  $+V_{CC}$ .

To find the resistor values needed for a given set of trip points, we first divide equation (3) by equation (2). This gives us the ratio:

$$\frac{\Delta V_A}{V_{A2}} = \frac{1 + \frac{R_1}{R_3} + \frac{R_1}{R_2}}{1 + \frac{R_2}{R_3} + \frac{R_2}{R_1}} \quad (6)$$



$$\frac{\Delta V_A}{V_{A2}} = n \quad (7)$$

We can then obtain an expression for  $R_2$  from equation (1) which gives

$$R_2 = \frac{R_1 \parallel R_3}{+V_{CC} - n V_{A1}} \quad (8)$$

The following design example is offered:

Given:  $V^+ = +15V$

$$R_{LOAD} = 100 \text{ k}\Omega$$

$$V_{A1} = +10V$$

$$V_{A2} = +5V$$

To find:  $R_1, R_2, R_3, R_{PULL-UP}$

Solution:

From equation (4)  $R_{PULL-UP} < R_{LOAD}$

$$R_{PULL-UP} < 100 \text{ k}\Omega$$

$$R_{PULL-UP} = 3 \text{ k}\Omega$$

From equation (5)  $R_3 > R_{LOAD}$

$$R_3 > 100 \text{ k}\Omega$$

$$R_3 = 1 \text{ M}\Omega$$

$$R_3 = 1 \text{ M}\Omega$$

$$\text{so let } R_3 = 1 \text{ M}\Omega$$

$$\text{From equation (7) } n = \frac{\Delta V_A}{V_{A2}} = \frac{10 - 5}{5} = 1$$

$$\text{and since } R_1 = n R_3$$

$$R_1 = 1 R_3 = 1 \text{ M}\Omega$$

$$\text{this gives } R_1 = 1 \text{ M}\Omega$$

$$\text{From equation (8) } R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

$$R_2 = \frac{500 \text{ k}\Omega}{15 - 1} = 1 \text{ M}\Omega$$

The trip voltage,  $V_A$ , at the positive input is shifted about  $V_{REF}$  as  $V_O$  changes between  $+V_{CC}$  and ground. Again for analysis, assume that the input voltage,  $V_{IN}$ , is low so that the output,  $V_O$ , is also low ( $V_O = \text{GND}$ ). For the output to switch,  $V_{IN}$  must rise up to  $V_{IN1}$  where  $V_{IN1}$  is given by:

$$V_{IN1} = \frac{V_{REF} (R_1 + R_2)}{R_2} \quad (9)$$

As soon as  $V_O$  switches to  $+V_{CC}$ ,  $V_A$  will step to a value greater than  $V_{REF}$  which is given by:

$$V_A = V_{IN} + \frac{[V_{CC} - V_{IN1}] R_1}{R_1 + R_2} \quad (10)$$

To make the comparator switch back to its low state ( $V_O = \text{GND}$ )  $V_{IN}$  must go below  $V_{REF}$  before  $V_A$  will again equal  $V_{REF}$ . This lower trip point is now given by:

$$V_{IN2} = \frac{V_{REF} (R_1 + R_2) - V_{CC} R_1}{R_2} \quad (11)$$

The hysteresis for this circuit,  $\Delta V_{IN}$ , is the difference between  $V_{IN1}$  and  $V_{IN2}$  and is given by:

$$\Delta V_{IN} = V_{IN1} - V_{IN2} = \frac{V_{REF} (R_1 + R_2) - V_{CC} R_1}{R_2} \quad (12)$$

As a design example consider the following:

Given:  $R_{LOAD} = 100 \text{ k}\Omega$

$$V_{IN1} = 10V$$

$$V_{IN2} = 5V$$

$$+V_{CC} = 15V$$

To find:  $V_{REF}, R_1, R_2$  and  $R_3$

Solution:

Again choose  $R_{PULL-UP} < R_{LOAD}$  to minimize loading, so let

$$R_{PULL-UP} = 3 \text{ k}\Omega$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_1 = \frac{\Delta V_{IN}}{V_{REF}}$$

$$R_2 = \frac{V_{CC}}{V_{REF}}$$

$$R_3 = \frac{\Delta V_{IN}}{V_{REF}}$$

FIGURE 8. Non-Inverting Comparator with Hysteresis