AUTOMATIC RAILWAY SIGNALING USING MICRO CONTROLLER

Project Report

Submitted by

A. Ganesh Kumar
D. Kannan
D. S. Nanda Kumar
S. Yogaprakash

P-705

Guided by

Prof. K. Rajan M.E.,
Assistant Professor,
Department of EEE.

2001 - 2002

In Partial fulfillment of the requirements
for the award of Degree of
BACHELOR OF ENGINEERING in
ELECTRICAL AND ELECTRONICS ENGINEERING
Of Bharathiar University.

P.705

Department of Electrical and Electronics Engineering

Kumaraguru College of Technology

Coimbatore – 641 006.

Department of Electrical and Electronics Engineering

Kumaraguru College of Technology

Coimbatore – 641 006.

CERTIFICATE

This is to certify that the report entitled
"AUTOMATIC RAILWAY SIGNALING USING MICRO CONTROLLER"
has been submitted by

A. Ganesh kumar
D. Kannan
D. S. Nanda Kumar
S. Yokaprakash

In partial fulfillment of the requirements for the award of Bachelor of Engineering in Electrical and Electronics Engineering Branch of Bharathiar University, Coimbatore – 641 046 during the academic year 2001 – 2002.

..... Guide

..... Head of the Department

V. DURAISAMY, M.E., M.I.S.T.E.,
Assistant Professor of Elec. Engg.
Kumaraguru College of Tech.
COIMBATORE - 641 006.

Certified that the candidate with University Registration Number .....

Was examined in the project work viva – voice Examination held on.....

....

Internal Examiner External Examiner

ACKNOWLEDGEMENT

With deep sense of gratitude we express our heartful thanks to our guide **Prof. K. Rajan,** M.E., Assistant Professor in EEE for his guidance valuable suggestions and constant interest evinced by him throughout the course of the project work.

We are highly grateful to our beloved professor and Head of the Department **Prof. V. Duraiswamy,** M.E., for his remarkable support and encouragement.

Our sincere thanks are due to our principal **Dr. K. K. Padmanabhan,** B.Sc(Engg.), M. Tech., Ph.D., M.I.S.T.E., F.I.E., for having made available all the facilities to do this project.

We have no words to express our profound gratitude to our class advisor **Ms. Rani Thottungal,** M. E., for her valuable help for the project.

We are indebted to the support, encouragement and help rendered by all the faculty members and non – teaching staff of EEE department.

Last but not the least we thank our dear friends for helping us a lot with their innovative ideas.

Synopsis

This project is basically a traffic management system that can be used for railways. This is intended towards proper automatic signaling by sensing the train and the number of wheels in the train. The principle of electromagnetic flux is being used. As the sensing is done using the ferrite block transmitter and receiver, the control of the system is done by the combination of hardware and software logics.

The simplest and trouble free circuits are being designed for the conversion of the signal to sensible pulses. The brain of the system is the central MICRO CONTROLLER, which governs the entire system and activates the signal accordingly. The ATMEL 89C51 Micro Controller is being used for mastering the system.

The sole system is self capable for effective signaling for performing the below mentioned tasks

- ✓ Time and gap maintenance between trains
- ✓ Check for full train passage without leave over
- ✓ Accident prevention by opposing trains
- ✓ Circuit clearance during shunting

Entire software logics are used for wheel count comparison and activation of appropriate signals for which the troubleshooting is easier. The system is economical and accurate in operation.

Contents

CHAPTER 1

Introduction

Indian railways is the largest rail network in Asia and world's second largest under one management.. The economy of India depends upon various industries. The Indian Railways is one such industry, which is wide spread throughout the country. It is one of the largest industries that has huge capital. The whole railway system of India is run by the Central Government of India for the welfare of people.

The railways that criss – cross the country's vast geographical spread for a stretch more than 1lakh route Kilometers. Carrying in excess of 11 Million passengers and hauling a million of tones of freight each day

Safety is one of the important aspects in railways. When accidents do happen they can be catastrophic in their toll of human life and property. This has provoked public concern over safety. Though the railways department of the central government takes lots of steps towards the safety aspects and thereby tries to prevent accidents, we can see accidents happening around us frequently. The problem is still lying because of the improper human actions. Though the system is strong, unless the operator is cautious and sends correct information to the gang man and the gang man works appropriately, the prevention of accidents are very difficult.

One way to solve the problem is having separate tracks for the on going and down coming trains all the way. This is practically impossible because of the other constrains like space, investment, maintenance costs etc.

The challenge face by signaling is to safely transport more and more transport and freight, on the same tracks will decrease operation and maintenance cost.

Accidents on the network are frequent exceeding 300 per year. However this can be seen in the context of the massive scale of the operation.

Indian Railway Accident Rate

Year	Collisions	Derailments	Level Crossing accidents	Fire in trains	Total	Train accidents per million train Km.
1995-96	29	296	68	5	398+	0.61
1996-97	**26**	**286**	**65**	**4**	**381***	**0.57**

+ Includes 1 accident on Metro railway.
* Includes 1 accident on Metro Railway and 3 on Konkan Railway.

Source: Indian Railways

The size and complexity of their operations, growing traffic and changing technologies, placed inevitably a heavy burden on this

manual information system. Need for its modernization was therefore felt for sometime.

The aim of our project is to make an automatic system for the effective signaling and track train maintenance. The automation is tried out with the usage of a MICRO CONTROLLER thereby the size and ease of control and signaling is made simple and economic.

Our system is one, which will sense the train's wheel one by one at 2 different destinations separated by a distance (say 10 KM). Compares the signal counts and provides the signals appropriately.

1.1 INTRODUCTION TO MICRO CONTROLLERS:

A Micro Controller is nothing but a true computer on a single chip. The design incorporates all the feature found in a microprocessor like CPU, ALU, PC, SP and registers. It also has added other features needed to make a complete computer like inbuilt Timer, Interrupt controller, interfacings, ROM, RAM, Parallel I/O, serial I/O, counters, clock circuit etc.

Like a micro processor, a micro controller is a general purpose device, that is mean to read data, perform limited calculations on that data and control its environment based on those calculations. The primary use of a micro controller is to control the operations of the

machine using a fixed programme that is stored in ROM and that does not change over the life of a system.

The micro controller design uses a much more limited set of single and double byte instructions that are used to move code and data from the memory to ALU. Many instructions are coupled with pins on the integrated circuit package. The pins are programmable, which means several different functions depending on the wishes of the programmer can be made.

89C51 Micro Controller :

The ATMEL 89C51 family is a highly versatile general-purpose 8-bit system with 4K bytes flash. Its enhanced architecture offers applications requiring a high degree of on chip functionality. It is most suited for control-oriented applications.

Features :

- compatible with MCS-51 Products
- 4K Bytes of in – system programmable Flash Memory (1000 write/erase cycles)
- Fully static operation : 0 Hz to 24 MHz
- Three level programme memory lock
- 128x8 bit internal RAM
- 32 Programmable I/O lines
- Two 16 bit timer/counter
- 6 interrupt sources

- Programmable serial channel
- Low-power idle and power-down modes

CHAPTER 2

HARDWARE DESCRIPTIONS

CHAPTER 2

Hardware Description

2.1 MICROCONTROLLER :

Ports structure and operation

The 89C51 consists of 4 ports P0, P1, P2, and P3.

Port 0:

Port 0 is an 8 bit open drain bi – directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pin, the pin can be used as high impedance inputs. Port 0 may also be configured to be multiplexed low order address/data bus during access to external program and data memory. In this mode P0 has internal pull-ups. Port 0 also receives the code bytes during programming and outputs the code bytes during verification. External pull-ups are required during the program verification.

Port 1 :

Port 1 is an 8 bit bi – directional I/O port with internal pull-ups. The port 1 output buffer can sink/source four TTL inputs. When 1s are written to port 1 pins they are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally being pulled low will source current(I_{il}) because of the internal pull-ups. Port 1 also receives the low order address bytes during Flash programming and verification.

Port 2 :

Port 2 is an 8 bit bi – directional I/O port with internal pull-ups. The port 2 output buffers can sink/source four TTL inputs. When 1s are written to port 2 pin they are pulled high buy the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current(I_{IL}) because of the internal pull-ups. Port 2 emits the high – order address byte during fetches from external program memory and during accesses to external data memory that use 16 bit addresses. Port 2 also receives the high order address bits and some control signals during flash programming and verification.

Port 3 :

Port 3 is an 8 bit bi – directional I/O port with internal pull-ups. The port 3 output buffer can sink/source four TTL inputs. When 1s are written to port3 pins they are pulled high by the internal pull-ups and cab be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups.

Port 3 also serves the function of various special features of the AT89C51 as listed below:

Port Pun	Alternate Functions
P3.0	RXD(Serial input port)
P3.1	TXD(Serial Output Port)
P3.2	$\overline{\text{Int0}}$ (External Interrupt 0)
P3.3	$\overline{\text{Int1}}$ (External Interrupt 1)
P3.4	T0(timer 0 external input)
P3.5	T1(timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$(Externla data memory read Strobe)

RST:

Reset input. A high on this pin for two machine cycles while the oscillator is funning reset the device.

$\overline{\text{ALE}}$/PROG:

Address latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input during flash programming.

In normal operation ALE is emitted at a constant rate of 1/6the oscillator frequency, and may be used for external timing or clocking purpose. If desired, ALE operations can be disabled by setting bit 0 of SFR location 8Eh. With the bit set, ALE is active only during a mov x instruction.

$\overline{\text{PSEN}}$:

Program store enable is the read strobe to external program memory.

While AT89C51 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

EA/VPP:

External Access Enable. It must be strapped to GND in order to enable the device to fetch code from external program memory locations starting from 0000H up to FFFFH.

EA should be strapped to V_{cc} for internal program executions.

XTAL1 :

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL 2:

Output from the inverting oscillator amplifier.

MICROCOMPUTER HARDWARE LAYOUT

2.2 Programmable Peripheral Interface : Intel 8255

The 8255 is a widely used programmable parallel I/O device. It can be programmed to transfer data under various conditions from simple I/O to interrupt I/O. The 8255 has 24 I/O pins that can be grouped primarily in two 8 bit parallel ports : A & B with remaining 8 bits as port c. The 8 bits of port C can be used as individual bits or be grouped into two 4bit ports: C upper and C lower. The functions of the 8255 can be classified into two modes:

Bit Set / Reset (BSR)

I / O Mode

The BSR mode is used to set or reset the bits in port C. The I/O mode is further divided into 3 Modes : Mode 0,Mode 1,Mode2.

In mode 0,all ports function as simple I/O ports.

Mode 1 s a handshake mode, whereby ports A and /or B use bits from Port C as handshake signals.

