MICROCONTROLLER BASED PROGRAMMABLE TIMER

PROJECT WORK

SUBMITTED BY

T. S. BALAJI
T. MUTHUKUMAR
K. RAMADAS

UNDER THE GUIDANCE OF

Miss. P. Arul Selvi, M.E.

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF BACHELOR OF ENGINEERING IN ELECTRICAL & ELECTRONICS ENGINEERING OF THE BHARATHIAR UNIVERSITY COIMBATORE

1994 - 1995

Department of Electrical & Electronics Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE 641 006

P207

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Kumaraguru College of Technology

COIMBATORE - 641 006.

CERTIFICATE

This is to certify that the Project Report entitled

MICRO CONTROLLER BASED PROGRAMMABLE TIMER

has been submitted by

Mr.....

in partial fulfilment for the award of the degree of

Bachelor of Engineering in Electrical and Electronics Engineering

Branch of the Bharathiar University, Coimbatore during the academic year 1994-95

Dr. K. A. PALANISWAMY, B.E.,M.Sc.(Engg).,Ph.D
M.I.E.(India),F.I.E
Professor and Head
Department of Electrical and Electronics Engineering
Kumaraguru College of Technology
Coimbatore 641 006

.....
Guide

.....
Head of the Department

Certified that the candidate with University Register No. .....

was examined in project work Viva-Voce by us on .....

ACKNOWLEDGEMENT

We are highly grateful to **Miss P. ARUL SELVI,** M.E ., Associate Lecturer in Electrical and Electronics Engineering Department for her able guidance and readiness to extend all the required help to do this project.

We express our sincere thanks to our beloved professor **DR. K.A. PALANISWAMY** B.E., M.Sc (ENGG), Ph.D., M.I.S.T.E., C.ENG (I), F.I.E., Head of the Department of Electrical and Electronics Engineering for his constant inspiration and continued encouragement to do this project.

We thank our principal **DR. S. SUBRAMANIAN** B.E., M.Sc., (ENGG)., Ph.D., SMIEEE for providing all facilities in the college to carry out this project.

We express our gratitude to **MR. E. PALANIAPPAN** B.E., **MR. D. SENTHIL KUMAR** B.E., AGT ELECTRONICS, Coimbatore and **MR. E. JAGANATHAN** B.E., GURUKULAM ELECTRONICS for their help in the successful completion of this project.

Thanks are due to all those who directly or indirectly helped us for the completion of this project. Thanks are also due to members of staff in Electrical and Electronics Engineering Department for their continued support and help.

SYNOPSIS

A microcontroller based system has many advantages like high speed, reliability, userfriendly, versatility and accuracy. In this system the microcontroller operations are co-ordinated using Realtime clock and external memory. The realtime clock chip provides precise clock pulse which enhances the accuracy.

In this project, an innovative attempt is made to incorporate multiple facilities using microcontroller devices to design a programmable timer. To carry out different operations in the programmable timer, different mode are set in micro controller through software and proper accessing of these modes and other operations are carried over by 4 different keys that are provided externally.

The designed hardware circuit in fabricated and a software in assembler language is developed. In view of the extended facilities provided by the software, the timer is suitable for application in industries where time controlled operations are involved.

CONTENTS

CHAPTER-I

INTRODUCTION

Timers are generally of the type (a) Electromechanical (b) Analog (c) Digital in nature. The programmable timer developed by us is digital in nature and is controlled by dedicated micro controller which is having a very high level of reliability and accuracy.

The following facilities are provided through software namely.

1. Time setting mode : This mode is used for setting the real time clock to current time , date, day etc.

2. Alarm setting mode : The alarm setting mode is used for setting the time and duration for which bell has to ring multiple inputs can be given.

3. Alarm deleting mode : This mode is used for deleting the unnecessary alarm setting.

4. Holiday setting mode : This mode is used for turning the alarm off on holidays.

5. Holiday Deleting mode : This mode is used for deleting the unnecessary holiday inputs.

6. Day setting mode : This allows the user to turn the alarm setting on or off for that particular day.

7. Reset mode: This mode is used for resetting the timer to initial position.

8. Display mode : This mode is used for switching from time option to calendar option & vice versa. The external key control is through a set of four (4) keys namely.

1. submenu key : This key is used for setting year, month,hours, say e.t.c to a particular value after a particular mode is selected.

2. increment key : This key is used for incrementing the year, month e.t.c to the required value.

3. Toggle / Accept key : This key used for toggling between time display & calendar display modes. In all other modes it is used as accept key. Thus this key prevents accidental changing of the time.

4. mode changing key : This key is used for changing from one mode to another mode.

CHAPTER II

MICRO-CONTROLLER

INTRODUCTION :

The 8051 family of products is based on the industry standard for 8-bit high performance micro-controller. The architecture for the family has been optimized for sequential real time control applications. The 8051 family of products are used in a wide range of applications from those that are relatively simple to applications in medical instrumentation and automobile control systems. All of the devices included in the family are available in versions that have either internal ROM, EPROM, or CPU only.

In this project 8031 micro-controller which does not have any internal program memory is used for controlling the programmable timer Fig. 2.1 gives the block diagram of 8031.

2.2 HARDWARE DESCRIPTION :

The hardwares used in a micro-controller are

1. 8-bit CPU optimized for control applications.
2. Extensive Boolean processing (Single bit logic) capabilities
3. 32bit-bi-directional and individual addressable i/o lines.
4. 128 bytes of on chip data RAM
5. Two 16 bit Timer / Counters.
6. Full Duplex UART.
7. Source Interrupt structure with two priority levels.
8. On chip clock oscillator.
9. 64 KB programmable memory address space.
10. 64 KB data memory address space
11. 40 pin DIP

The port drives and how they function both as ports as ports and for port 0 and port 2 in bus operation, the timers/counters, the serial interface, the interrupt system and reset are described below.

2.3 SPECIAL FUNCTION REGISTERS :

2.3.1 ACCUMULATOR :

ACC is the accumulator register. The mnemonics for accumulator specific instructions however refer to the accumulator simply as A.

2.3.2 B REGISTER :

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

2.3.3 PROGRAM STATUS WORD:

MSB							LSB
CY	AC	FO	RSI	RSO	OV	-	P

SYMBOL	POSITION	NAME AND SIGNIFICANCE
CY	PSW.7	Carry flag
AC	PSW.6	auxilliary carry flag (for BCD operations)
FO	PSW.5	flag 0
RS1	PSW.4	register bank select control bits 0 set
RSO	PSW.3	cleared by software to determine working register bank.
OV	PSW.2	overflow flag.
-	PSW.1	user definable flag.
P	PSW.0	parity flag.

2.3.4. STACK POINTER :

The stack pointer register is 8 bits wide. PUSH and CALL (POP) executions can be done.

2.3.5. DATA POINTER :

DPTR consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16 bit address.

2.3.6. PORTS 0-3 :

PO,P1,P2 and p3 are the SFR latches of port 0,1,2 and 3 respectively.

2.4 PORT STRUCTURES AND OPERATIONS :

All ports in 8051 are bi-directional. Each consists of a latch as an output driver and an input buffer.

The output drivers of ports 0 and 2 and the input buffer of port 0 are used to access the external memory. In this application port 0 output, the low byte of external memory address, time multiplexed with the byte of external memory address, time multiplexed with the byte being written or read.

All the port 3 pins are multi-functional. They are port pins, but also serve the functions of varied features as listed below.

PORT PIN	ALTERNATE FUNCTION
P3.0	RXD (serial input)
P3.1	TXD (serial output port)
P3.2	INTO (external interrupt)
P3.3	INT1 (external interrupt)
P3.4	TO (Timer/Counter 0)
P3.5	T1 (Timer/Counter 2)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

2.5 PORT LOADING AND INTERFACING :

The output buffers of port 1,2 and 3 can each drive 4 LS TTL inputs. These ports on HMOS versions can be driven in

a normal manner by any TTL or NMOS circuit. Both NMOS and CHMOS pins can be driven by open collector and open drain outputs, but 0 to 1 transition will not be fast.

2.6 TIMER/COUNTER :

In the timer functions, the register is incremented in every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the counter function a register is incremented in response to a 1 or 0 transition at the corresponding external input pin t0 or t1. The timer 0 and 1 have four operating modes to select.

MSB							LSB
GATE	C/T	M1	MO	GATE	C/T	M1	MO
TIMER 1				TIMER 0			

GATE - Gating control when set. Timer/counter 'x' is enabled only while TNTX pin is high and 'IRX' control pin is set. When cleared Timer is enable whenever 'TRX' control bit is set.

C/T - Timer/Counter selector Clear for timer operation (input from internal system clock). Set for counter operation.

M1	M0	OPERATION MODE
0	0	8048 Timer TLX serves as 5 bit pre-scaler.
0	1	16 - bit timer/Counter. THX and TLX are cascaded. There is no pre-scaler.
1	0	8 bit auto - reload timer/Counter THX holds a value which is to be reloaded into TLX each time it overflows.
1	1	(Timer 0) TLO is an 8 bit controlled by standard timer 0 control bits. THO is an 8 bit timer only, controlled by timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

2.7.1 MODE 0 :

Putting either timer into mode 0 makes it look like a timer, which is an 8 bit counter with a divide by 32 prescaler.

In this mode the timer register is configured as a 32 bit register. As the counter rolls over, from all 1s to all 0s it sets the timer interrupt flag TFI. The counter input is enabled to the timer TR1 = 1 and either GATE = 0 or INT1 =1. Setting GATE = 1 allows the lower to be controlled by external input INT1 to facilitate pulse width measurements. TRI is a control bit in the special function register TCON GATE in TMOD.

The 13 bit register consists of all 8 bits of TH1 as the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag TR1 does not clear the register.

Mode 0 operation is the same for timer 0 and timer 1. Substitute TRO, TFO and INTO for the corresponding timer 1 signals in figure. There are two different gate bits, one for timer 1 (TMOD.7) and the other for timer 0 (TMOD.3).

2.7.2. MODE 1 :

Mode 1 is the same as mode 0 except that the timer register is being run with all 16 bits.

2.7.3 MODE 2:

The mode 2 configuration , the timer register is an 8 bit (TL1) with automatic reload.

2.7.4 MODE 3.

Timer 1 in mode 1 simply holds its count. The effect is the same as setting TR1 = 0.

2.8 STANDARD SERIAL INTERFACE :

MSB LSB

TF1 TR1 TFO TRO IE1 1I1 IE0 IT0

SYMBOL	POSITION	NAME AND SIGNIFICANCE
TF1	TCON.7	Timer 1 overflow flag.
TR1	TCON.6	Timer 1 run control bit set/cleared by software to turn Timer/Counter ON/OFF.
TFO	TCON.5	Timer 0 overflow flag set by hardware on timer/counter overflow.
TRO	TCON.4	Timer 0 run control bit.
IE1	TCON.3	Interrupt 1 edge flag set by hardware when external interrupt edge detected. Cleared when interrupt processed.

SYMBOL	POSITION	NAME AND SIGNIFICANCE
IT1	TCON.2	Interrupt 1 type control bit/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 edge flag set by hardware when external interrupt edge detected. Cleared when interrupt Processed.
IT0	TCON.0	Interrupt 0 type control bit set/cleared by software to specify falling edge/low level triggered external interrupts.