In mode 2, port A can be setup for Bi directional data transfer using handshake signals from Port C and Port B can be set up either in mode 0 or mode 1.

2.3 DECODER LS 74139 :

It is a 2x4 decoder with 2 binary weighted inputs (B, A or A1, A0) and one active low enable input. When it is enable, one of the 4 output signals corresponding to the decimal equivalent of the input goes active low. When it is not enabled all output signals remain high.

2.4 LATCH LS 74373 :

It is a 4 bit latch, each pair of bits is controlled by high enable input E. It also has complementary outputs ($\overline{Q}$ and Q). The information at D input is transferred to Q output when E is high and Q follows D input as long as E is high. When E goes low D input is latched at the output and remains latched until E goes high again.

CHAPTER 3

Project Description

3.1 Components:

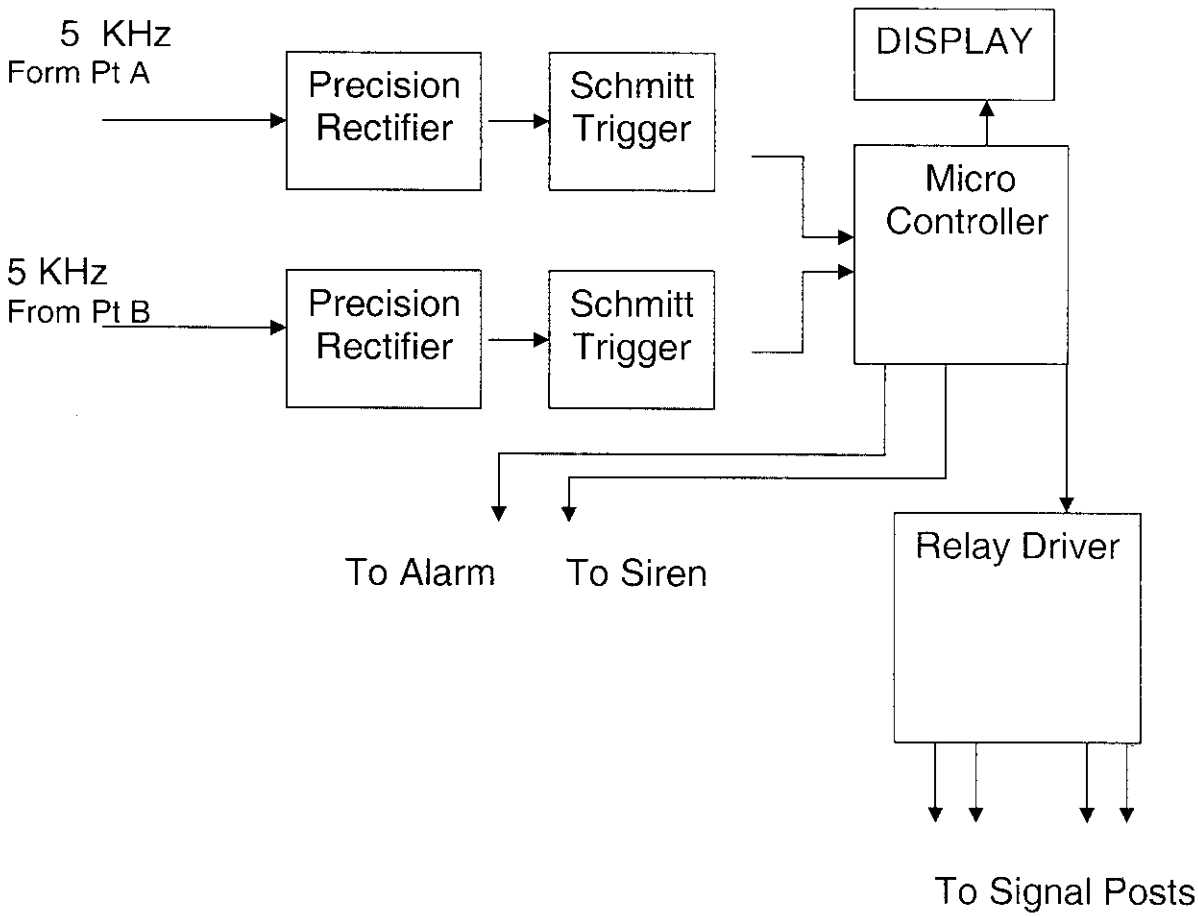
The generalized block diagram is shown in the figure 3.1. The arrangement

SIGNAL GENERATOR :

It is a frequency generator. 5 KHz sustained sine wave oscillation is being produced in the oscillator circuit. The oscillator to be used may be of crystal oscillator. The output waveform may be 5 KHz $5V_{pp}$. The generated signal is electric signal which can be transferred to the transmitter by a shielded telephone cable.

TRANSMISSION MEDIUM :

Conventionally used telephone cables are noise prone though they are shielded. They also have a possibility of radio interference and produce pronounceable attenuation. So, fibre optic cables of required specifications are to be used. Proper interfacing circuits for encoding the electrical signals to optical and optical to electrical decoding are to be done.


5 KHz
Form Pt A
Precision Rectifier
Schmitt Trigger
DISPLAY
Micro Controller
5 KHz
From Pt B
Precision Rectifier
Schmitt Trigger
To Alarm
To Siren
Relay Driver
To Signal Posts

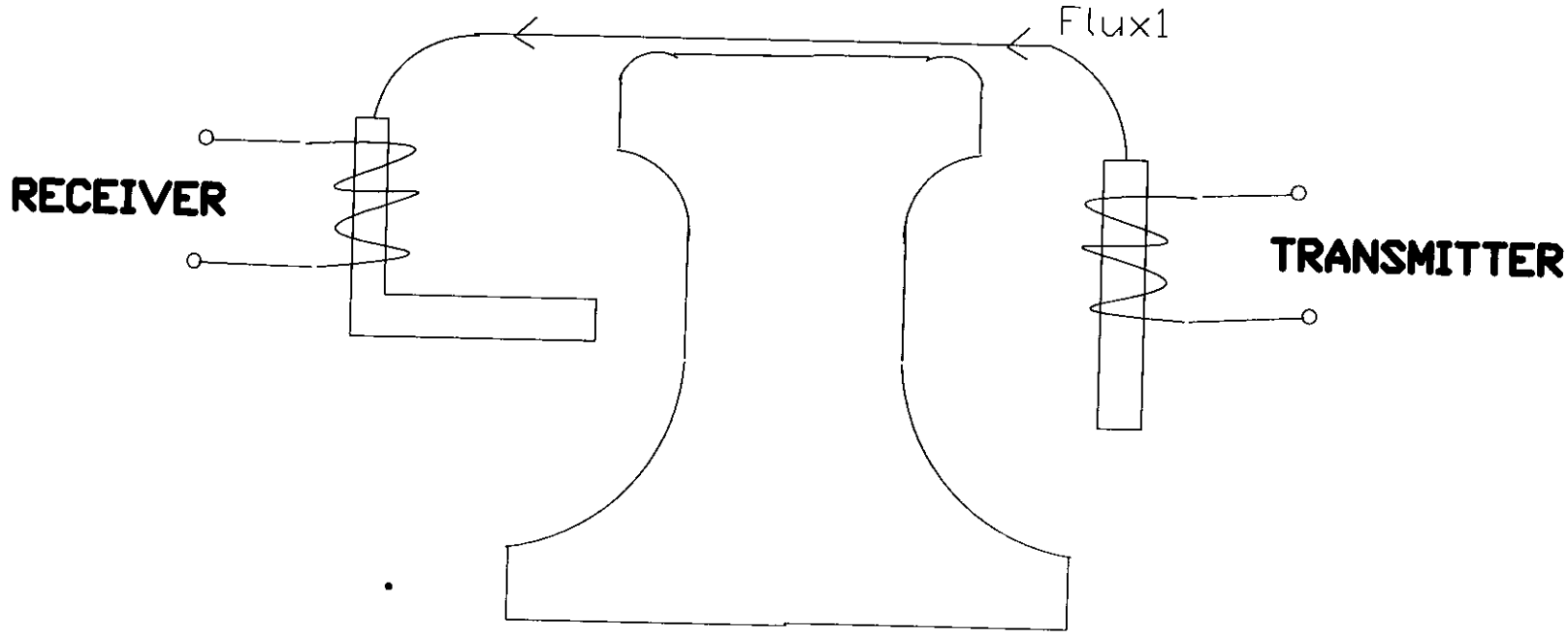
TRANSMITTER :

An optical to electrical decoder is to be used to convert the signal and fed to the transmitter ferrite block, which is totally rectangular in cross section. 5KHz signal is supplied to the transmitter and it transmits the flux lines through the medium of air and iron rail.

FLUX PATH :

The rail, which is made of soft iron, will serve the path for flux. As long as the transmitter has the electrical signal the flux lines will be passing. Whenever there is any disturbance due to the passage of a train wheel then flux path gets altered and it flows through the iron wheel also and so the flux path will be deviated.

MAGNETIC FLUX PATH IN RAIL



RECEIVER :

Ferrite block of rectangular cross section is used as a receiver. As long as the rail passes the flux lines, the receiver gets the signal and sends out continuous signal output. Whenever the wheel passes, the path gets deviated and no signal is received. These signals are encoded to optical signals and then transported.

3.2 DIP :

A Dip is the sensing signal that is caused due to the distortion in the flux path which causes a no signal in the receiver. When the train's metallic wheel passes, the flux lines will pass through the wheel also and the signal at the receiver is not felt.

THE DIP IS OF SEVERAL TYPES :

Depending on the sensing signal and the speed of sensing, the dip is classified to several types :

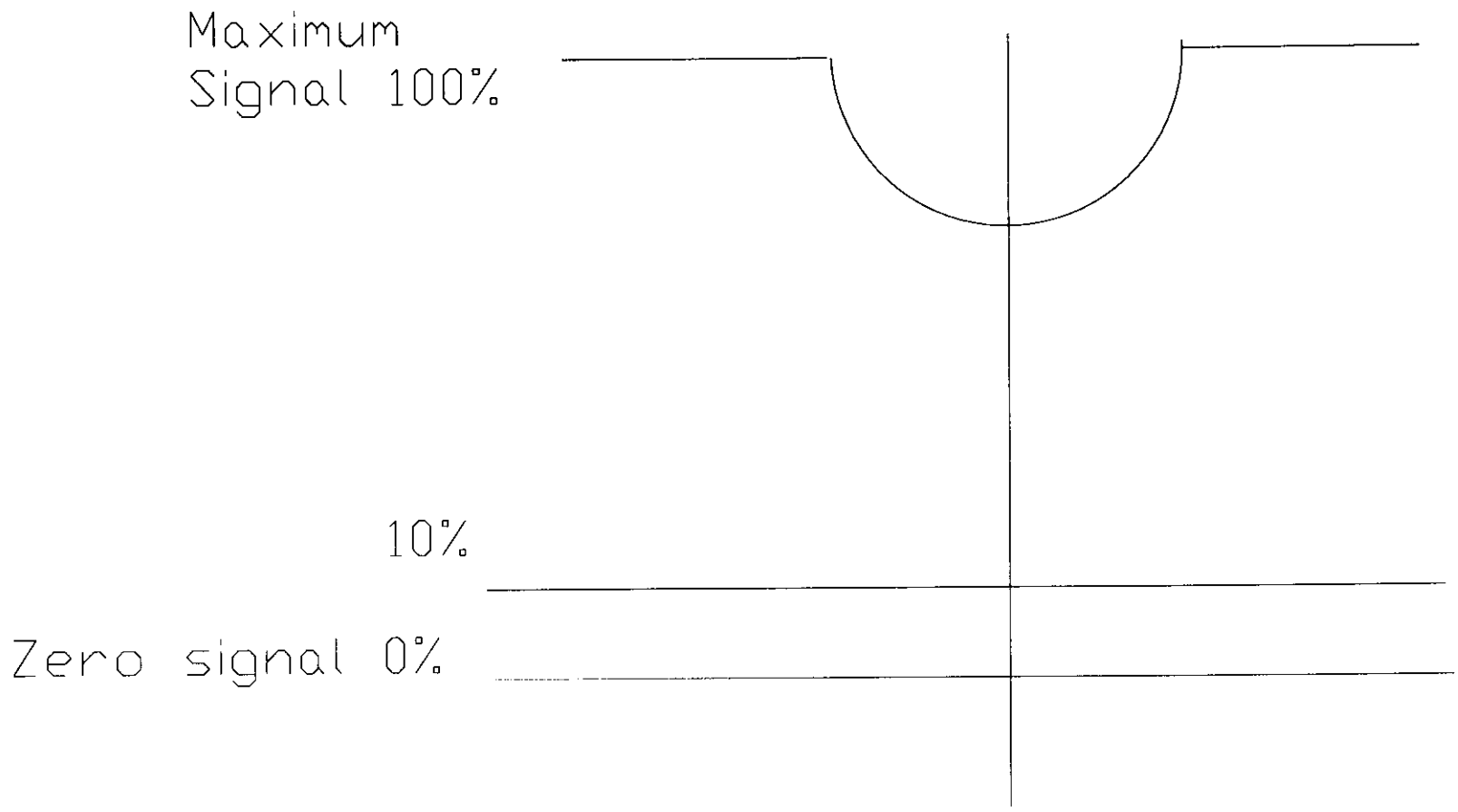
1. Inadequate Dip
2. Predominant double dip
3. Full dip
4. Wide and full dip

When the wheel passes between the TX and RX coils, the magnetic flux path gets disturbed and induce voltage in the RX coil reduce substantially. The drop in the RX coil output under the influence of wheel is called "Wheel Dip".

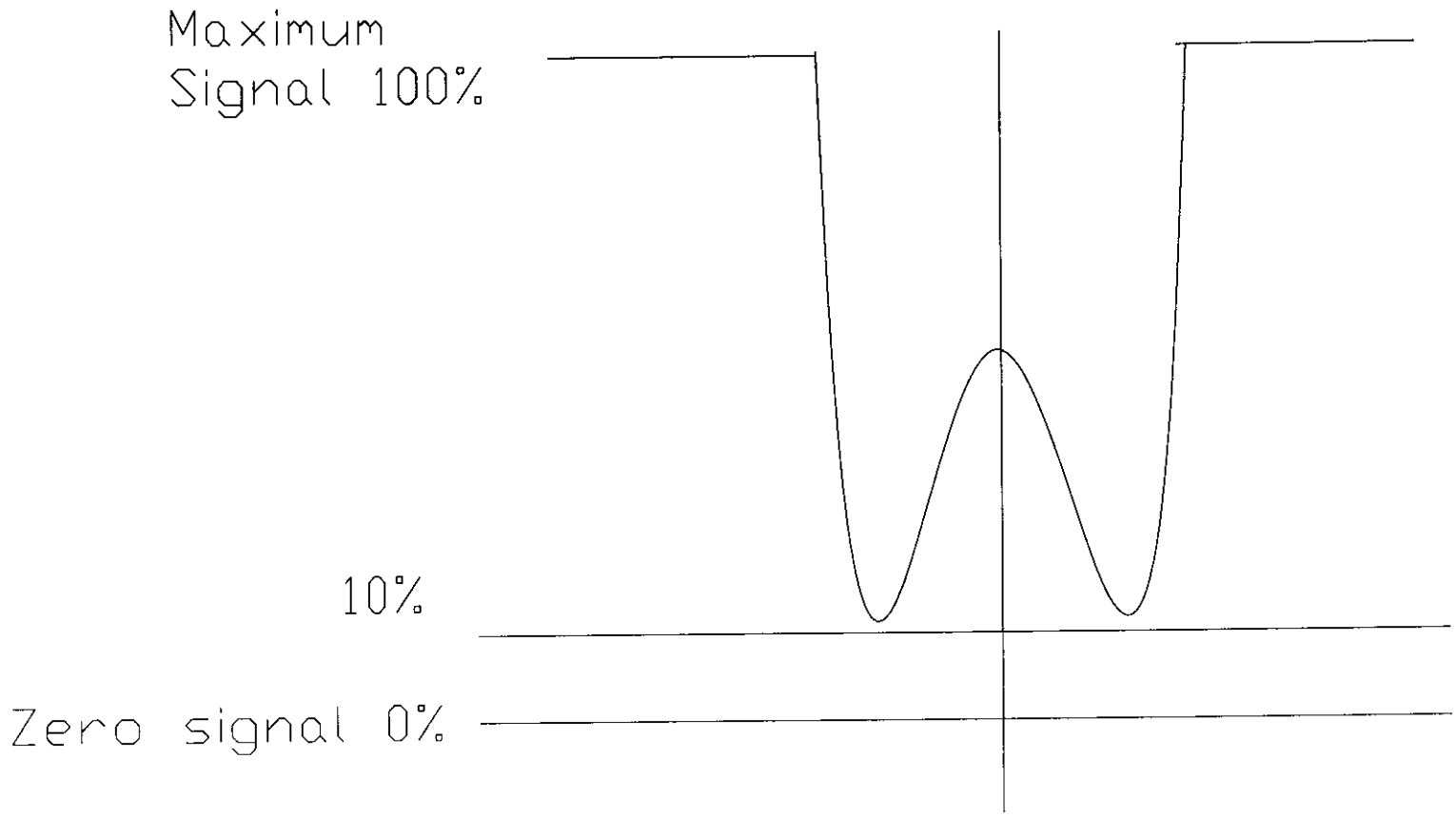
Wheel dip depends on several factors such as type of wheel, rail profile, type of sleepers and orientation of the wheel in relation to transmitter and receiver coils.

1. ' INADEQUATE WHEEL DIP ' - causes where signal does not fall fully. In this position the system may miss some counts. This is undesirable.
2. 'PREDOMINANT DOUBLE DIP' – causes where the signal level fall to minimum but as the wheel move further towards the center line of the track devices, the signal level rises again, falls to minimum second time and then rises as the wheel moves further away from the track device. This type of dip may cause extra counts. This is undesirable.
3. 'CORRECT DIP' – produced as a sharp single dip and the signal level falls to minimum only when the wheel is over the center line of the track device. In this position the system will count correctly.
4. 'DOUBLE DIP' causes where the dip is slightly broadened and rise in the signal at the center line of the track device is less that 5%. This is considered the most correct adjustment of the wheel dip.