2.9 SERIAL PORT CONTROL (SCON) REGISTER :

2.9.1 INTERRUPTS :

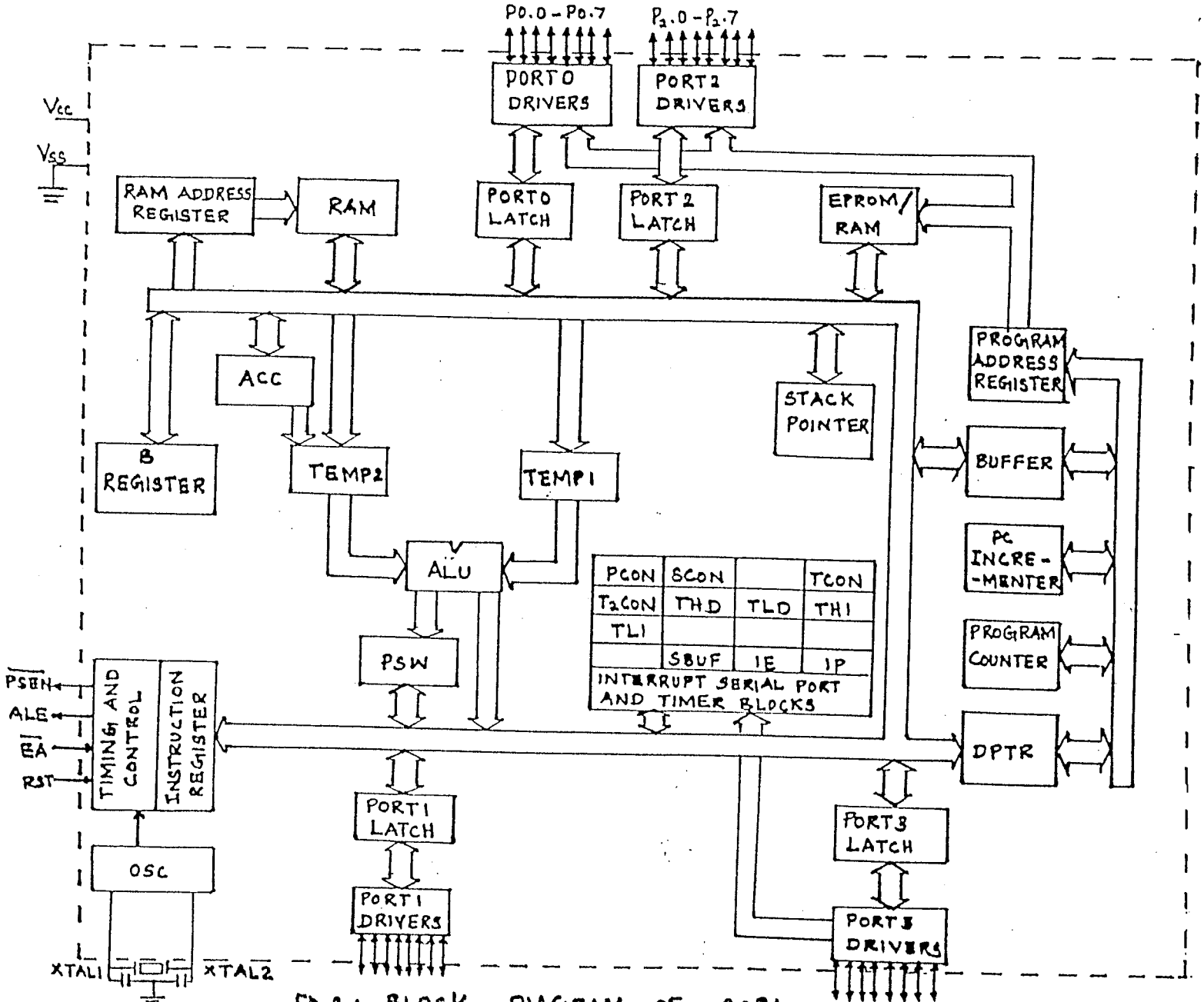
The 8031 provides 5 interrupt sources. The external interrupts INTO and INT1 can each be either level activated or transition activated depending on bits IT0 and IT1 in register TCON. The flags that actually generate these interrupt bits are bit IE0 and IE1 in TCON. When the external interrupt is generated the flag that is generated is cleared by the hardware. The service routine is vectored to only if the interrupt is transition activated.

2.9.2 PRIORITY OF INTERRUPT :

SOURCE	PRIORITY LEVEL	VECTOR ADDRESS
IE0	highest	0003H
TE0		000BH
IE1		0013H
TF1		001BH
R1+T1	lowest	0023H

2.9.3. RESET

The reset input is the RST pin, which is the input to a Schmitt trigger. A reset is accomplished by holding the RST pin high for atleast two machine cycles, while the oscillator is running. The CPU responds by generating an external reset.


P0.0 – P0.7
P2.0 – P2.7
PORT0 DRIVERS
PORT2 DRIVERS
Vcc
Vss
RAM ADDRESS REGISTER
RAM
PORT0 LATCH
PORT2 LATCH
EPROM/ RAM
PROGRAM ADDRESS REGISTER
ACC
STACK POINTER
B REGISTER
TEMP2
TEMP1
BUFFER
ALU
PC INCRE- -MENTER
PCON
SCON
TCON
T2CON
TH0
TL0
TH1
TL1
SBUF
IE
IP
INTERRUPT SERIAL PORT AND TIMER BLOCKS
PSW
PROGRAM COUNTER
PSEN
ALE
EA
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DPTR
PORT1 LATCH
PORT3 LATCH
OSC
PORT1 DRIVERS
PORT3 DRIVERS
XTAL1
XTAL2

CHAPTER III

SYSTEM DESCRIPTION

DIRECT BUS CONNECTED CMOS REAL TIME CLOCK / CALENDAR

3.1.1. GENERAL DESCRIPTION

The MSM6242B is a silicon gate CMOS Real time clock / Calendar for use in direct bus connection Microprocessor/Microcomputer applications. An on-chip 32.768 KHZ crystal oscillator time base is divided to provide addressable 4-bit 1/0 data for SECONDS, MINUTES, HOURS, DAY OF WEEK, DATE MONTH, and YEAR. Data access is controlled by 4-bit address, chip selects (CSO,CSI) WRITE,READ,and ALE.control registers D,E,and F provied for 30 SECOND error adjustment, INTERRUPT REQUEST (IRQ FLAG) and Busy status bits, clock STOP,HOLD, and RESET FLAG bits, 4 selectable INTERRUPTS rates are available at the STD. P (STANDARD PULSE) output utilizing control Register inputs T0, T1 and the ITRPT/STND (INTERRUPT/STANDARD). Masking of the interrupt output (STD.P) can be accomplished via the MASK bit. The MSM6242 B can operate in a 12/24 hour format and leap year timing is automatic.

The MSM 6242B normally operates from a 5v + or - 10% supply at - 30 to 85'c Battery backup operation down to 2.0v allows continuation of time keeping when main power is off. The MSM 6242B is offered in a 18-pin plastic DIP a 24 pin FLAT package, and a 18-pin PLCC package.

3.1.2. FEATURES

DIRECT MICROPROCESSOR / MICROCONTROLLER BUS CONNECTION

The main features can be listed below as

* 4 bit data bus
* 4 bit address bus
* READ, WRITE, ALE and CHIP SELECT INPUTS
* Status registers - IRQ and BUSY
* Selectable interrupt outputs - 1/64 second, 1 second 1 minute, 1 hour
* Interrupt masking
* 32.768 KHZ crystal controlled operation
* 12/24 hour format
* Auto leap year
* + or - 30 second error correction

* Single 5 v supply
* Battery backup down to V_{DD} = 2.0 v
* Low power dissipation :
    20 microwatt max at V_{DD} = 2V
    150 microwatt max at V_{DD} = 5
* 18 pin plastic DIP, 24-pin FLAT and 18-pin PLCC package.

3.1.3 PIN DESCRIPTION

DATA BUS (D_0 - D_3)

Data input / output pins to be directly connected to a microcontroller bus for reading and writing of the clock/calendar's registers. DO=LSB and D3 = MSB.

ADDRESS BUS (A_0 - A_3)

Address input pin for use by a microcomputer to select internal clock/calendar's registers and control registers for read/write operations. Address input pin A0 -A3 are used in combination with ALE for addressing registers.

ALE (ADDRESS LATCH ENABLE)

This pin enables writing of address data when ALE = 1 and CSO = 0 ; address data is latched when ALE = 0 Microcontroller / Microprocessors having an ALE output should connect to this pin ; other it should be connected at V_{DD}.

$\overline{WR}$ (WRITE)

Writing of data is performed by this pin.

When CS_1 =1 and CS_0 = 0 D_0 D_3

data is written into the register at the lowering edge of WR.

$\overline{RD}$ (READ)

Reading of register data is accomplished using this pin. When CS_1=1, CS_0=0 and RD =0, the data of the register is output to D_0 D_3. If both RD and WR are set at 0 simultaneously, RD is to be inhibited.

CHIP SELECT (CS_0 , CS_1)

These pins enable/disable ALE, RD, and WR operation. CS_0 and ALE work in combination with one another, while CS_1 work independent of ALE CS_1 must be connected to power failure detection.

STD.P

Output pin of N - CH OPEN DRAIN type. The output data is controlled by the D_1 data content of E register. This pin has a priority to CS_0 and CS_1.

XT, $\overline{XT}$

32.768 KHZ crystal is to be connected to these pins. When an external clock of 32.768 KHZ is to be used for MSM 6242's oscillation source, either CMOS output or pull-up TTL output is to be input from XT, while ST should be left open.

POWER SUPPLY (V_{DD})

+2 to +6v power is to be applied to this pin.

GROUND (GND)

This is used for grounding the ckt

3.2 MEMORY

3.2.1 EPROM 27256

This is an erasable programmable read only memory, which can be only read but not written into. It is of nonvolatile type.

This EPROM is used to store valuable data and monitor programs which should not be lost during power failures and ON/OFF of the system.

3.2.2 RAM 6264

6264 is a static random access memory with a standard 28 pin package configuration . It has 12 address lines, 8 input/output lines. 2 chip select lines of which CS1 is active low while CS2 is active high. No clock or timing strobe is required with the access time equal to the cycle time. The pin requires a single +5v supply and is directly TTL compatible.

3.3 74LS145

FEATURES

* 80 MA output drive capability
* 15V output breakdown voltage

DESCRIPTION

The 145 is a 1-of-10 decoder with open collector outputs. This decoder accepts BCD inputs on the A_0 to A_3 address lines and generates 10 mutually exclusive active LOW outputs. When an input code greater than "9" is applied, all outputs are HIGH. This device can therefore be used as a 1-of-8 decoder with A_3 used as an active low enable.

The 145 features an output breakdown voltage of 15v. This device is ideal as a lamp or solenoid driver. Refer Appendix for pin diagram.

3.4 74L245

FEATURES

* Octal bidirectional bus interface
* 3 - state buffer outputs
* PNP inputs for reduced loading
* Hysteresis on all data inputs.

DESCRIPTION

The LS 245 is an octal transceiver featuring non-inverting 3-state bus compatible outputs in both send and receive directions. The outputs are all capable of sinking 24mA and sourcing up to 15mA, producing very good capacitive drive characteristics. The device features a chip enable (CE) input for easy cascading and a send/Receive (S/R) input for direction control. All data inputs have hysteresis built into minimize AC noise effects.Refer Appendix for pin diagram.

3.5 74LS138

FEATURES

* Demultiplexing capability
* Multiple input enable for easy expansion
* Ideal for memory chip select decoding
* Direct replacement for Intel 3205

DESCRIPTION

The 138 decoder accepts three binary weighted inputs (A_0,A_1,A_2) and when enabled, provides eight mutually exclusive, active LOW outputs (0 - 7). The device features three enable inputs ; two active LOW (EP_1, E_2) and one active HIGH (E_3). Every output will be HIGH unless E_1 and E_2 are LOW and E_3 is HIGH. This multiple enable function allows easy parallel expansion of the device to a 1-of-32 (5 lines to 32 lines) decoder with just four 138S and one inverter.

The device can be used as an eight 0/P demultiplexer by using one of the active LOW Enable inputs as the Data input and remaining Enable inputs as strobes.Enable inputs not used must be permenantly tied to their appropriate active HIGH or active LOW state.

3.5 74LS373

FEATURES

* 8 bit transport latch - 373
* 8 bit positive edge triggered register - 374
* 3 state output buffers
* common 3 state output Enable
* Independent register and 3 state buffer operation.

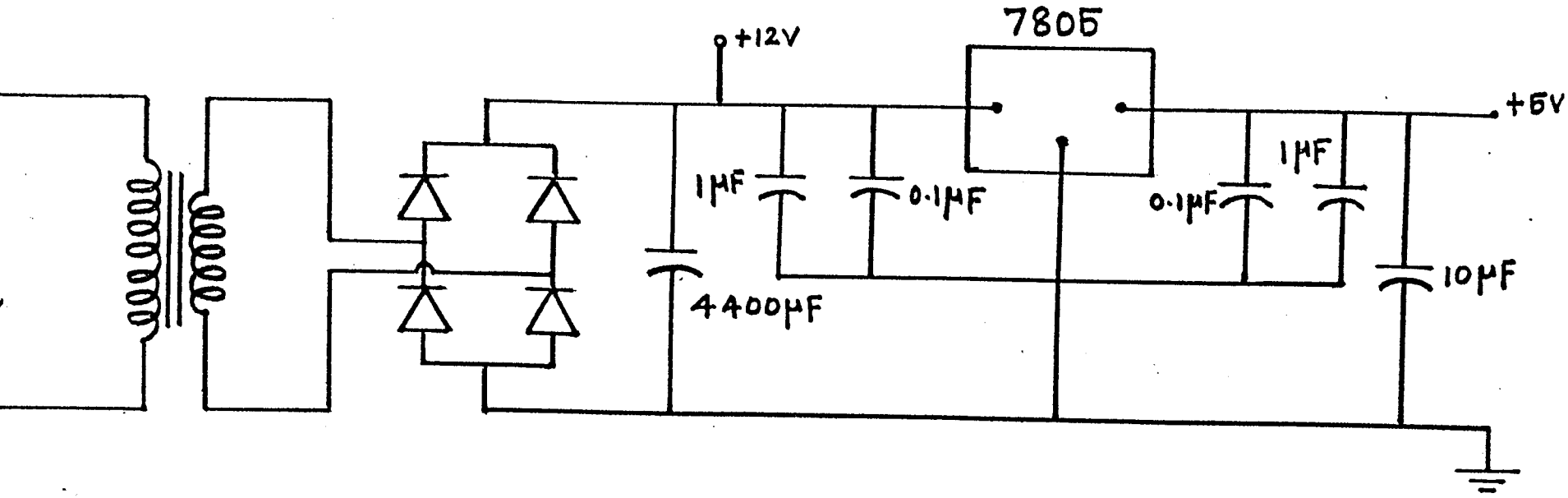
DESCRIPTION

The 373 is an octal transport latch coupled to eight 3 state output buffers. The two sections of the device are controlled independently by Latch Enable (LE) and output Enable (OE) control gates.