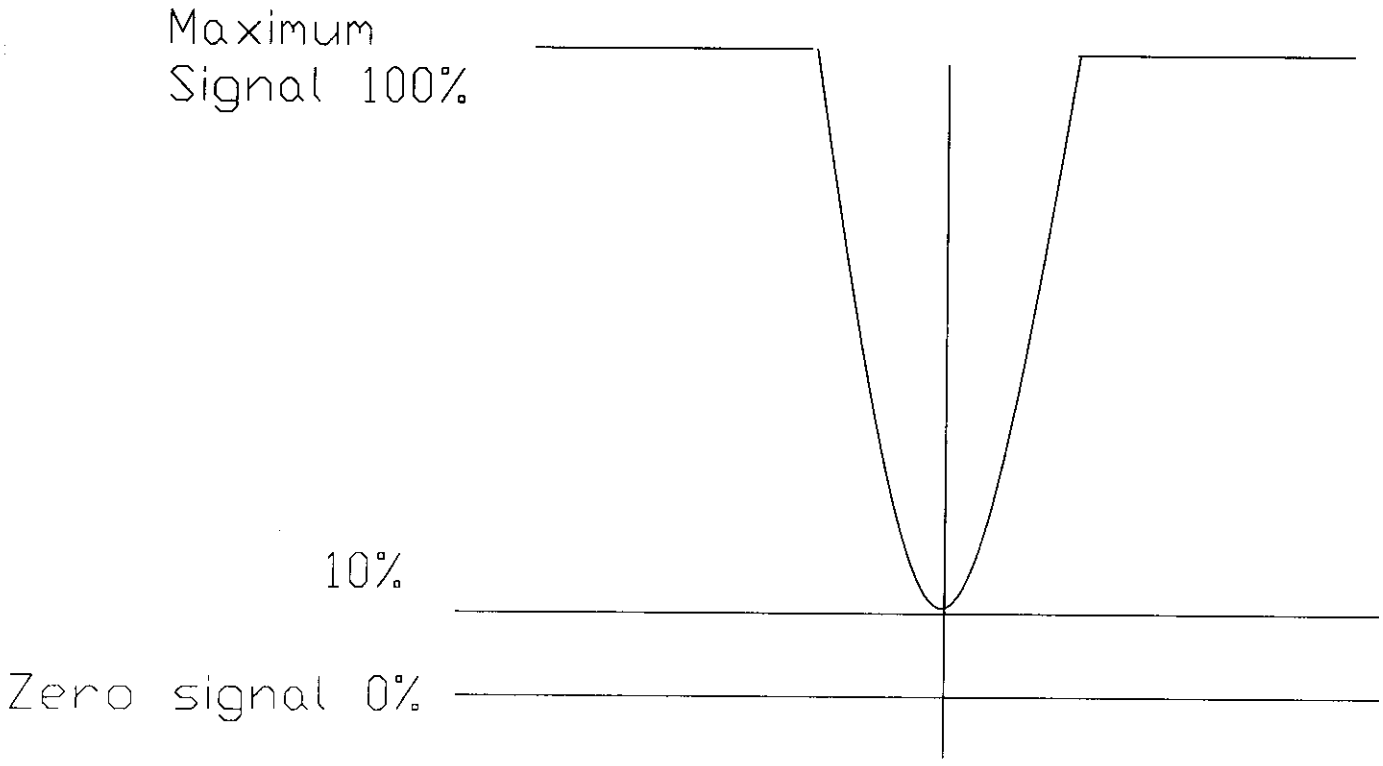
INADEQUATE DIP



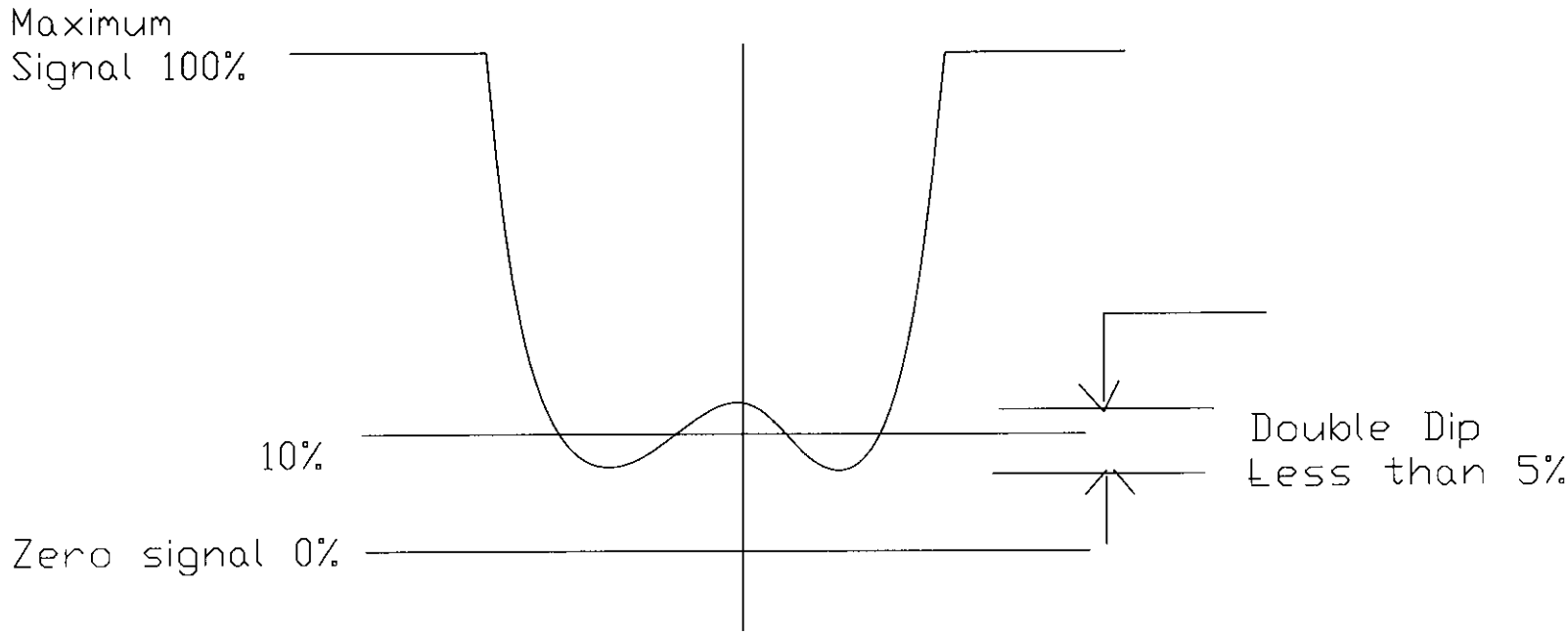
PREDOMINATE DOUBLE DIP



FULL DIP (Correct Adjustment)



WIDE AND FULL DIP (OPTIMALLY CORRECT)



3.3 PRESICION RECTIFIER:

The optical signal from the receiver is decoded to 5V signal of 5 KHz frequency. It is fed to the precision rectifier. The output of the precision rectifier is a full-rectified DC waveform. The diagram shows a precision rectifier.

OPERATION :

A full wave rectifier or absolute value circuit is shown in the figure. For positive input i.e . $v_i>0$, diode D_1 in *on* and D_2 is *off*. Both the op – amps A_1 and A_2 act as inverter as shown in the equivalent circuit in the figure. It can be seen that $V_0 = V_1$.

For negative input i.e. $V_1<0$, diode D_1 is *off* and D_2 is *on*. The equivalent circuit is shown in the figure. Let the output voltage of the op – amp A_1 be V. Since the differential input to A_2 is zero, the inverting input terminal is also at the voltage V.

KCL at node A fives

$$V_1 / R + V / 2R + V / R = 0$$

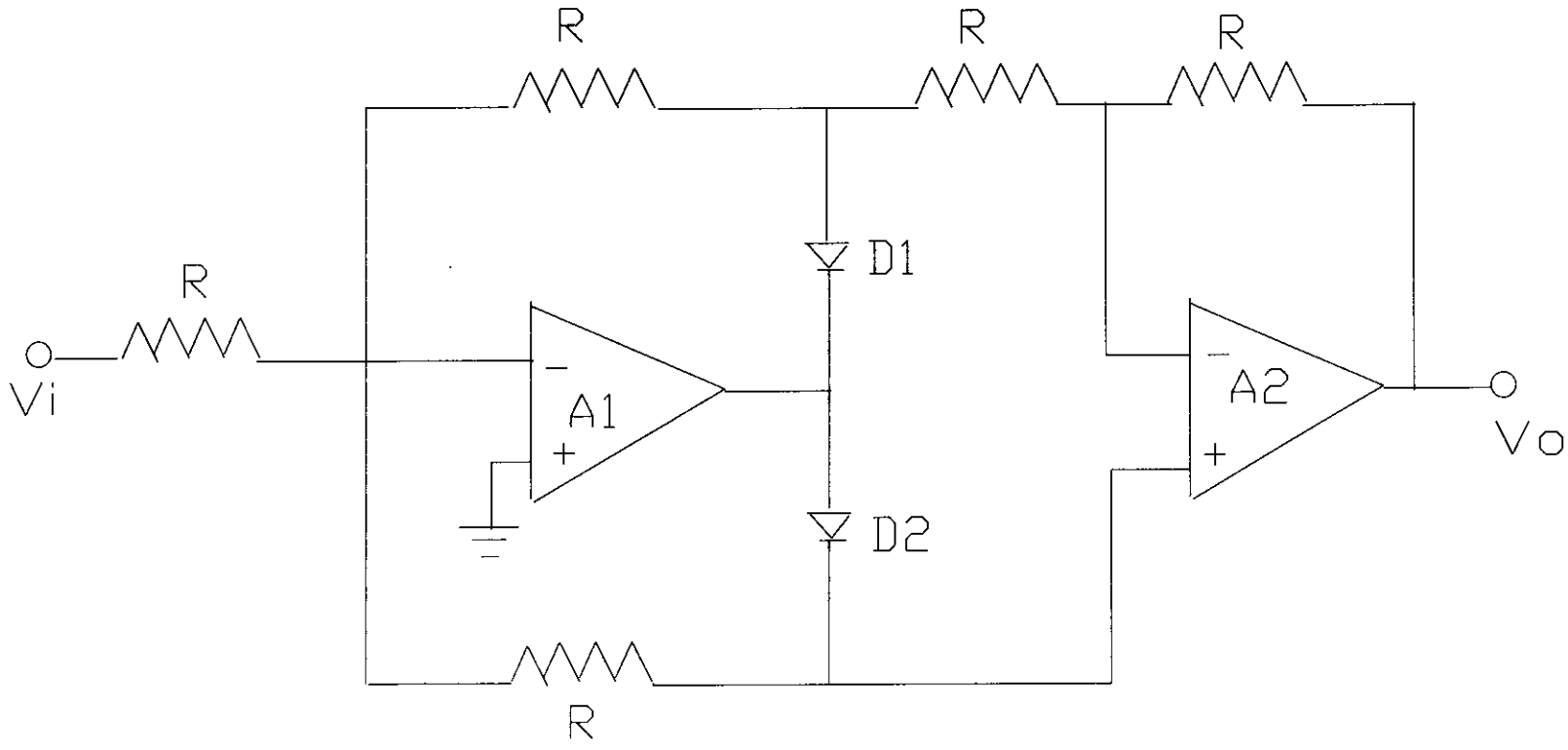
or $$V = -2/3V_1$$

The equivalent circuit of the figure is a non – inverting amplifier as shown in the next figure. The outputV_0 is

$$V_0 = (1 + R/2R)(-2/3V1) = V_2$$

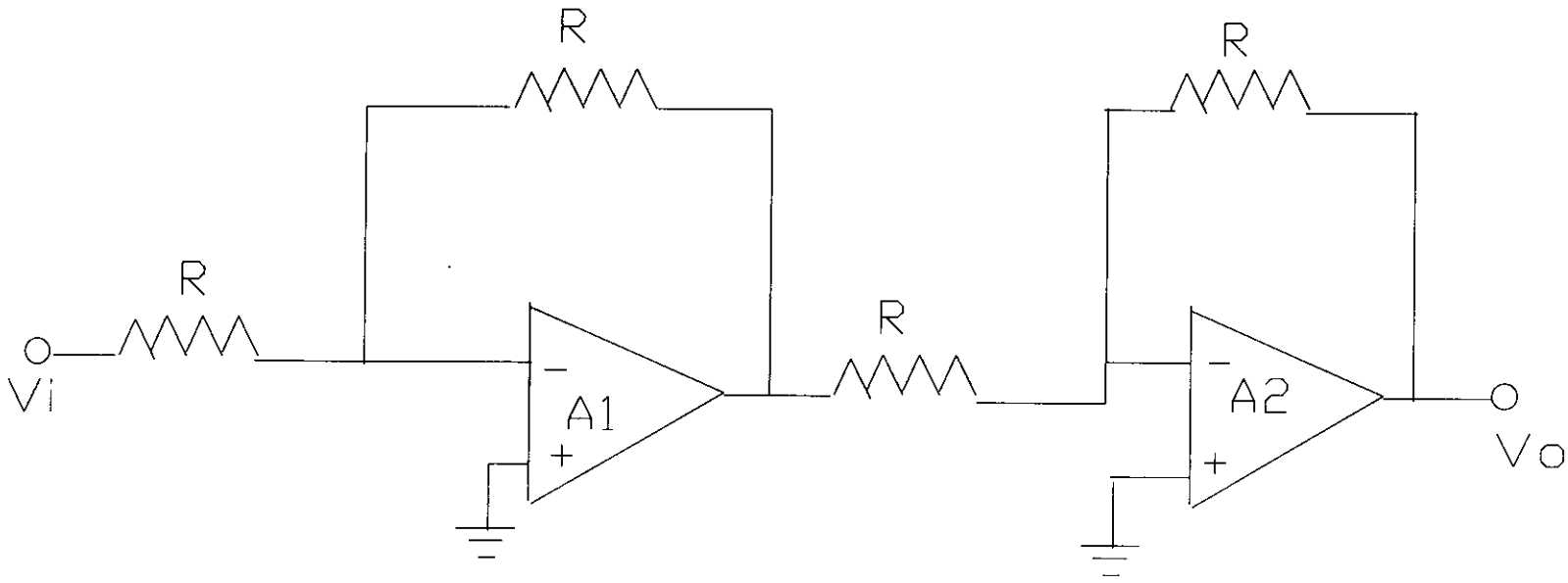
Hence for $V_1<0$, the output is positive. The input and output waveform are shown.