3.7 Block Diagram of the Power Supply

The power supply to the microcontroller system for programmable timer is as shown in Fig 3.1. The AC current in the mains is 230v/5A. This is then converted to 12v using a

fullwave bridge rectifier. This is then passed through capacitors to cut down the ripples. The voltage with very less ripple factor is then passed through a voltage regulator IC7805 to reduce the O/P to regulated value of 5v which is given as Vcc to all the components.



3.7 BLOCK DIAGRAM OF THE POWER SUPPLY

CHAPTER IV

BLOCK DIAGRAM AND OPERATION OF THE PROGRAMMABLE TIMER

4.1 Block Diagram

The block diagram of the timer is as shown in fig 4.1 The block diagram of the timer can be mainly divided into two section namely.

1. RAM, EPROM, RTC interconnected by address, data and control bus.

2. Display section consisting of decoder, keyboard and seven segment LED display unit.

4.2 RTC & MEMORY UNIT

The RAM, EPROM, RTC are interconnected by address, date, control bus. The functions of these are as follows.

i. Address bus : it is a unidirectional bus consisting of sixteen lines namely A_0 to A_{15}. The microcontroller uses the address bus to identify a peripheral or a memory location

ii. Data bus : it is a bidirectional bus. Microcontroller uses the data bus to transfer data.

iii. Control bus : The control bus is comprised of various single lines that carry synchronization signals. The microcontroller uses such lines to provide timing signals.

4.3 KEYBOARD AND DISPLAY UNIT

From the port of 8031, 3 lines are connected to the 3 x 8 decoder the o/p of which is connected to the keyboard and display unit. A retrace line is taken from the keyboard and connected back to port of 8031 as input line.

The scan code generated by microcontroller keeps hanging for every millisecond and if a particular key is ressed it is sensed through the retrace line and the partic- lar scancode generated at the time is the valid one. Now isplay data is changed accordingly and this is displayed in he display unit for (eg) if increment key is pressed then he display value is incremented and this value is displayed n the display unit.

.4 Operation

The operation of the programmable timer is mainly con- rolled through four keys namely.

1. Sub menu key
2. increment key
3. Toggle/ Accept key
4. Mode changing key

The various modes provided through the software is as ollows

1. Time setting mode
2. Alarm setting mode
3. Alarm deleting mode
4. Holiday setting mode
5. Holiday deleting mode
6. Reset mode
7. Day setting mode
8. Display mode

The functions of the keys are as follows.

4.5 Submenu key :

This is used for setting particular year, month, date, hours, minutes, day, etc., to a particular value after a particular mode is selected.

4.6 Increment key :

This key is used for incrementing the year, month, date, to the required value. For example if we want to set

the date to 15th, we must press the increment key 15 times so that it changes from 00 to 15 in steps of one. i.e increment key increments the value of contents in steps of one.

4.7 Toggle / Accept key :

In the normal display mode, the calendar is displayed in the seven segment display. This key is used for toggling between the time & calendar display mode. Thus if the display is in calender mode if we use press the toggle key the mode changed time display mode of vice versa. In all other modes it is used as accept key. The function of key is that, if we want to set the different values of alarm time, the micro controller will accept the data, only when the accept key is pressed. Otherwise the value is not accepted. Thus this prevents from accidental changing of the time.

4.8 Mode changing key :

This key is used for changing the mode from one to another. The function of different modes being as follows.

4.8.1 Time Setting Mode :

In this mode the display of the seven 7-segment LED's is as shown in fig 4.2 A.

In this mode, first in the display 'year' appears in the 3 seven segment LED and 'TS' in the next 7 segment LED's. We have to set the year to say '95' by incrementing the increment key '95' from 00 to 95.

After the submenu key pressed to next value. i.e month i.e./ it display 'mon' in the first 3 seven segment LED display then it is set to the desired value using increment and finally the accept key is pressed to accept the foll cycle is obtained the data as shown in Fig 4.2 B when we press the submenu key. Thus these values are set using the increment & accept key.

4.8.2 Alarm setting mode :

The alarm setting mode is used to set the time and duration for which the bell has to ring. In this mode the display of the seven - seven segment LED's is as shown in Fig 4.3 A first the hour is displayed. The no of setting is indicated as '00' at starting . Since we haven't input any time & duration before. Initially the data is at 0'. Now we set the pressing the increment key.

Next Month & duration is set in the similar way. Thus following cycle is followed as shown in Fig.4.3 B Now the no. of setting indicate the value of '01'. Thus this procedure is followed for all the time input & the time setting values indicates the value of no. of inputs given. Thus in this mode, time at which the bell should ring & duration is belling is given.

4.8.3 Alarm deleting mode :

This mode is used for deleting an alarm set value if we don't need that particular value. The display is as shown in Fig.4.4 In the first seven - segment (LED), the letter 'D' appears to indicate it as delete mode here each time & duration can be seen by the incrementing key the in crement key & we can delete the unwanted value by pressing the delete key (accept key).

4.8.4 Holiday Setting Mode :

This mode is used for turning the Alarm off on the holidays. Thus the holiday date & month is given i/p & the microcontroller turns off alarm for these days. The first three display the 'mon' for month & no of setting is zero & we have to indicate the month, say march by 03 in the data incrementing the increment. Then submenu key is pressed to follow the cycle as shown in Fig.4.5.

4.8.5 Hoilday Deleting Mode

This mode is used for deleting the holiday i/p given.This mode is useful since the date on which holiday occurs changes for each year. The display will be as shown in Fig4.6.

4.8.6 Reset Mode :

This mode is used for deleting all the setting given including time setting, alarm setting & holiday setting and bringing it to the original condition.

This display in this mode will be as shown in the Fig.4.7 A

In this mode , the word 'PASS' appears on the display to indicate that pass word is needed resetting the timer. If we press the accept key, when the data is '99' then the timer is reset. The Display will be now as shown in the Fig.4.7B

4.8.7 DAY SETTING MODE

This mode allows the user have to turn the alarm setting ON or OFF for that particular day. For (eg) if we want to turn the alarm 'off' on the Sunday or leave days, this mode comes in handy. The display will be as shown in the Fig.4.8A

The first three seven segment LED's indicate the day. The last two seven segment LED's indicate whether the alarm setting is ON or OFF. The increment key is used for toggling between ON or OFF. The following cycle followed as shown in Fig4.8B is followed when we press the submenu key. Finally to accept the setting we press the accept key.

4.8.8 DISPLAY MODE

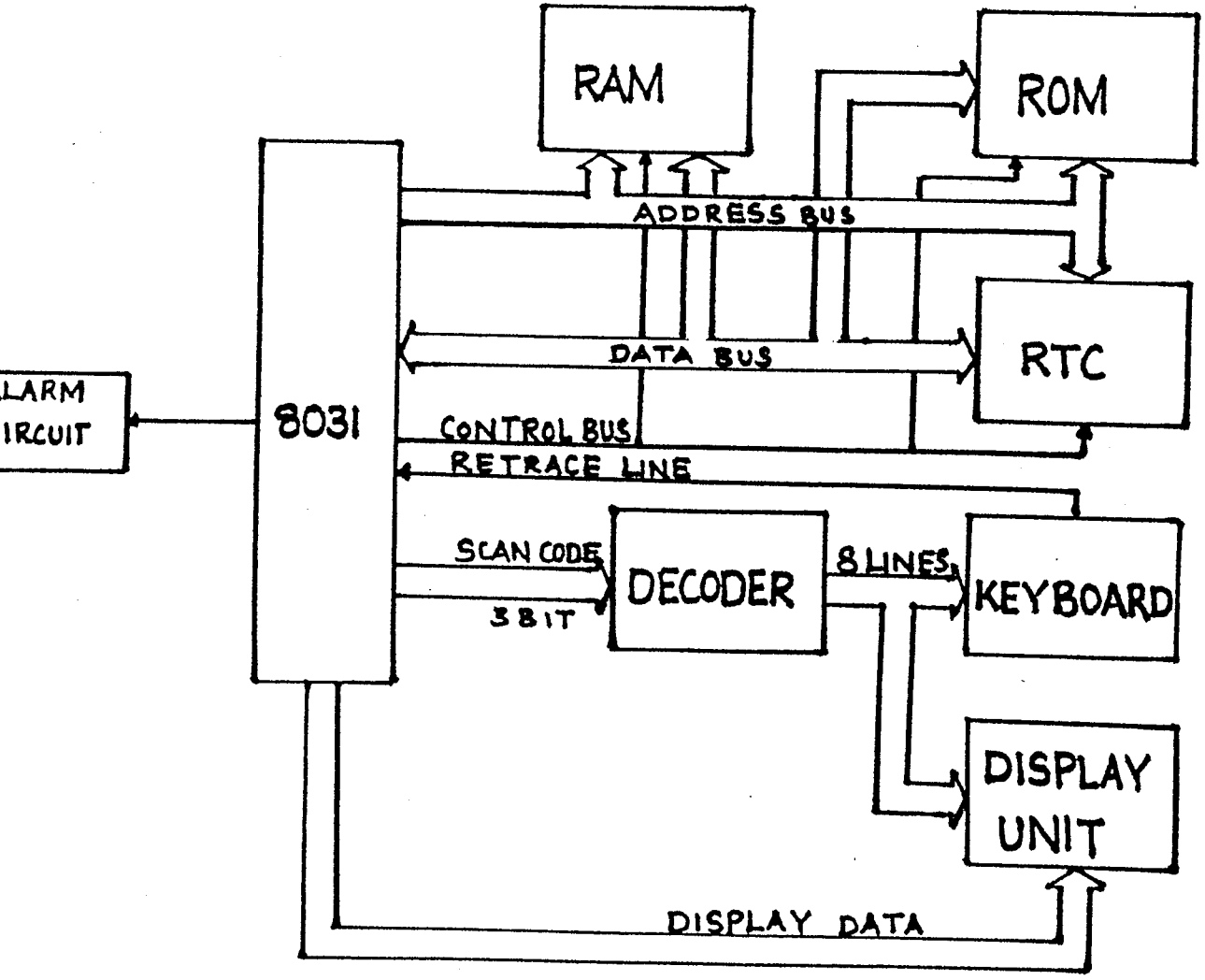
This mode essentially consists of two mode namely time display & calendar display mode

In time display mode ,the display of 7 - segment LED is as shown in Fig 4.9A. The first three segments display the day of week say sun (for Sunday) the next two indicate the hour & the next two seven seg. LED's the minutes. Thus this mode display the current time after the real time clock is synchronized using the time setting mode.

Calendar Display Mode

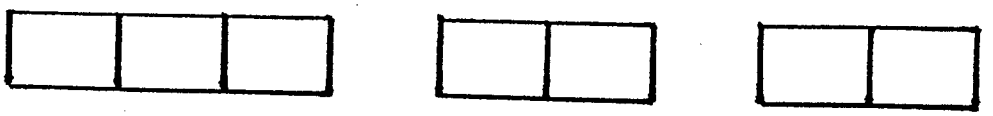
The calendar display mode is used for displaying the current Date/mon/year. After the RTC is synchronized using the time setting mode. The toggle/accept key is used for changing the mode from calender to time display or vice versa

In this mode display will be shown in Fig4.9B here the &3 indicates the current date. The fourth & fifth the month and the last two the year.

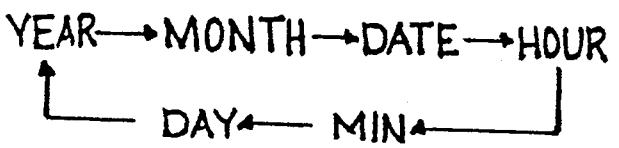


4.1 BLOCK DIAGRAM OF THE PROGRAMMABLE TIMER

TIME SETTING MODE :-



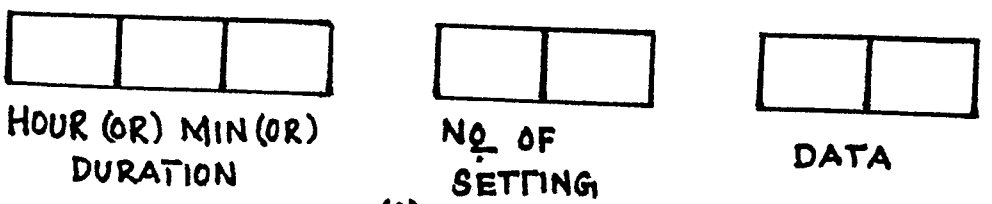
(a)



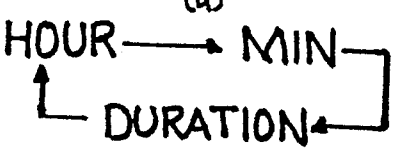
(b)

Fig 4.2

ALARM SETTING MODE :-



(a)



(b)

Fig 4.3

ALARM DELETING MODE :-

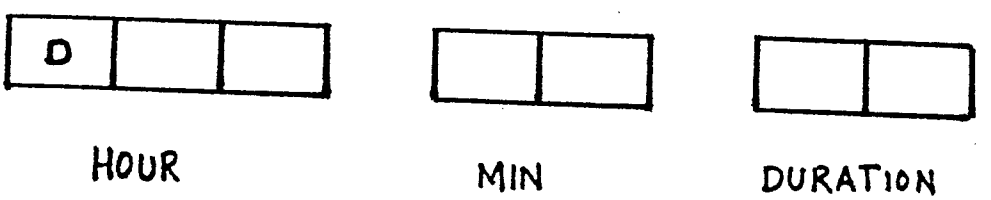


Fig. 4.4.

HOLIDAY SETTING MODE :-

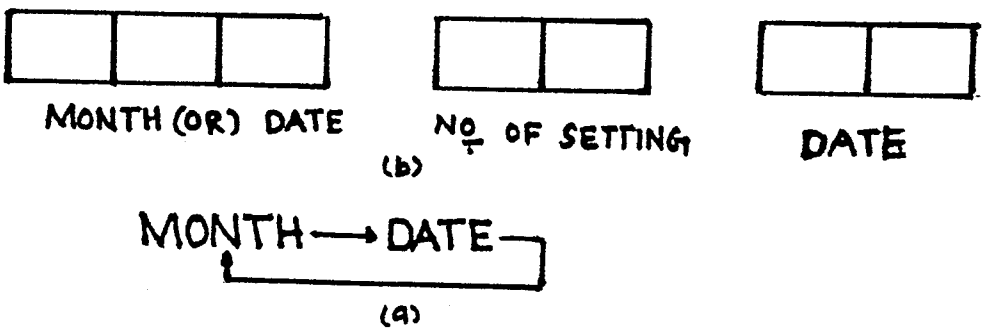


Fig. 4.5

HOLIDAY DELETING MODE :-

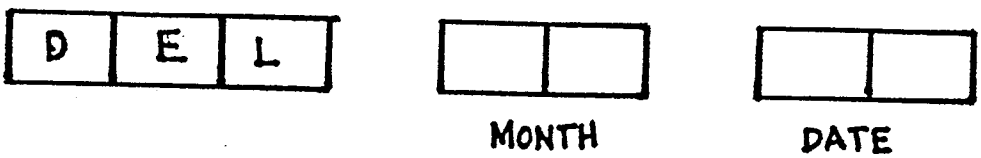


Fig 4.6

RESET MODE :-

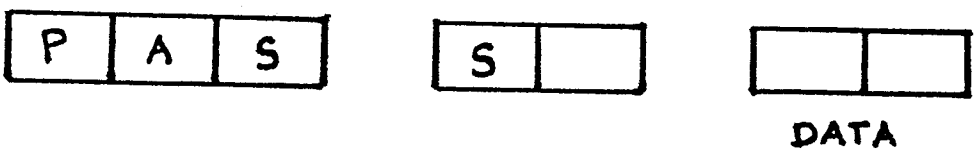
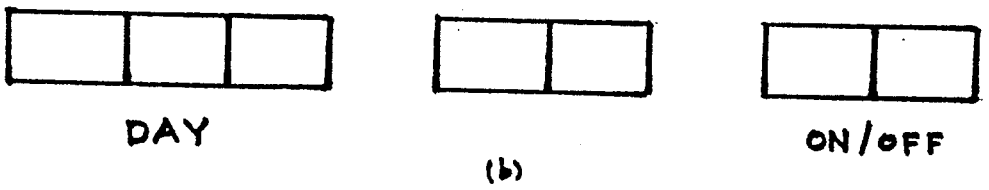


Fig 4.7

DAY SETTING MODE :-



(b)

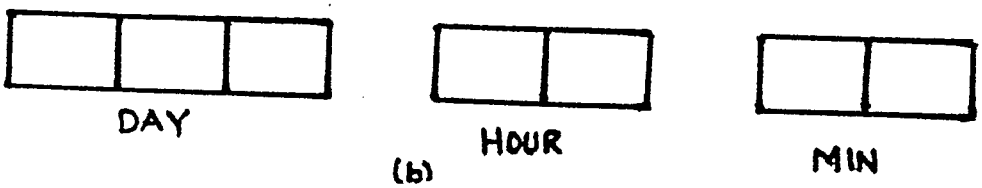
SUN → MON → TUE → WED
SAT ← FRI ← THUR

(a)

Fig. 4.8

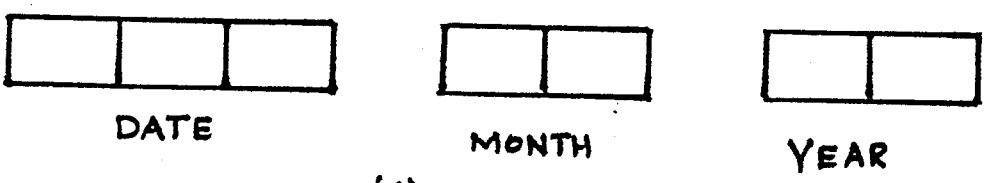
DISPLAY MODE :-

TIME DISPLAY MODE



(b)

CALENDAR DISPLAY MODE



(a)

Fig 4.9

Fabrication...

CHAPTER V

FABRICATION

5.1 INTERFACING MEMORY WITH 8031

The ports P_0 and P_2 gives the address of the memory. The port po constitutes address & data line. The address lines are latched with Arithmetic Latch enable signal (ALE). This signal is given to IC 74LS 373 as shown in Fig 5.1. The Data lines are pulled with the help of 10k resistance. The O/P of 74LS373 is connected with Data bus of 27256 and 6264 (RAM). The memory interfacing is done as shown in Fig 5.2. The address lines A_0 -A_{12} of 8031 is connected with RAM as such A_{13}, A_{14} is left free & A_{15} is used for selecting the RAM .

5.2 INTERFACING 6242 REAL TIME CLOCK (RTC)

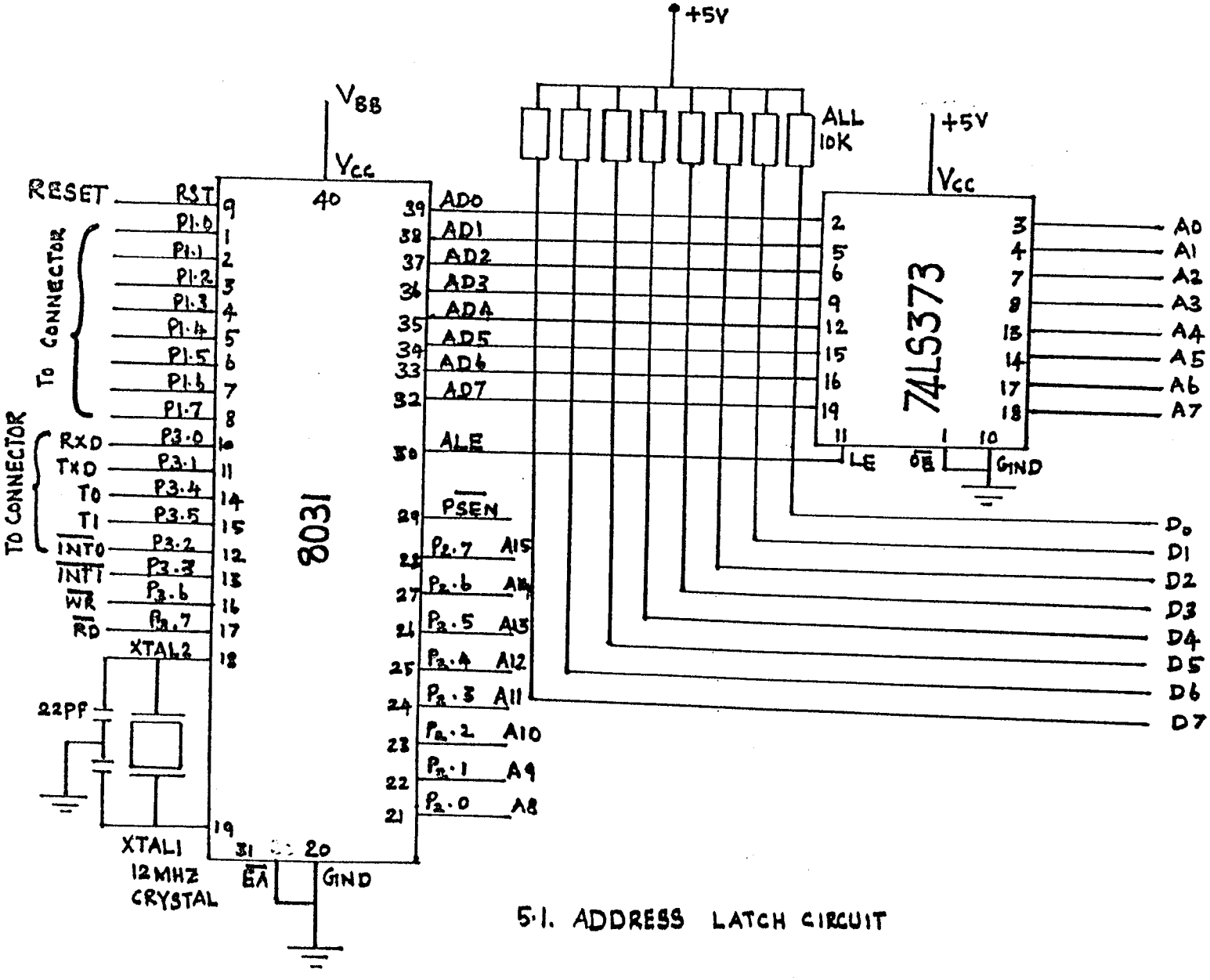
RTC has got 4 bit data bus and a four bit address bus. $A_0 - A_3$ & $D_0 - D_3$ of 8031 is connected with 6242 and ALE & RD is properly interfaced. The chip select signal is obtained from A_{14}.

5.3 INTERFACING DISPLAY WITH 8031

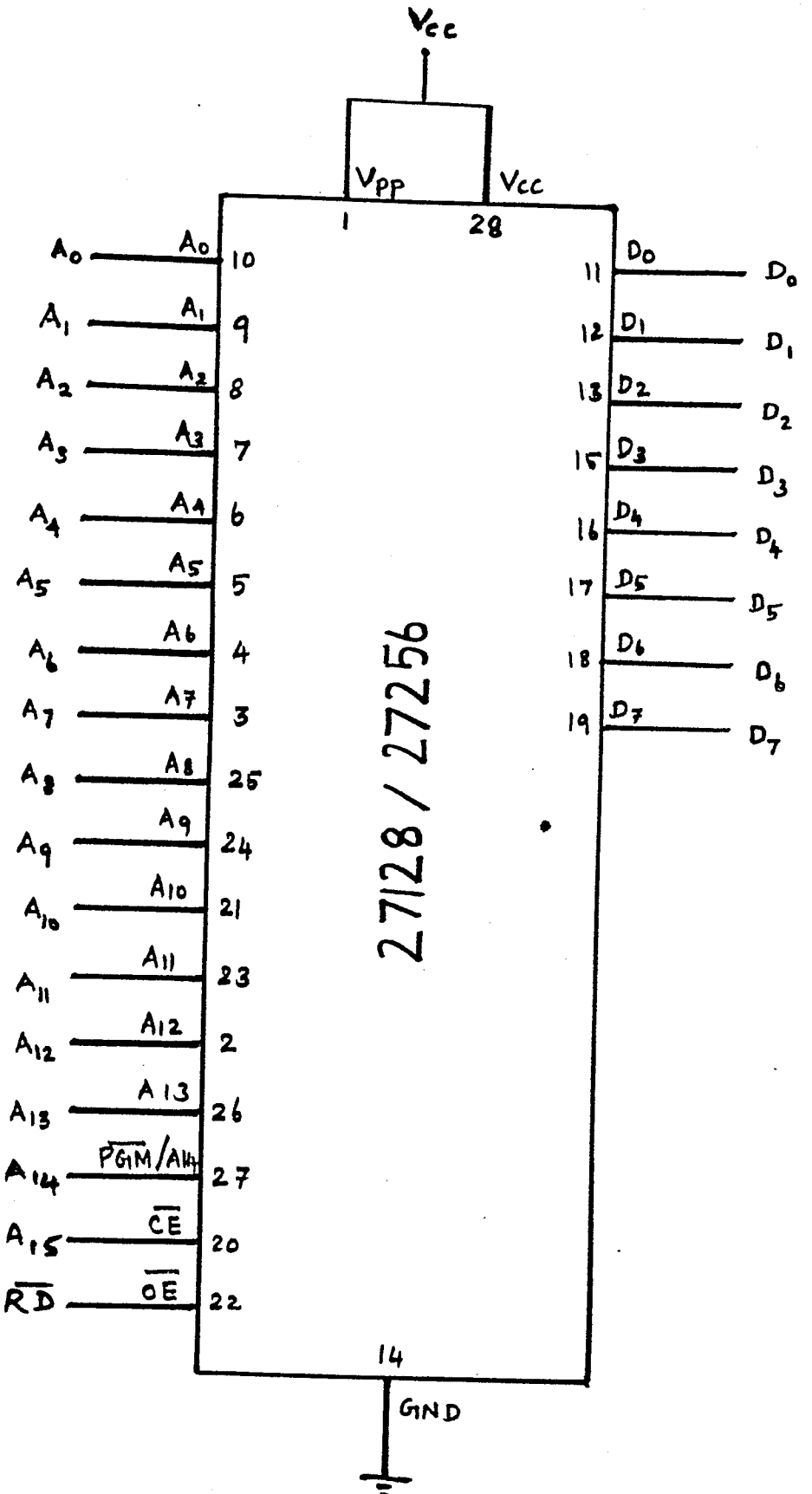
8 lines namely 1 to 8 of 8031 are taken and given as input to the octal transreciever (i.e. 74LS245). From one output (here pin 18) of 74LS 245 is connected to the segment 'a' of all the seven segment LED DISPLAY . Similar procedure is followed for interfacing other segments with the O/P of the 74LS245. Pin 9,10,11 of 8031 is given as input to the 74LSL145 so that any one of the seven segment LED's can be selected at any particular time