FULL WAVE PRECISION RECTIFIER



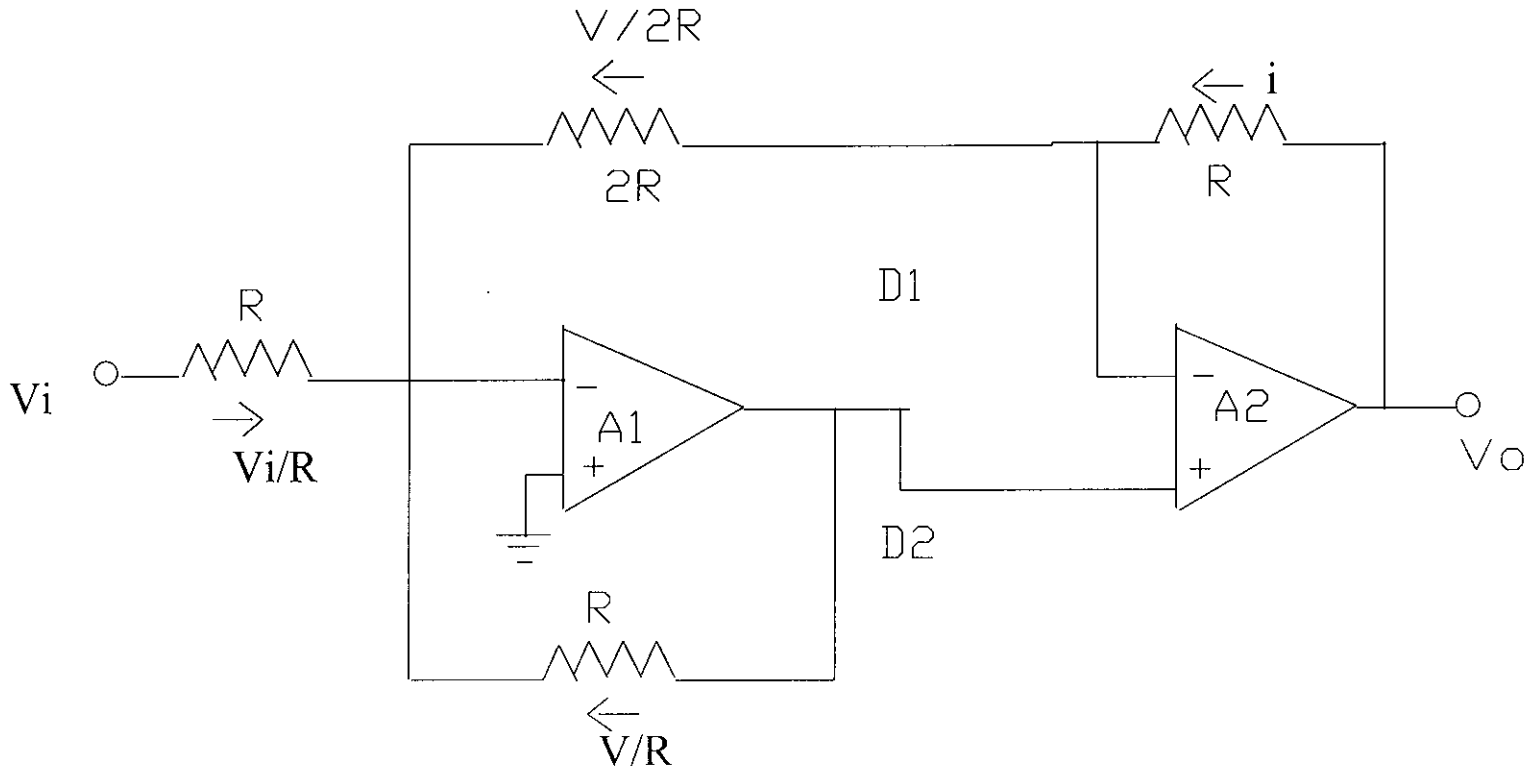
Precision Rectifier

Equvalent Circuit for Vi > 0



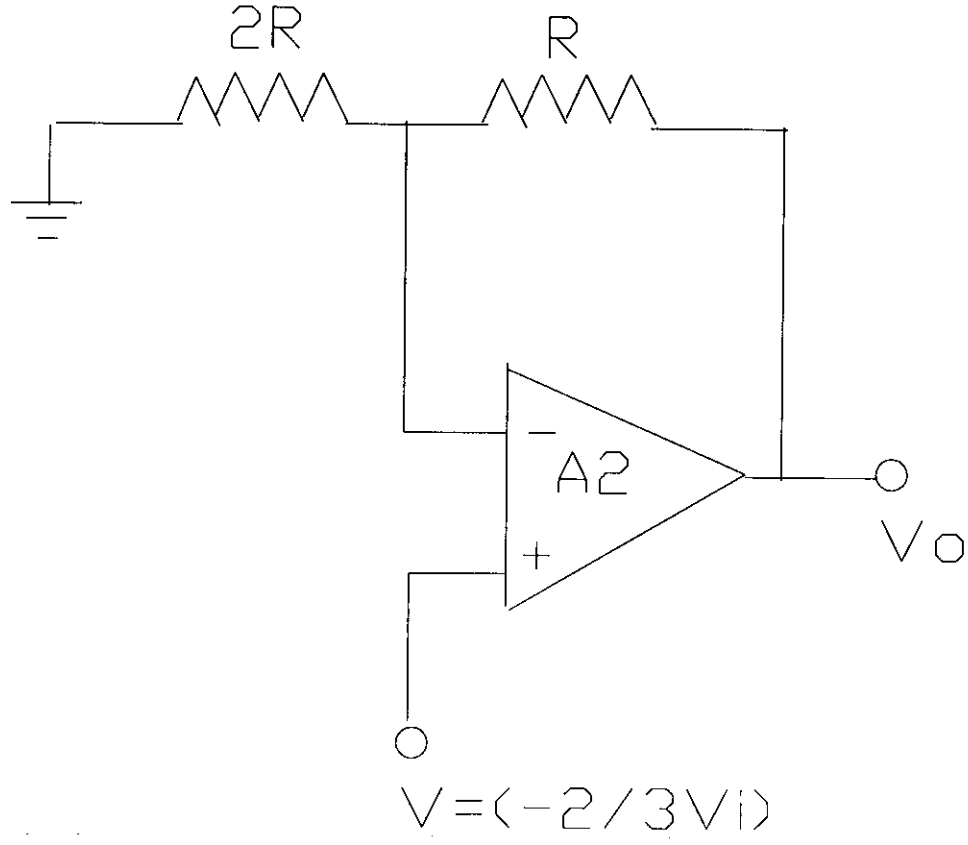
Precision Rectifier

Equivalent Circuit for Vi < 0

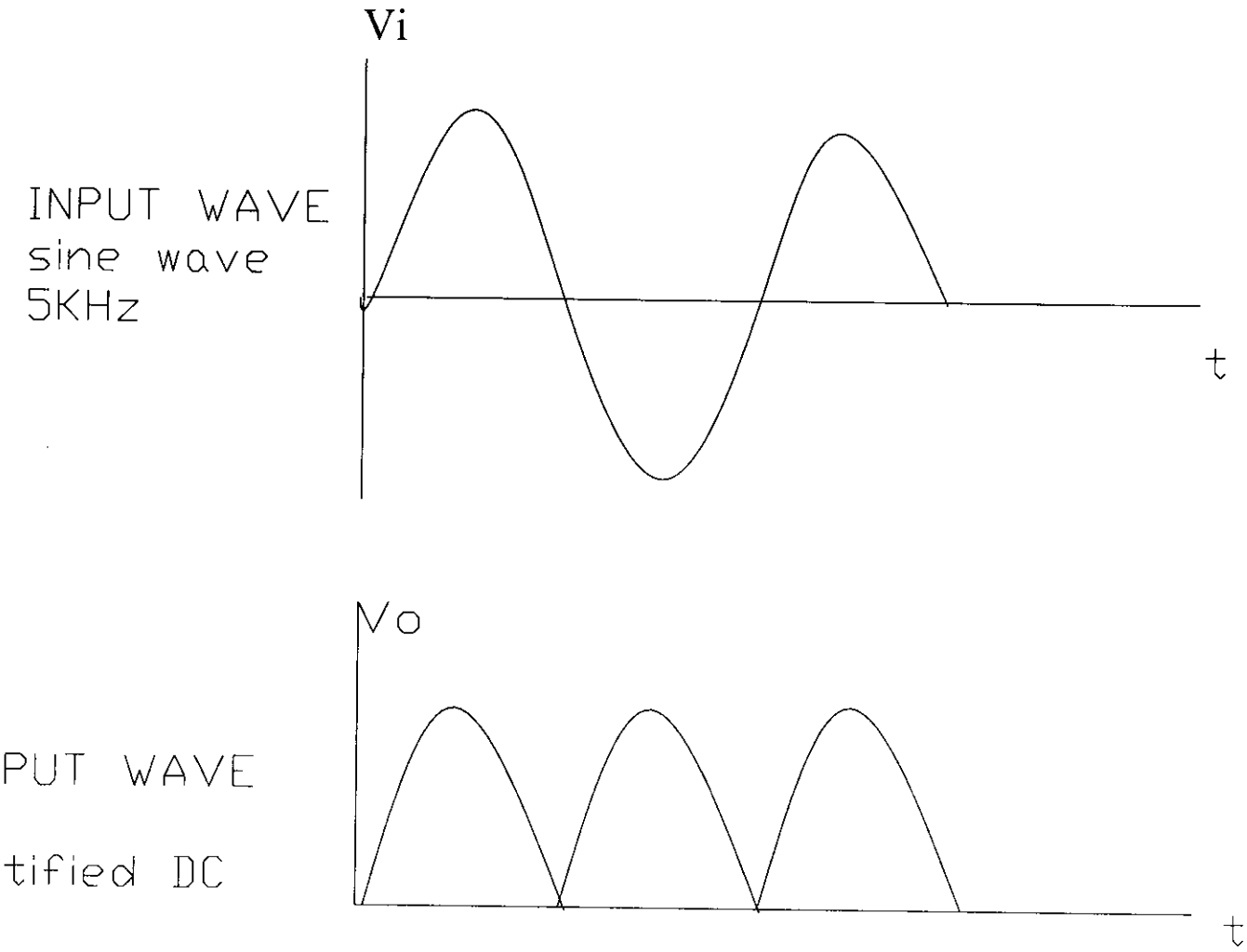


Precision Rectifier

Equivalent Circuit for Vi < 0



WAVE FORMS



SCHMITT TRIGGER :