5.4 INTERFACING KEYBOARD WITH 8031

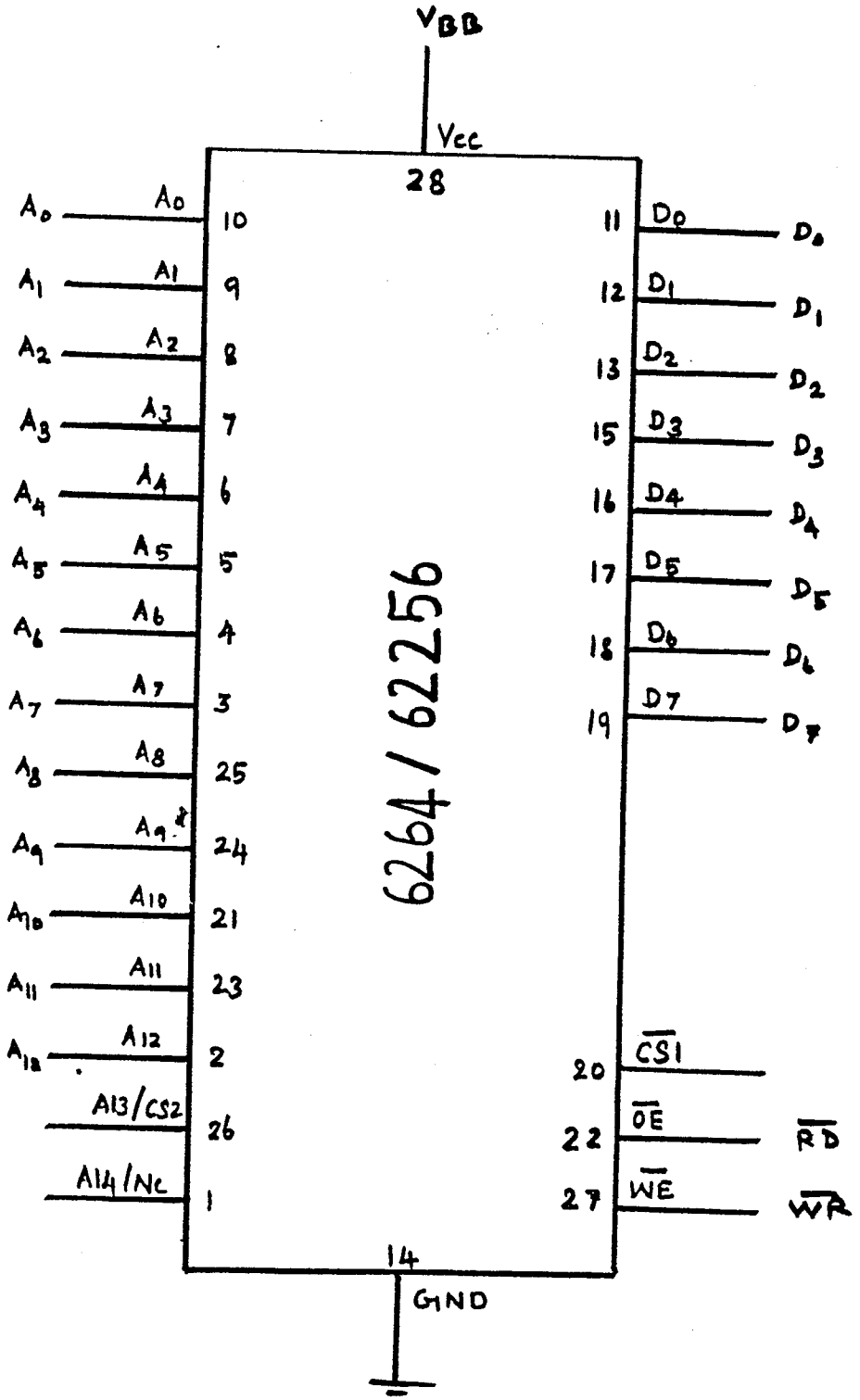
Lines 9,10,11 of 8031 are given as inputs to the decoder 74LS138 and the output of the decoder is directly connected with keyboard. From the keyboard a Retrace line is taken and given back as input to 8031.



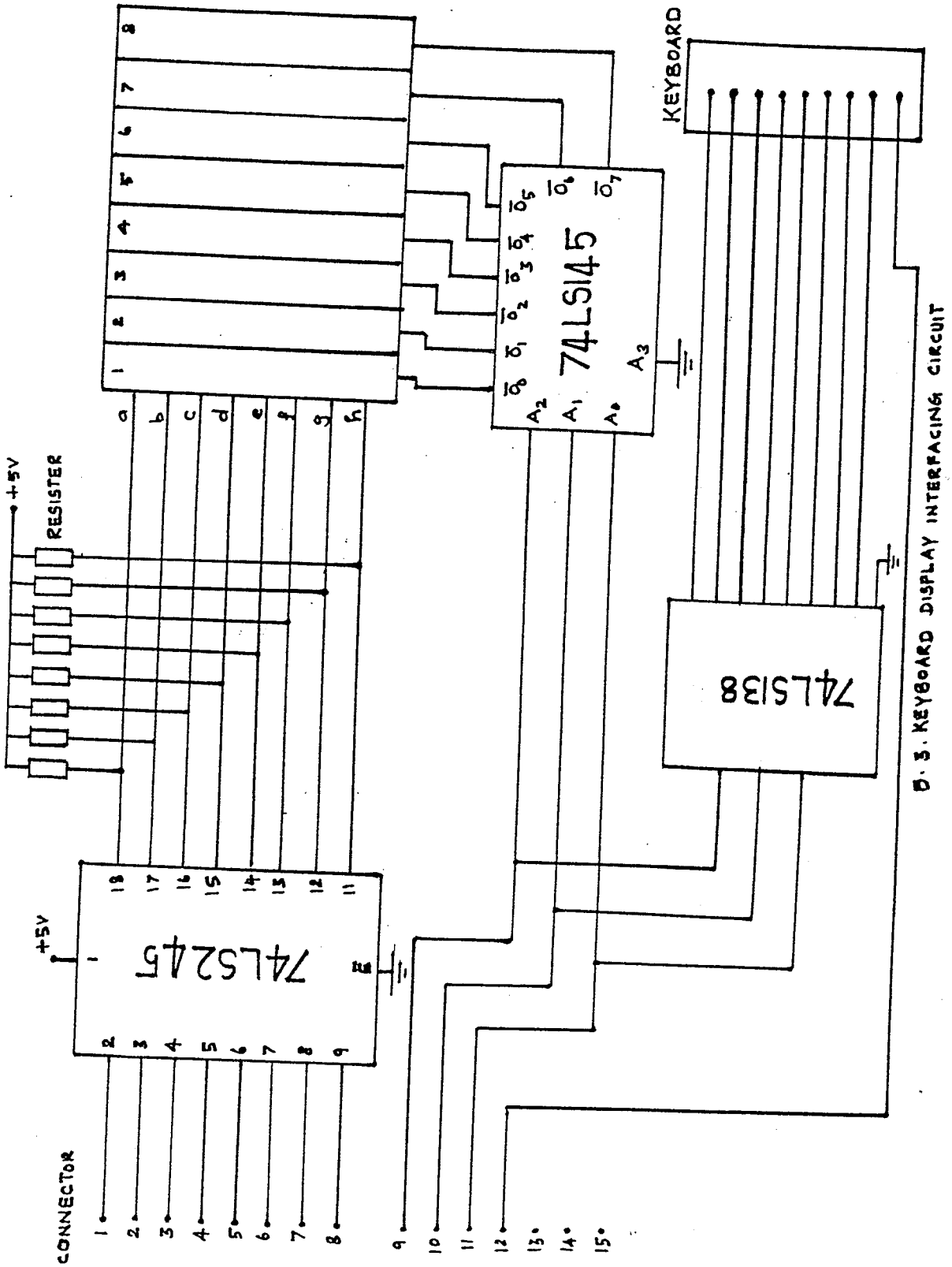
5·1. ADDRESS LATCH CIRCUIT



5.2. Memory Interfacing

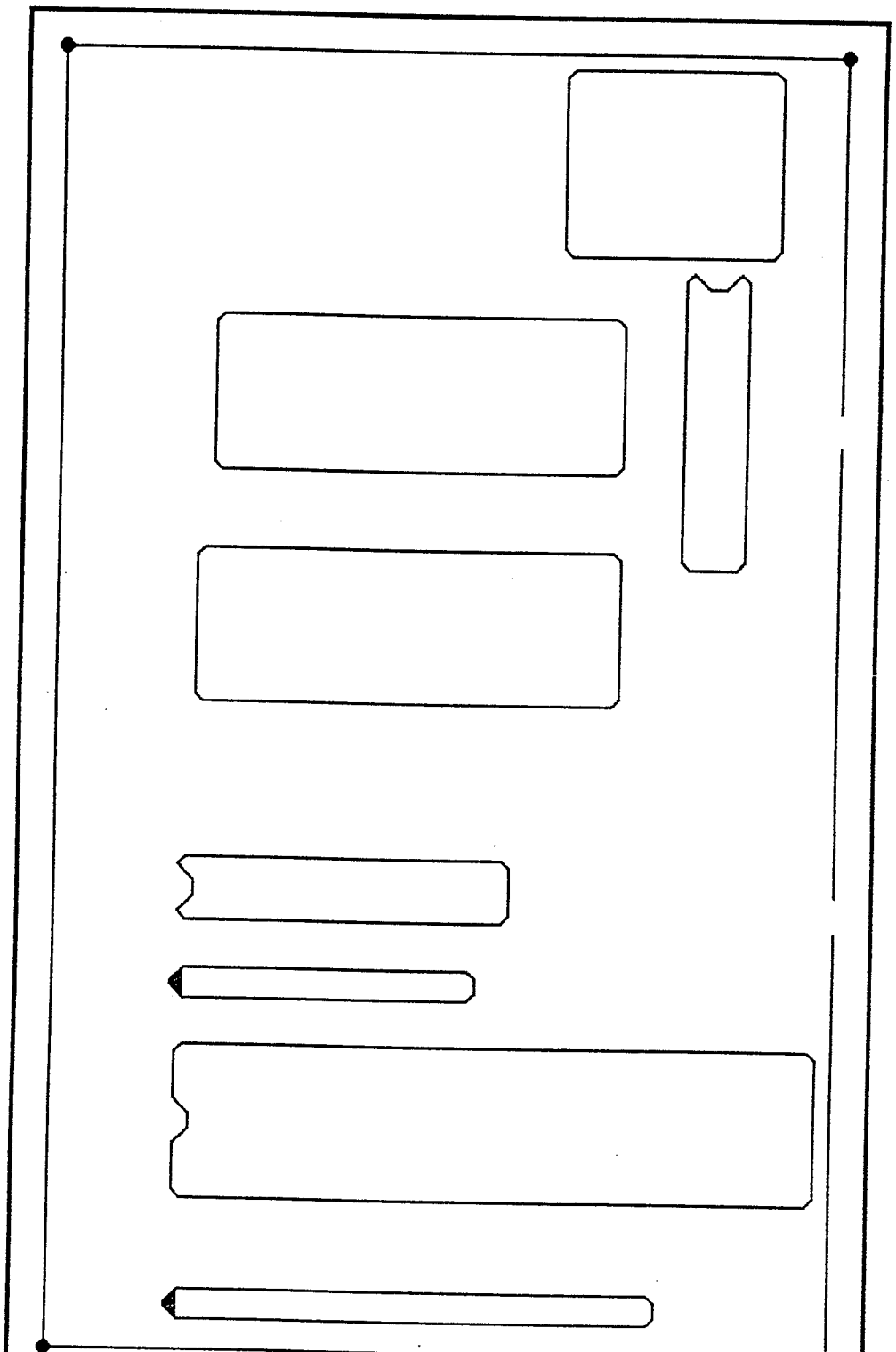


5.2 MEMORY INTERFACING



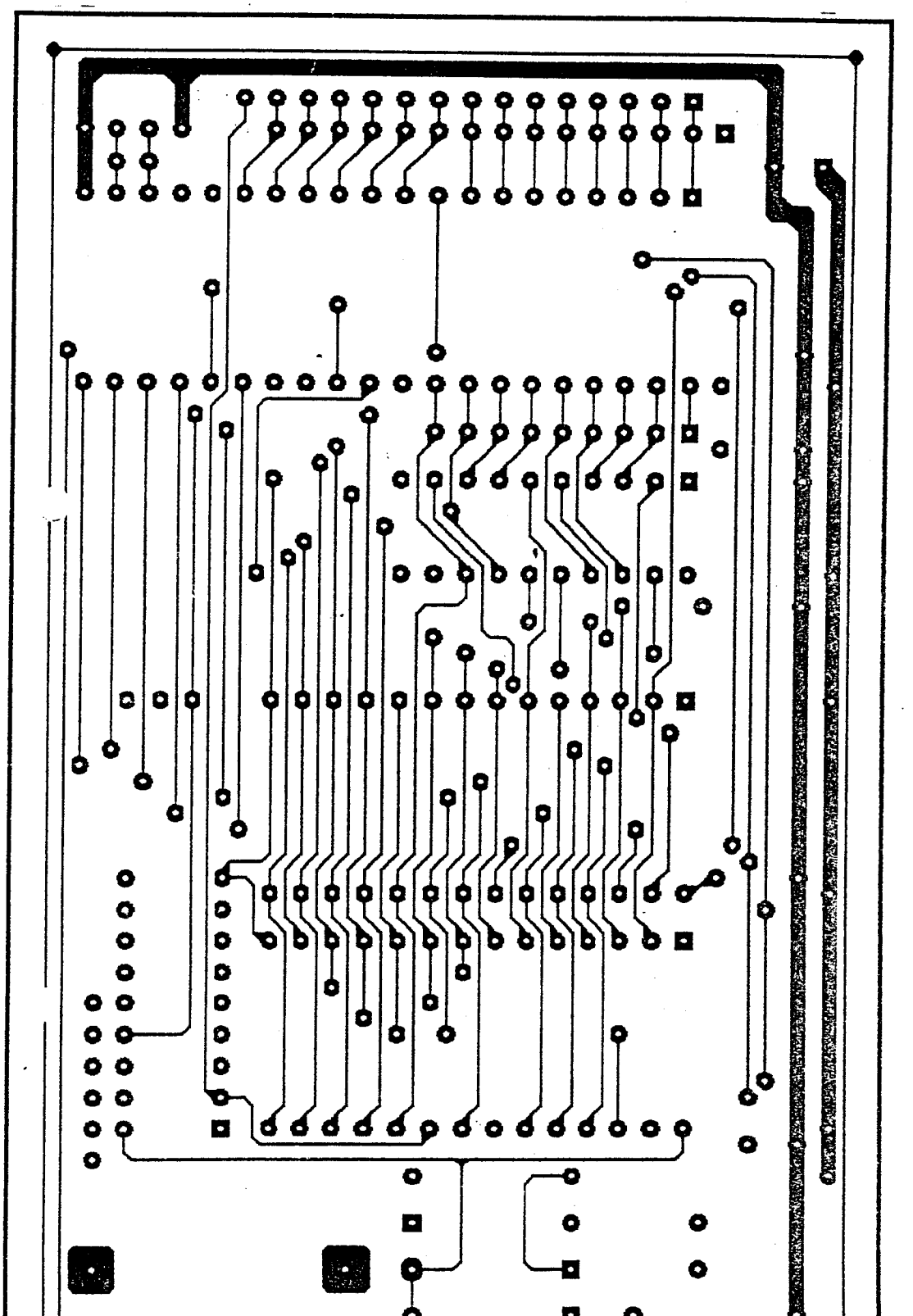
D. 3. KEYBOARD DISPLAY INTERFACING CIRCUIT

2X artwork 31 Aug 94 14:52:58
a:dee.pcb
v1.3 r4 holes: 279 silkscreen
approximate size: 2.55 by 4.25 inches

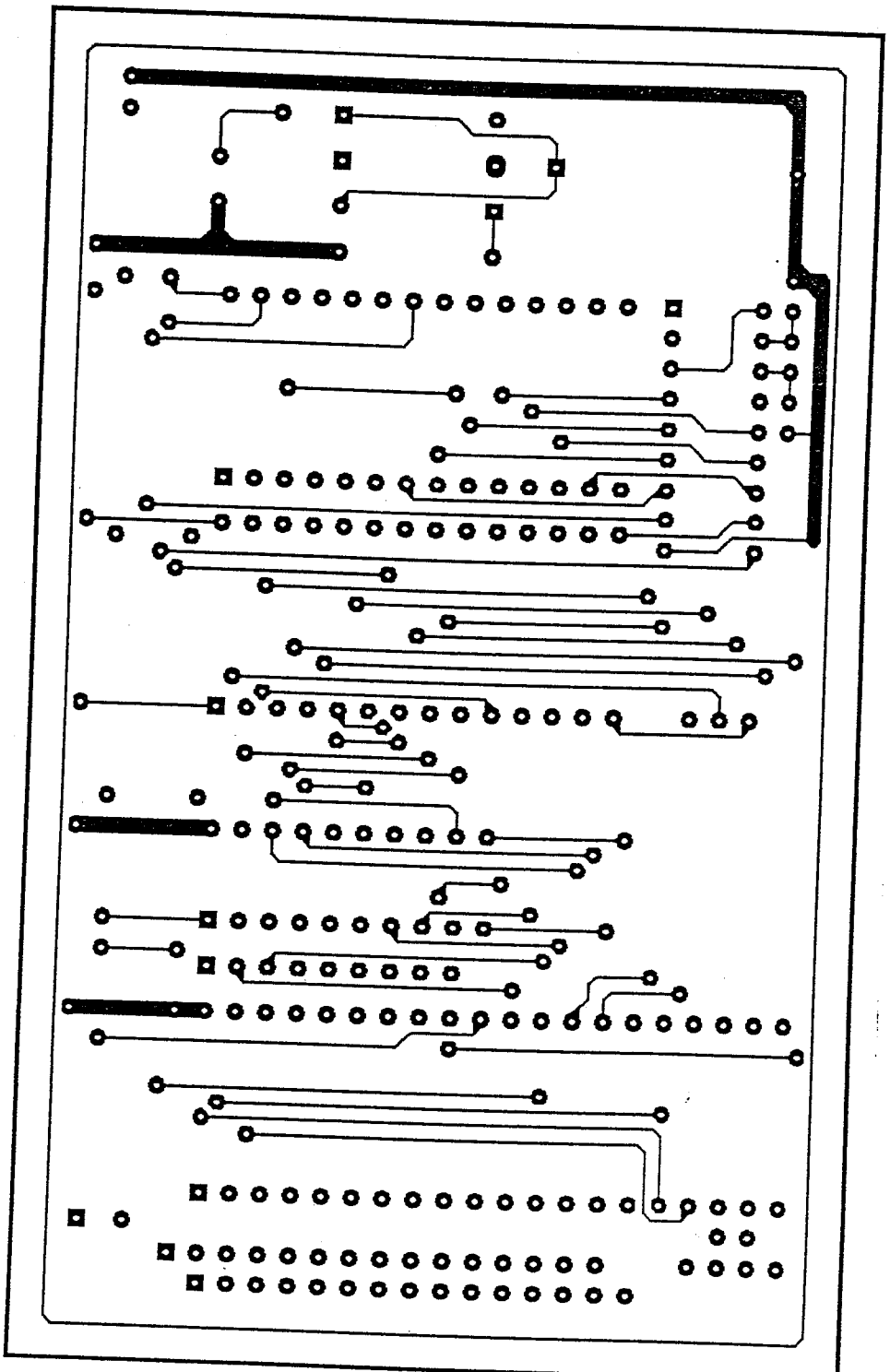


COMPONENT SIDE

2.55 BY 4.25 INCHES

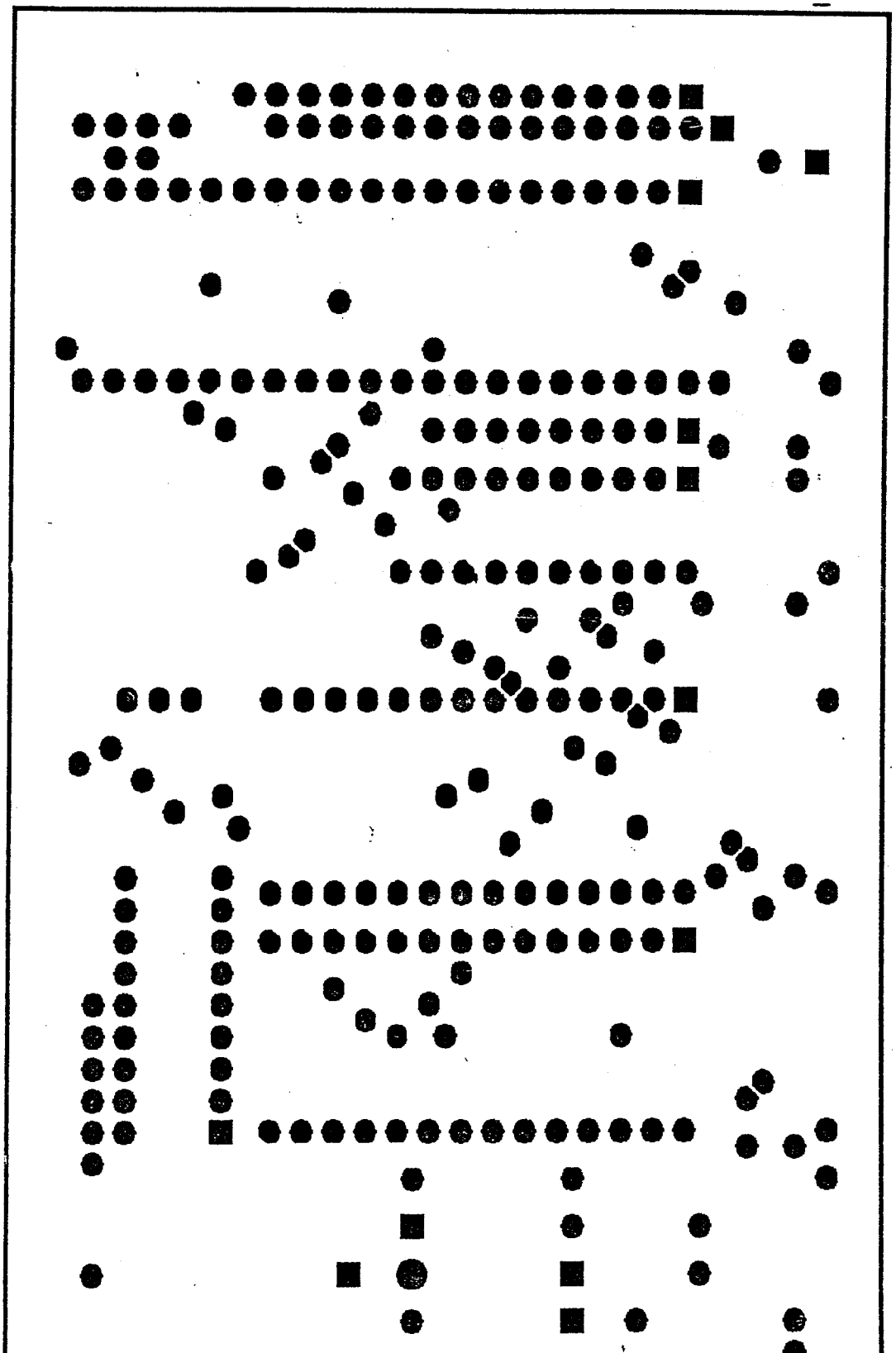


2X artwork 31 Aug 9 13:31:30
a:dee.pcb
v1.3 r4 holes: 279 solder side
approximate size: 2.55 by 4.25 inches