It is also named as a Regenerative Comparator. If the positive feedback is added to the comparator circuit, gain can be increased greatly. Consequently, the transfer curve of the comparator becomes more close to ideal curve. Theoretically, if the loops gain $-\beta A_{oL}$ is adjusted to unity, then the gain with feedback, A_{vf} becomes infinite. This result is an abrupt (Zero rise time) transition between the extreme values of the output voltage. In practical circuits, however, it may not be possible to maintain loop gain exactly equal to unity for a long time because of the supply voltage and temperature variation. So a value greater than unity is chosen. This also gives an output waveform virtually discontinuous at the comparison voltage. This circuit, however, now exhibit a phenomenon called hysteresis of backlash.

The figure shows a regenerative comparator. The input is applied to the (-) input terminal and the feedback voltage to the (+) input terminal. The input V_i triggers the output V_0 every time it exceeds certain voltage levels. These voltage levels are called Upper threshold Voltage(V_{UT})and lower Threshold Voltage (V_{LT}). The hysteresis width is the difference between these two threshold voltages. These threshold voltages are calculated as follows.

Suppose the output $V_0 = +V_{sat}$, The voltage at(+) input terminal will be

$$V_{ref} + R_2/(R_1+R_2)(V_{sat} - V_{ref}) = V_{UT}$$

This voltage is called upper threshold Voltage. As long as V_i is less than V_{ut}' the output V0 remains constant. at $+V_{sat}$. When V_i is just greater than V_{ut}, the output regeneratively switches to $-V_{sat}$ and remains at this level as long as $V_i > V_{UT}$ as shown in the figure.

For $V_0 = -V_{sat}$, the voltage ar the (+) input terminal is

$$V_{ref} - R_2(V_{sat}+V_{ref})/R1 + R2 = V_{LT}$$

If an input sinusoidal of frequency F = 1/Tis applied to such a comparator, a symmetrical square wave is obtained at the output. The vertical edge of the output waveform however will not occur at the time the sine wave passes through zero but is shifted in phase by θ where $\sin \theta = V_{UT}/V_m$ and V_m is the peak sinusoidal voltage.

Thus for the DC voltage fed to the Schmitt trigger, logic level 1 or 0 output is formed accordingly. The output is fed to the micro controller.

The IC's used in our project for the OP AMP's operations is

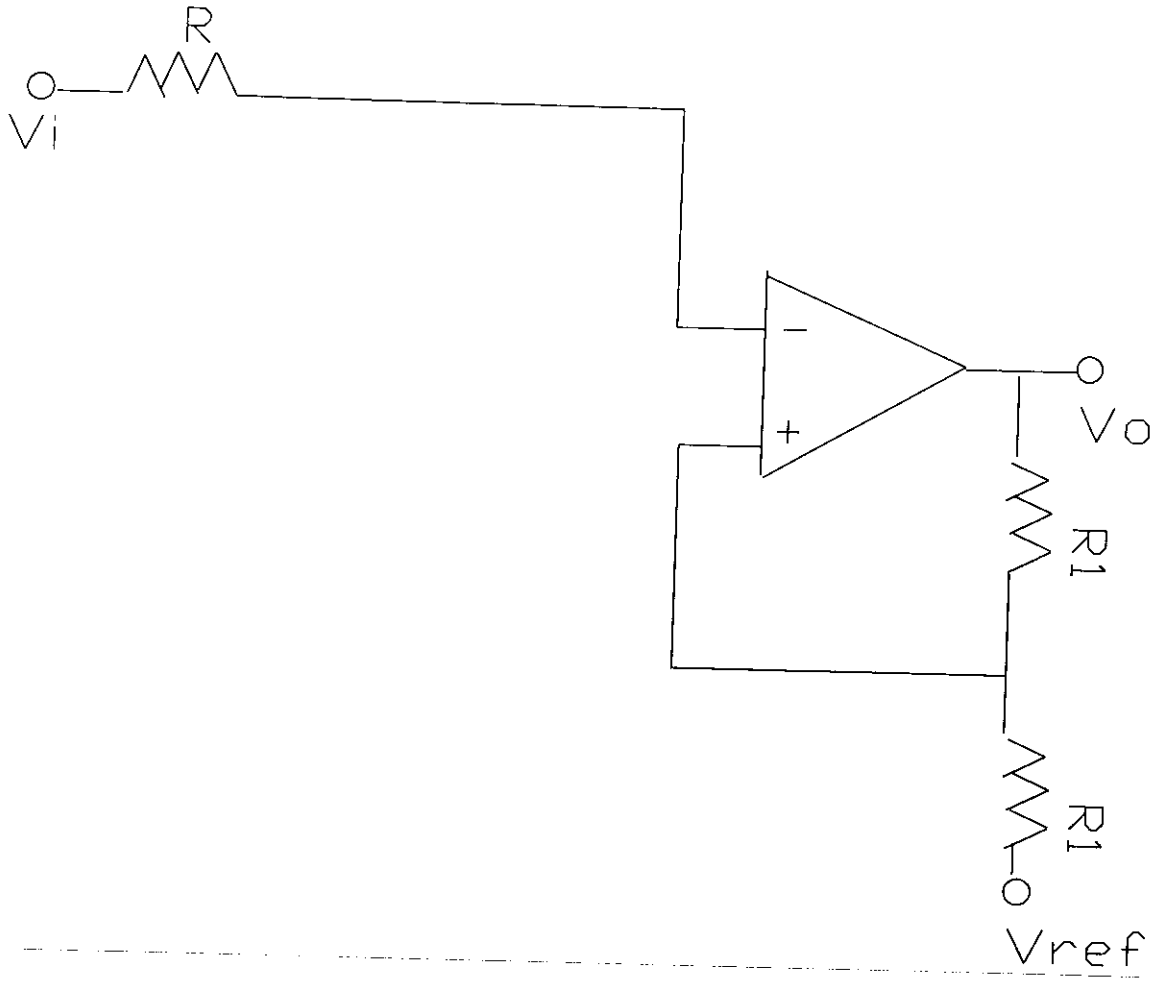
LM 324

LM 324 DESCRIPTION :

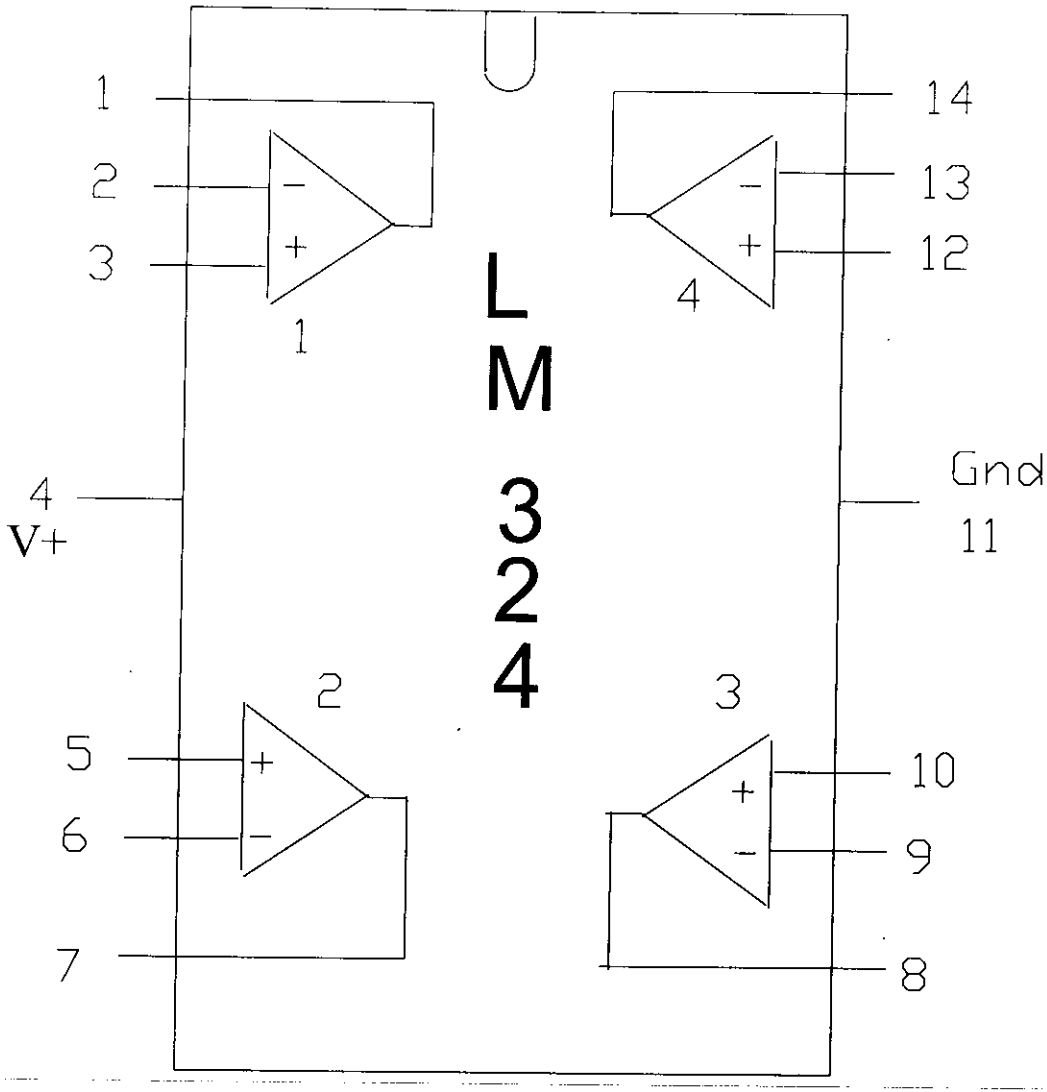
LM 324 IC consists of 4 independent, high gain, internally frequency compensated operational amplifiers which are designed specifically to operate from a single power supply
Over a wide range of voltages.