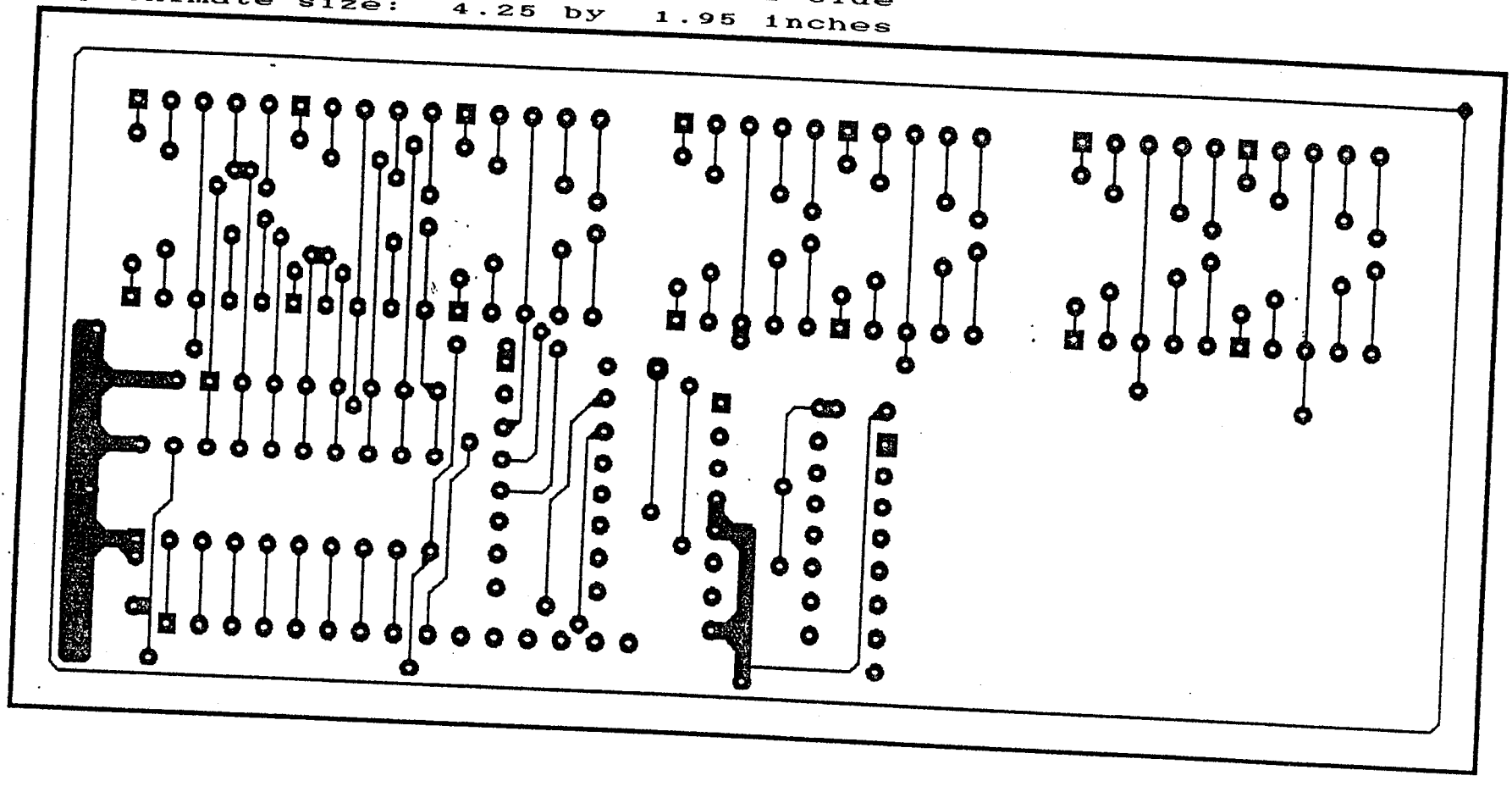


COMPONENT SIDE MASK

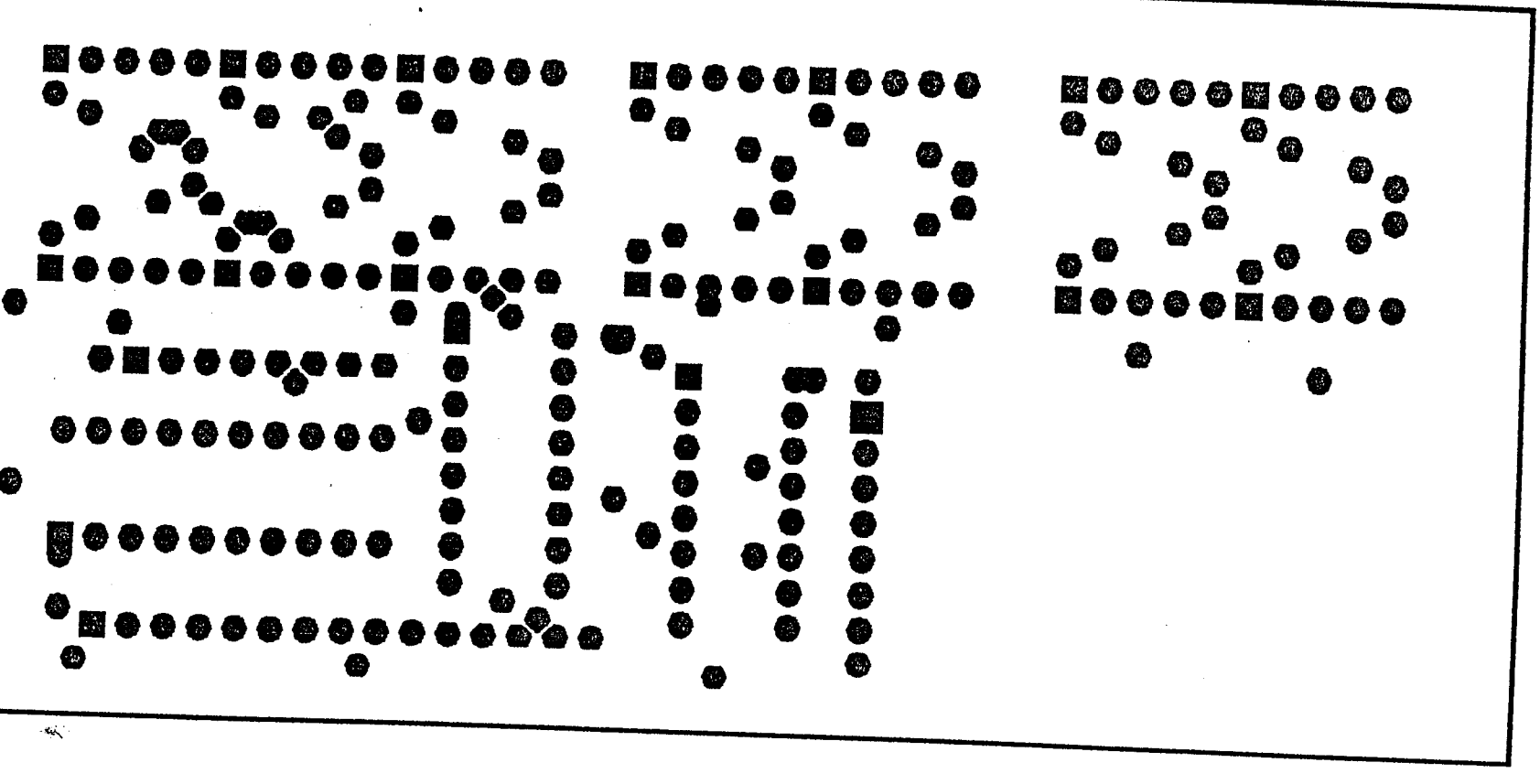
2.55 BY 4.25 INCHES



2X artwork
a:\jag 21 Oct 92 01:31:18
v1.3 r4 holes: 245 solder side
approximate size: 4.25 by 1.95 inches



artwork 21 Oct 92 01:54:46
Jag
.3 r4 holes: 245 component side mask
proximate size: 4.25 by 1.95 inches



CHAPTER VI

SOFTWARE

5.1 ALGORITHM

1. Set the time, thus synchronizing the RTC with current year, month, date, hour, minute, day.

2. get the multiple inputs for which the alarm is to ring

3. set the timer off on holidays by getting the date & month of the holiday.

4. set the timer ON or OFF for each particular day of the week.

5. delete the unnecessary alarm and holiday in puts.

6. make the timer to operate in the time display mode.

7. check whether today is a holiday, if so exit from the system else go to next step

8. check whether the particular day of the week is set if so go to step 9 else exit from the system.

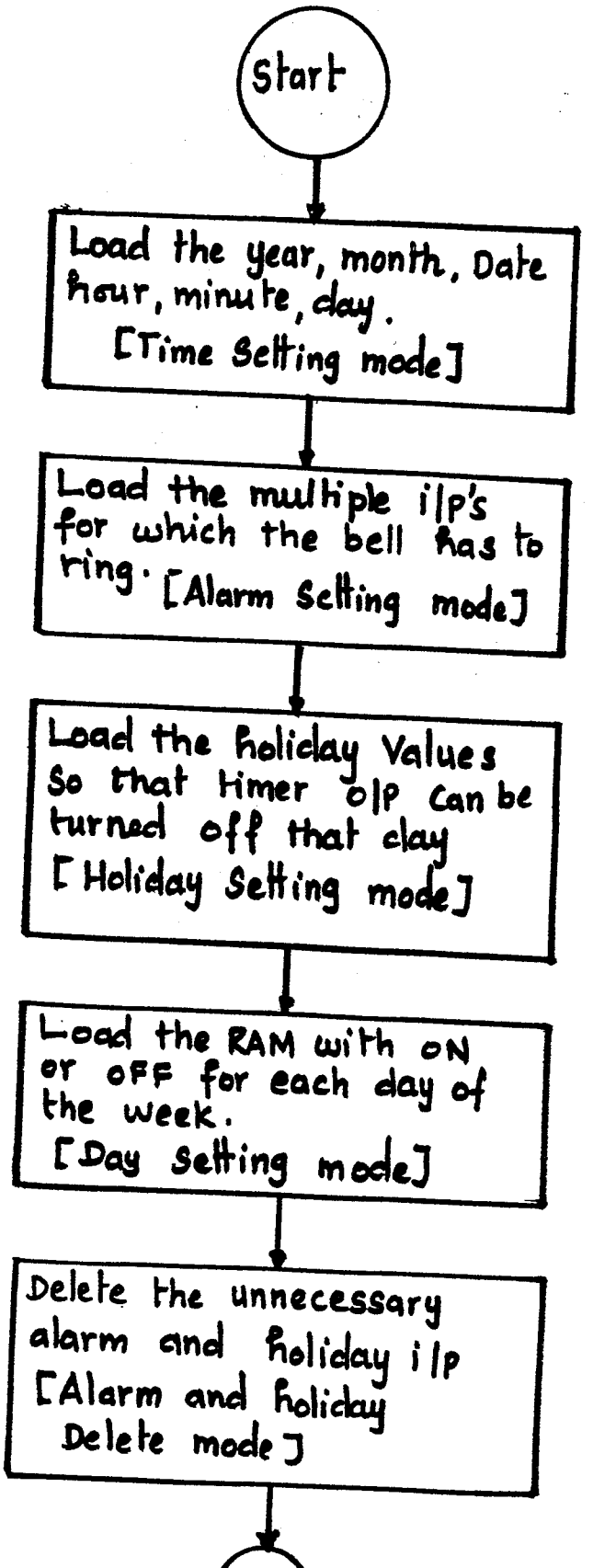
9. check whether if any of the alarm input is equal to the RTC value.

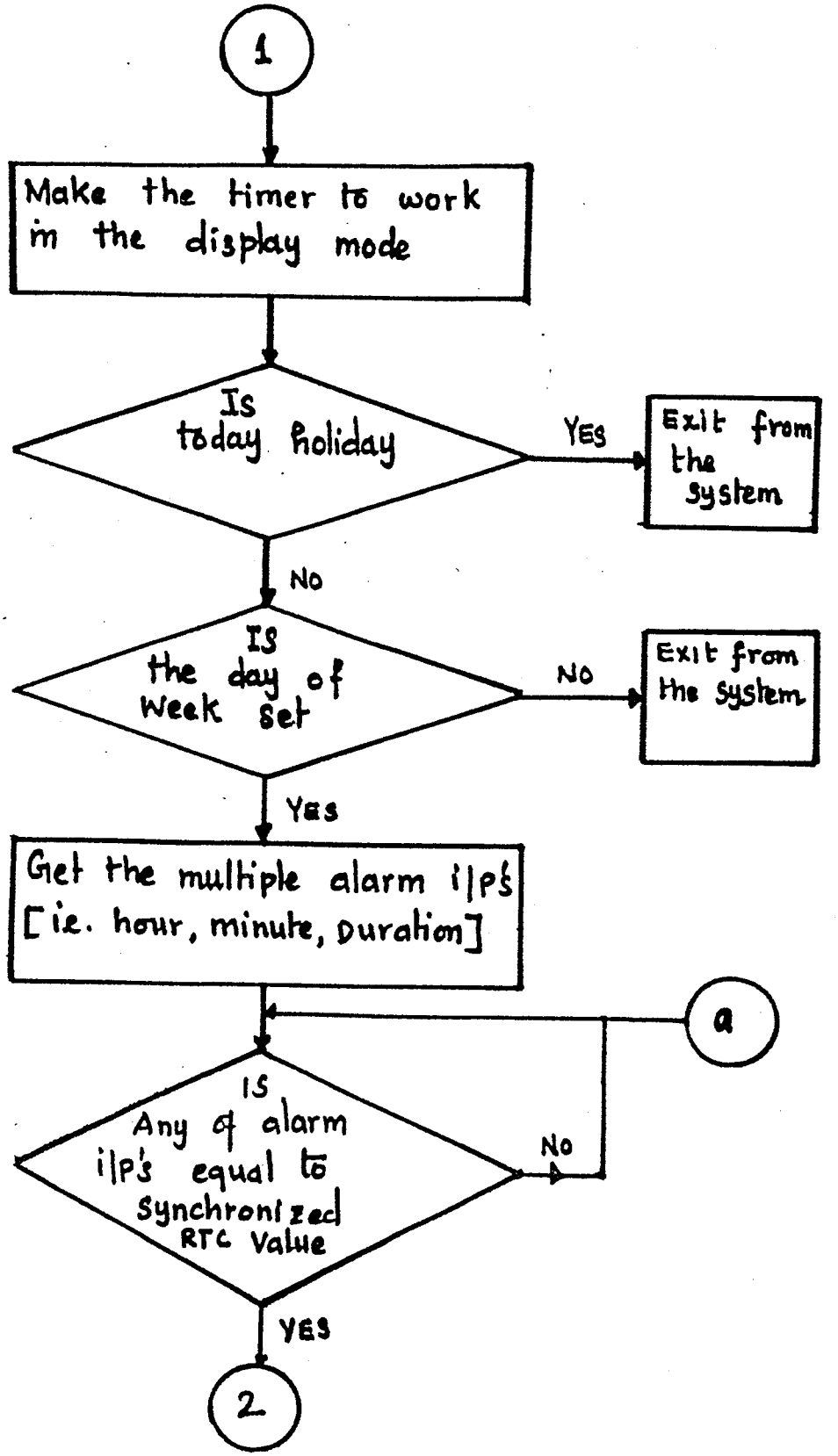
10. if it is not equal then to back to step 9