It can operate at 32 V DC or $\pm$ 16 V DC supply voltages.

INVERTING SCHMITT TRIGGER



LM 324 OP AMP PIN OUT



SENSING POINTS :

The sensing points for the train are provided at two different points for our project. A distance of 10 KM for the real system may separate them.

3.4 MICRO CONTROLLER :

The **ATMEL 89C51** Micro controller is used as an evaluator for the system. The Schmitt trigger output is fed as a signal to micro controller. The central microcomputer will sense every dip (wheel crossing) caused due to the train movement between the rail's transmitter and receiver, and it increments the **IN COUNT** register for every wheel. Similar counts are made in **OUT COUNT** register for every out going axle of the train from the other destination point.

The micro controller is programmed in such a manner that it will sense s every wheel and stores the counts at either ends. All the Signal drive circuits are explained with LED's in out project. As soon as the wheel starts passing, various sequences will make the required signal.

CHAPTER 4

Software Coding

MICRO CONTROLLER : ATMEL 89C51

```
        org    0000h
        ljmp   main

        org    000bh
        ljmp   timer0

        org    001bh
        ljmp   timer1


        org    0003h
        ljmp   init_key1

        org    0013h
        ljmp   init_key2


        org    100h
main:   mov    p1,#00h
        setb   p1.2
        setb   p1.5
        mov    p2,#0ffh
        mov    30h,#00h
        mov    31h,#00h
        mov    50h,#00h          ;working   program
        mov    a,#088h
        mov    r0,#01bh
        movx   @r0,a
        mov    a,#0ffh
        mov    r0,#19h
        movx   @r0,a
        mov    a,#000h
        mov    r0,#01ah
        movx   @r0,a
        lcall  delay
        mov    r0,#018h
```

```
        mov    a,#038h
        movx   @r0,a
        lcall  clock
        lcall  del
        mov    a,#008h
        mov    r0,#018h
        movx   @r0,a
        lcall  clock
        lcall  del
        mov    a,#001h
        mov    r0,#018h
        movx   @r0,a
        lcall  clock
        lcall  del
        mov    a,#006h
        mov    r0,#018h
        movx   @r0,a
        lcall  clock
        lcall  del
        mov    r0,#018h
        mov    a,#00ch
        movx   @r0,a
        lcall  clock
        lcall  del
        mov    a,#045h
        movx   @r0,a
        lcall  clock
        lcall  del
in_dis: mov    a,#080h        ; display blank
        mov    r0,#18h
        movx   @r0,a
        lcall  clock
        mov    r6,#"K"
        lcall  dr_w
        mov    r6,#"U"
        lcall  dr_w
        mov    r6,#"M"
        lcall  dr_w
        mov    r6,#"A"
        lcall  dr_w
        mov    r6,#"R"
        lcall  dr_w
        mov    r6,#"A"
        lcall  dr_w
        mov    r6,#"G"
        lcall  dr_w
        mov    r6,#"U"
        lcall  dr_w
```

```
mov    r6,#"R"
lcall  dr_w
mov    r6,#"U"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    a,#0c0h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#"C"
lcall  dr_w
mov    r6,#"O"
lcall  dr_w
mov    r6,#"L"
lcall  dr_w
mov    r6,#"L"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#"G"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"O"
lcall  dr_w
mov    r6,#"F"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"T"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#"C"
lcall  dr_w
```

```
mov    r6,#"H"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
mov    a,#080h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#"D"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#"P"
lcall  dr_w
mov    r6,#"A"
lcall  dr_w
mov    r6,#"R"
lcall  dr_w
mov    r6,#"T"
lcall  dr_w
mov    r6,#"M"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#"N"
lcall  dr_w
mov    r6,#"T"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"O"
lcall  dr_w
mov    r6,#"F"
lcall  dr_w
```

```
mov    r6,#" "
lcall  dr_w
mov    a,#0c0h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"*"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#"E"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"*"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
```

```
mov    a,#080h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"A"
lcall  dr_w
mov    r6,#"U"
lcall  dr_w
mov    r6,#"T"
lcall  dr_w
mov    r6,#"O"
lcall  dr_w
mov    r6,#"M"
lcall  dr_w
mov    r6,#"A"
lcall  dr_w
mov    r6,#"T"
lcall  dr_w
mov    r6,#"I"
lcall  dr_w
mov    r6,#"C"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    a,#0c0h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#"R"
lcall  dr_w
mov    r6,#"A"
lcall  dr_w
mov    r6,#"I"
lcall  dr_w
mov    r6,#"L"
lcall  dr_w
```

```
mov    r6,#"W"
lcall  dr_w
mov    r6,#"A"
lcall  dr_w
mov    r6,#"Y"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"S"
lcall  dr_w
mov    r6,#"I"
lcall  dr_w
mov    r6,#"G"
lcall  dr_w
mov    r6,#"N"
lcall  dr_w
mov    r6,#"A"
lcall  dr_w
mov    r6,#"L"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
setb   ea
setb   ex0
setb   ex1
mov    p3,#0ffh
mov    60h,#00h
mov    51h,#00h
mov    50h,#00h
mov    70h,#00h
mov    3bh,#00h
mov    3ch,#00h
mov    8ch,#0bh      ; th0(timer\counter0 high byte)
mov    8ah,#0dbh     ; tl0( ,,     ,,    low byte)
setb   0afh          ; enable the interrupt sourse
setb   0a9h          ; enable timer0 overflow interrupt
```

```
        mov    89h,#11h      ; tmod(timer\counter0 mode control reg)
        mov    8dh,#0bh
        mov    8bh,#0dbh
        setb   0afh
        setb   0abh          ; enable the interrupt sourse
        mov    a,p2
        anl    a,#01h
        cjne   a,#00h,kkkkk
        lcall  dis_tr1
        clr    p1.2
        setb   p1.0
        lcall  del
        lcall  del
        lcall  del
        ljmp   key
kkkkk:  cjne   a,#01h,kkkkk1
        lcall  dis_tr2
        clr    p1.5
        setb   p1.3
        lcall  del
        lcall  del
        lcall  del
kkkkk1:    ljmp   key



key:    jb     p3.4,key1
        lcall  del
        inc    60h
        lcall  hex_dec1
        lcall  set1
        lcall  del
        setb   8eh
        ljmp   key

key1:     jb     p3.5,key2
        lcall  del
        inc    60h
        lcall  hex_dec1
        lcall  set2
        lcall  del
        setb   8eh
key2:     ljmp   key



timer0:    push   psw
```

```
        inc   3bh
        mov   a,3bh
        subb  a,#0aah
        jc    jreti
        clr   ex0
        clr   ex1
        mov   3bh,#00h
jreti:  lcall t_load
        pop   psw
        reti

t_load: mov   8ch,#0bh
        mov   8ah,#0dbh
        ret

timer1: push  psw
        inc   3ch
        mov   a,3ch
        subb  a,#0aah
        jc    jretti
        lcall comp
        mov   3ch,#00h
jretti: lcall tload
        pop   psw
        reti

tload:  mov   8ch,#0bh
        mov   8ah,#0dbh
        ret



init_key1:
      lcall del
      inc   70h
      lcall hex_dec2
      lcall set1
      lcall del
      clr   p1.0
      setb  p1.2
      setb  p1.6
      mov   c,ie0
      jc    kkkk
      setb  8ch
kkkk: reti
init_key2:
      lcall del
```

```
        inc   70h
        clr   p1.3
        setb  p1.5
        lcall hex_dec2
        lcall set2
        setb  p1.6
        lcall del
        mov   c,ie1
        jc    llll
        setb  8ch
llll:   reti

comp:   mov   a,70h
        mov   b,60h
        subb  a,b
        jz    dis1
        jnc   dis2
        jc    dis3
        ret


dis1:   ljmp  cross
        clr   p1.6
        mov   60h,#00h
        mov   70h,#00h
        ret

dis2:   ljmp  notcros
        mov   60h,#00h
        mov   70h,#00h
        ret

dis3:   ljmp  mismatch
        mov   60h,#00h
        mov   70h,#00h
        ret



hex_dec1:  mov    a,60h
           mov    b,#64h
           div    ab
           mov    20h,a     ;100
           mov    a,b
           mov    b,#0ah
           div    ab
           mov    21h,a     ;10
```

```
        mov    22h,b     ;1
        ret
hex_dec2:  mov    a,70h
        mov    b,#64h
        div    ab
        mov    23h,a     ;100
        mov    a,b
        mov    b,#0ah
        div    ab
        mov    24h,a     ;10
        mov    25h,b     ;1
        ret

set1:
        mov    a,#080h          ; display blank
        mov    r0,#18h
        movx   @r0,a
        lcall  clock
        mov    r6,#"T"
        lcall  dr_w
        mov    r6,#"r"
        lcall  dr_w
        mov    r6,#"a"
        lcall  dr_w
        mov    r6,#"i"
        lcall  dr_w
        mov    r6,#"n"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#"V"
        lcall  dr_w
        mov    r6,#"i"
        lcall  dr_w
        mov    r6,#"a"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#"A"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    a,23h
        add    a,#30h
        mov    r6,a
        lcall  dr_w
```

```
mov    a,24h
add    a,#30h
mov    r6,a
lcall  dr_w
mov    a,25h
add    a,#30h
mov    r6,a
lcall  dr_w
mov    a,#0c0h      ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#"T"
lcall  dr_w
mov    r6,#"r"
lcall  dr_w
mov    r6,#"a"
lcall  dr_w
mov    r6,#"i"
lcall  dr_w
mov    r6,#"n"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"V"
lcall  dr_w
mov    r6,#"i"
lcall  dr_w
mov    r6,#"a"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"B"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    a,20h
add    a,#30h
mov    r6,a
lcall  dr_w
mov    a,21h
add    a,#30h
mov    r6,a
lcall  dr_w
mov    a,22h
add    a,#30h
```

```
        mov     r6,a
        lcall   dr_w
        ret


set2:
        mov     a,#080h         ; display blank
        mov     r0,#18h
        movx    @r0,a
        lcall   clock
        mov     r6,#"T"
        lcall   dr_w
        mov     r6,#"r"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"n"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"V"
        lcall   dr_w
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"B"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     a,23h
        add     a,#30h
        mov     r6,a
        lcall   dr_w
        mov     a,24h
        add     a,#30h
        mov     r6,a
        lcall   dr_w
        mov     a,25h
        add     a,#30h
        mov     r6,a
        lcall   dr_w
```

```
        mov     a,#0c0h     ; display blank
        mov     r0,#18h
        movx    @r0,a
        lcall   clock
        mov     r6,#"T"
        lcall   dr_w
        mov     r6,#"r"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"n"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"V"
        lcall   dr_w
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"A"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     a,20h
        add     a,#30h
        mov     r6,a
        lcall   dr_w
        mov     a,21h
        add     a,#30h
        mov     r6,a
        lcall   dr_w
        mov     a,22h
        add     a,#30h
        mov     r6,a
        lcall   dr_w
        ret


cross:  mov     a,#080h     ; display blank
        mov     r0,#18h
```

```
movx   @r0,a
lcall  clock
mov    r6,#"T"
lcall  dr_w
mov    r6,#"r"
lcall  dr_w
mov    r6,#"a"
lcall  dr_w
mov    r6,#"i"
lcall  dr_w
mov    r6,#"n"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"i"
lcall  dr_w
mov    r6,#"s"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    a,#0c0h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"c"
lcall  dr_w
mov    r6,#"r"
lcall  dr_w
```

```
mov    r6,#"o"
lcall  dr_w
mov    r6,#"s"
lcall  dr_w
mov    r6,#"s"
lcall  dr_w
mov    r6,#"e"
lcall  dr_w
mov    r6,#"d"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
setb   ex0
setb   ex1
clr    8ch
clr    8eh
mov    70h,#00h
mov    60h,#00h
mov    20h,#00h
mov    21h,#00h
mov    22h,#00h
mov    23h,#00h
mov    24h,#00h
mov    25h,#00h
clr    acc
mov    a,p2
anl    a,#01h
cjne   a,#00h,nnnxt
clr    p1.6
clr    p1.2
setb   p1.1
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
lcall  del
```

```
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        clr     p1.1
        lcall   del
        lcall   del
        setb    p1.0
        ret
nnnxt:    cjne    a,#01h,nnnnt
        clr     p1.6
        clr     p1.0
        clr     p1.5
        setb    p1.4
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        lcall   del
        clr     p1.4
        lcall   del
        lcall   del
        setb    p1.3
nnnnt:    ret

notcros:   mov     a,#080h        ; display blank
        mov     r0,#18h
        movx    @r0,a
        lcall   clock
        mov     r6,#"T"
        lcall   dr_w
        mov     r6,#"r"
        lcall   dr_w
        mov     r6,#"a"
```

```
        lcall  dr_w
        mov    r6,#"i"
        lcall  dr_w
        mov    r6,#"n"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#"i"
        lcall  dr_w
        mov    r6,#"s"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#"n"
        lcall  dr_w
        mov    r6,#"o"
        lcall  dr_w
        mov    r6,#"t"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    a,#0c0h          ; display blank
        mov    r0,#18h
        movx   @r0,a
        lcall  clock
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#"c"
        lcall  dr_w
        mov    r6,#"r"
        lcall  dr_w
        mov    r6,#"o"
        lcall  dr_w
        mov    r6,#"s"
        lcall  dr_w
        mov    r6,#"s"
        lcall  dr_w
        mov    r6,#"e"
```

```
        lcall   dr_w
        mov     r6,#"d"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        setb    ex0
        setb    ex1
        clr     8ch
        clr     8eh
        mov     70h,#00h
        mov     60h,#00h
        mov     20h,#00h
        mov     21h,#00h
        mov     22h,#00h
        mov     23h,#00h
        mov     24h,#00h
        mov     25h,#00h
        ret


mismatch:  mov     a,#080h        ; display blank
        mov     r0,#18h
        movx    @r0,a
        lcall   clock
        mov     r6,#"C"
        lcall   dr_w
        mov     r6,#"o"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
        mov     r6,#"c"
        lcall   dr_w
        mov     r6,#"h"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
```

```
mov    r6,#"A"
lcall  dr_w
mov    r6,#"d"
lcall  dr_w
mov    r6,#"d"
lcall  dr_w
mov    r6,#"e"
lcall  dr_w
mov    r6,#"d"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    a,#0c0h          ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"R"
lcall  dr_w
mov    r6,#"e"
lcall  dr_w
mov    r6,#"s"
lcall  dr_w
mov    r6,#"e"
lcall  dr_w
mov    r6,#"t"
lcall  dr_w
mov    r6,#"?"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
```

```
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        setb    ex0
        setb    ex1
        clr     8ch
        clr     8eh
        mov     70h,#00h
        mov     60h,#00h
        mov     20h,#00h
        mov     21h,#00h
        mov     22h,#00h
        mov     23h,#00h
        mov     24h,#00h
        mov     25h,#00h
        ret

dis_tr1:  mov     a,#080h       ; display blank
        mov     r0,#18h
        movx    @r0,a
        lcall   clock
        mov     r6,#"A"
        lcall   dr_w
        mov     r6,#"l"
        lcall   dr_w
        mov     r6,#"l"
        lcall   dr_w
        mov     r6,#"o"
        lcall   dr_w
        mov     r6,#"w"
        lcall   dr_w
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"n"
        lcall   dr_w
        mov     r6,#"g"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"T"
        lcall   dr_w
        mov     r6,#"r"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
```

```
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"n"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     a,#0c0h         ; display blank
        mov     r0,#18h
        movx    @r0,a
        lcall   clock
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"V"
        lcall   dr_w
        mov     r6,#"i"
        lcall   dr_w
        mov     r6,#"a"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#"A"
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        mov     r6,#" "
        lcall   dr_w
        ret

dis_tr2:  mov     a,#080h       ; display blank
        mov     r0,#18h
```

```
movx   @r0,a
lcall  clock
mov    r6,#"A"
lcall  dr_w
mov    r6,#"l"
lcall  dr_w
mov    r6,#"l"
lcall  dr_w
mov    r6,#"o"
lcall  dr_w
mov    r6,#"w"
lcall  dr_w
mov    r6,#"i"
lcall  dr_w
mov    r6,#"n"
lcall  dr_w
mov    r6,#"g"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"T"
lcall  dr_w
mov    r6,#"r"
lcall  dr_w
mov    r6,#"a"
lcall  dr_w
mov    r6,#"i"
lcall  dr_w
mov    r6,#"n"
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    a,#0c0h        ; display blank
mov    r0,#18h
movx   @r0,a
lcall  clock
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#" "
lcall  dr_w
mov    r6,#"V"
lcall  dr_w
```

```
        mov    r6,#"i"
        lcall  dr_w
        mov    r6,#"a"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#"B"
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        mov    r6,#" "
        lcall  dr_w
        ret

delay:   mov    r5,#010h
lmp:     mov    r4,#010h
loopd:   djnz   r4,loopd
         djnz   r5,lmp
         ret

del:     mov    r1,#0ffh
loops1:  mov    r2,#0ffh
loopd2:  djnz   r2,loopd2
         djnz   r1,loops1
         ret

dr_w:    mov    r0,#18h
        mov    a,r6
        movx   @r0,a
        lcall  clock1
        ret

clock:   mov    r0,#1ah
        mov    a,#04h
        movx   @r0,a
```

```
        lcall   delay
        mov     a,#00h
        movx    @r0,a
        lcall   delay
        ret


clock1:   mov     r0,#1ah
        mov     a,#05h
        movx    @r0,a
        lcall   delay
        mov     a,#01h
        movx    @r0,a
        lcall   delay
        ret
```

CHAPTER 5

Sequence of Operation

Sequence : 1

- If the train is needed to be allowed through destination 'A', then by selecting the **'Train Select'** switch the corresponding signal is activated and made Green.
- Every count of the wheel is sensed and the counter is incremented by 1
- As soon as the first wheel is passed, the signal is made red.
- Maximum time between counts is 10 Sec
- Similar counts are made at the other destination point
- Micro Controller compares both the counts
- If they tallies then it displays **"TRAIN CROSSED"** and activate the signal at 'A' to yellow and then Green to allow next train through that route.

Similar sequence in the opposite side can be made.

Implications:

- The entire train is passed
- The gap is maintained between two trains
- No leave over of any coach in between
- Opposing train accidents are prevented
- Circuit clearance is made during shunting

Sequence : 2

- If the train has missed a coach in between the points, then due to mismatch in the counts the micro controller, then signal stays in RED

- Since counts are not tallied it displays
  "TRAIN NOT CROSSED"

Implications:

- If due to some accidents, any leave over or derailment or accident has occurred then the accident of the next train with the previous is prevented

- The display in the stationmaster's room will indicate the problem to him.

Sequence : 3

- If the train has an added coach in between the points, then due to mismatch in the counts the micro controller. then signal stays in RED

- Since counts are not tallied it displays and it is more than the input it displays
  "COACH ADDED RESET ?"

Sequence : 4

- If an unmanned level crossing is present in between the points, then as soon as the train enters either of the points, a loud Siren is activated in the unmanned crossing indicating that a train is approaching the crossing

- The siren is off when the train has crossed the other point.

Implication :

Accidents at unmanned level crossings are avoided

CHAPTER 6

Conclusion

The project has been successfully completed with implementing the hardware and software logics. The system is found to give best results. Thus the objective of automation in the railway signaling is achieved.

This automation of the railway signaling includes the following advantages

- ✓ Trouble free circuit
- ✓ Easily programmable and reprogrammable
- ✓ Suitable for all operating conditions
- ✓ Compact in size
- ✓ Cost has been drastically reduced

Thus the automation of railway signaling is done using micro controller.

FUTURE PROSPECTS

This system is made as prototype for the signaling system. The kit is to be kept at the stationmaster's room and this is for a single channel operation for

Bi – directional trains at a single track.

Making the components sturdy and rigid to suit for practical applications can enlarge the system. Consecutive arrangements like this for the entire length of the track and thereby having this system as a whole for the entire railway signaling can be done. A master microcomputer is essential of any case and slight modifications in the system will make the system suitable.

References

1. Handbook of 8 Bit micro controller,

   Intel corporation, 1989 USA,.

2. Coughlin, R.F. and F. F. Driscall, Operational Amplifiers and Linear Integrated Circuits, Prentice – Hall Inc., N.J., 1977

3. Ruthwisky,G.B., Handbook of Integrated Circuits Operational Amplifier, Prentice – Hall, Englewood Cliffs, N.J., 1975

4. D. Roy Choudhury and Shail Jain, Linear Integrated Circuits, New Age International P.Ltd,1991.

5. Ramesh S. Gaonkar, Microprocessor Architecture, Programming and Applications, Penram International., 1997.

6. S&T workshop, Southern Railways, Coimbatore.

7. WWW. Atmel.com

8. WWW.microcontrollers.com

9. WWW. Icmaster.com