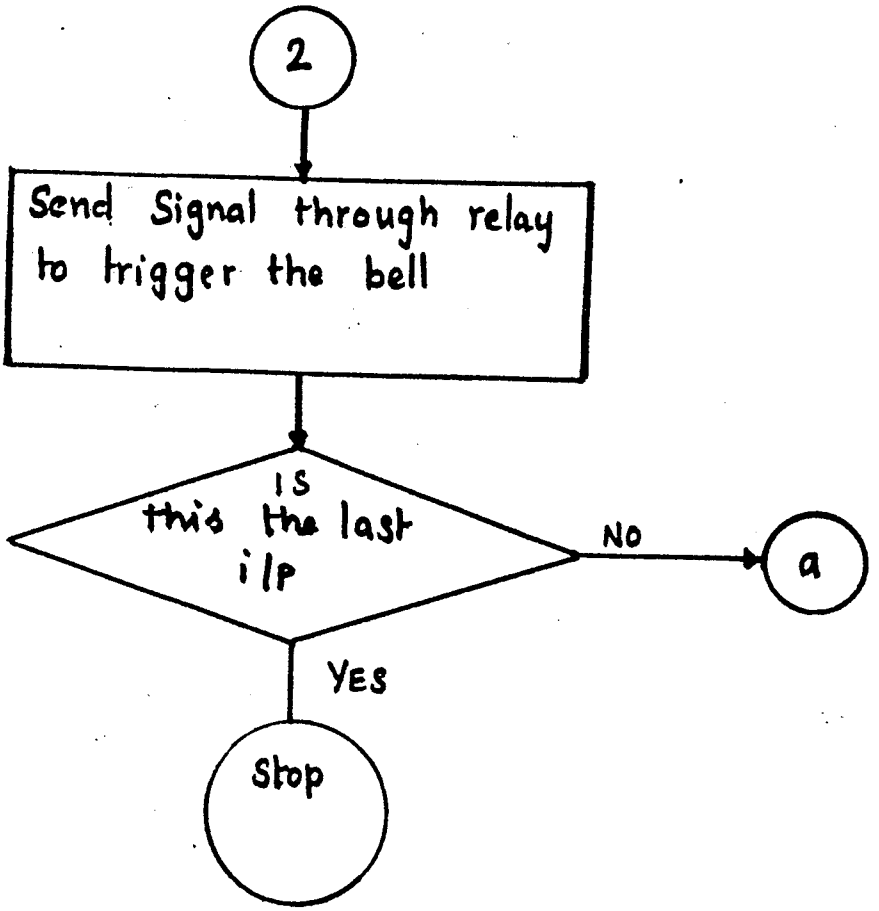
11. if it is equal then send signal through relay trigger the bell.

12. check whether if it is the last input, if so exit from the system else go to step 9.

FLOWCHART


Start
Load the year, month, Date hour, minute, day. [Time Setting mode]
Load the multiple i/p's for which the bell has to ring. [Alarm Setting mode]
Load the holiday Values so that timer o/p can be turned off that day [Holiday Setting mode]
Load the RAM with ON or OFF for each day of the week. [Day Setting mode]
Delete the unnecessary alarm and holiday i/p [Alarm and holiday Delete mode]


1
Make the timer to work in the display mode
Is today holiday
Yes
Exit from the system
No
Is the day of week set
No
Exit from the system
Yes
Get the multiple alarm i/p's [ie. hour, minute, duration]
a
IS Any of alarm i/p's equal to Synchronized RTC value
No
YES
2


2
Send Signal through relay to trigger the bell
IS this the last i/p
NO
a
YES
Stop

```
;ADDRESS
RTC             EQU     8000H           ; Address of Real Time Clo

;CONSTANTS
TMOD_VALUE      EQU     00010101B       ; TIMER 1 initialize code
TCNT_VALUE      EQU     0FAFFH          ; 1ms interrupt count valu

R0$     EQU     00H
R1$     EQU     01H
R2$     EQU     02H
R3$     EQU     03H
R4$     EQU     04H
R5$     EQU     05H
R6$     EQU     06H
R7$     EQU     07H

FUN1            EQU     10H
FUN2            EQU     FUN1+1
FUN3            EQU     FUN2+1
FUN4            EQU     FUN3+1
FUN5            EQU     FUN4+1
FUN6            EQU     FUN5+1
FUN7            EQU     FUN6+1

HOU_BUFF                EQU     FUN7+1
MIN_BUFF                EQU     HOU_BUFF+1
DU_BUFF                 EQU     MIN_BUFF+1

TEMP_HOU                EQU     DU_BUFF+1
TEMP_MIN                EQU     TEMP_HOU+1
TEMP_DUR                EQU     TEMP_MIN+1

HOU_BUFFX               EQU     TEMP_DUR+1
MIN_BUFFX               EQU     HOU_BUFFX+1

TEMP_HOUX               EQU     MIN_BUFFX+1
TEMP_MINX               EQU     TEMP_HOUX+1

FLK0            EQU     25H.0
FLK1            EQU     FLK0+1
FLK2            EQU     FLK1+1
FLK3            EQU     FLK2+1
FLK4            EQU     FLK3+1
FLK5            EQU     FLK4+1
FLK6            EQU     FLK5+1

FLTEST          EQU     FLK6+1
FLTEST1         EQU     FLTEST+1

FLINC           EQU     FLTEST1+1
FLENT           EQU     FLINC+1

FLDO            EQU     FLENT+1
FLLER           EQU     FLDO+1

FLK7            EQU     FLLER+1
FLLERX          EQU     FLK7+1

FLDURLOADED     EQU     FLLERX+1
```

```
FLWDS               EQU     FLSEC+1

FLSET_WE_DAY        EQU     29H

DISP                EQU     30H
CHAR2               EQU     DISP+2
CHAR3               EQU     CHAR2+2
CHAR4               EQU     CHAR3+2
DISP_PTR            EQU     DISP+0AH
BUFFER              EQU     DISP_PTR+1
BUFFER_PTR          EQU     BUFFER+5

SECSRT              EQU     45H
MINRT               EQU     SECSRT+2
HRSRT               EQU     MINRT+2
DATERT              EQU     HRSRT+2
MONTRT              EQU     DATERT+2
YEARRT              EQU     MONTRT+2
WD                  EQU     YEARRT+2

DURATION_BUFFER             EQU     55H

TEMPR                       EQU     56H

KEY_DEL                     EQU     57H

NUMB_KEYS           EQU     08H

MAX_TIME            EQU     44
MAX_MON             EQU     60

KEY_DEL_VALUE       EQU     50
MINS                EQU     4000H
HOURS               EQU     MINS+1
MONTH               EQU     HOURS+1
DAYX                EQU     MONTH+1
YEARX               EQU     DAYX+1
DURX                EQU     YEARX+1

BMINS               EQU     DURX+1
BHOURS              EQU     BMINS+1
BMONTH              EQU     BHOURS+1
BDAYX               EQU     BMONTH+1
BYEARX              EQU     BDAYX+1
BDURX               EQU     BYEARX+1

ST_TIME_BUFF        EQU     4050H

SORT_COUNT          EQU     4025H
SORT_COUNT1         EQU     SORT_COUNT+1
DEL_DATA            EQU     SORT_COUNT1+1
SORT_COUNTA         EQU     DEL_DATA+1

ST_TIME_BUFFX               EQU     4350H

SORT_COUNTX                 EQU     4030H
SORT_COUNTX1                EQU     SORT_COUNTX+1
DEL_DATAX                   EQU     SORT_COUNTX1+1
```

```
STORE_WEEK              EQU     DURATION_BUFFER1+1

        ORG     0000H

        JMP     START

;* External interrupt-0 *
        ORG     0003H
        RETI

;* Internal interrupt-0 *
        ORG     000BH
        RETI

        ORG     0013H
        RETI

        ORG     001BH
        JMP     TMR1_ISR        ; TIMER - 1 service routine

        ORG     0023H
        RETI                    ; For later modification

TMR1_ISR:
        PUSH    DPL
        PUSH    DPH

        CLR     TR1             ; Reload the Timer1
        MOV     TL1,#LOW (TCNT_VALUE)
        MOV     TH1,#HIGH (TCNT_VALUE)
        SETB    TR1
        CALL    KDI
        POP     DPH
        POP     DPL
        SETB    EA
        RETI

START:
        MOV     SP,#65H
        CLR     EA
        SETB    EA
        MOV     IE,#10001000B
        MOV     TMOD,#TMOD_VALUE
        MOV     TL1,#LOW (TCNT_VALUE)
        MOV     TH1,#HIGH (TCNT_VALUE)
        SETB    EA
        CALL    CLFL                    ;CLEARS ALL KEY FLAGS
        CALL    SEFL
        CALL    INT_FUC
        CALL    LOAD_SORT_COUNT1
        CALL    LOAD_SORT_COUNTX1
        CALL    DUR_WAIT
        CALL    L_SORT_C_A
        MOV     R2,#0FFH
        MOV     R1,#NUMB_KEYS
        MOV     R3,#08
        SETB    TR1
```

```
X:
        SJMP    XSTART

L_SORT_C_A:
        MOV     DPTR,#SORT_COUNT        ;LOAD SORT COUNT
        MOVX    A,@DPTR
        MOV     DPTR,#SORT_COUNTA
        MOVX    @DPTR,A
        RET

DUR_WAIT:
        MOV     DPTR,#DURATION_BUFFER1  ;DURATION WAIT 65 SECONDS
        MOV     A,#65
        MOVX    @DPTR,A
        RET

INT_FUC:                                ;NUMBER OF FUNCTIONS
        MOV     R0,#FUN1
        MOV     @R0,#06
        INC     R0
        MOV     @R0,#03
        INC     R0
        MOV     @R0,#00
        INC     R0
        MOV     @R0,#02
        INC     R0
        MOV     @R0,#00
        RET

SEFL:
        SETB    FLTEST1
        SETB    FLK0
        SETB    FLK1
        SETB    FLK2
        SETB    FLK3
        SETB    FLK4
        SETB    P3.3
        RET

PUTT:
        MOV     R0,#BUFFER
        MOV     DPTR,#BMINS
        MOVX    A,@DPTR
        MOV     @R0,A
        RET
PUTTH:
        MOV     R0,#BUFFER+1
        MOV     DPTR,#BMINS+1
        MOVX    A,@DPTR
        MOV     @R0,A
        RET

PUTTM:
        MOV     R0,#BUFFER+2
        MOV     DPTR,#BMINS+2
        MOVX    A,@DPTR
```

```
PUTTD:
        MOV     R0,#BUFFER+3
        MOV     DPTR,#BMINS+3
        MOVX    A,@DPTR
        MOV     @R0,A
        RET

KDI:                                    ;KEYBOARD AND DISPLAY UPDATION
        CALL    DISP_BLK
        DEC     R1

        CALL    KEY_SCAN

        CALL    SELECT_DIGIT

        CALL    DISP_DIGIT

        INC     R1
        DJNZ    R1,DISP11
        MOV     R1,#07

DISP11:                                 ;DISPLAYS REAL TIME CLOCK
        MOV     DPTR,#RTC+15
        MOV     A,#00000100B
        MOVX    @DPTR,A

SLAVE:
        CJNE    R3,#08,TIMEX1
        MOV     DPTR,#RTC+13
        MOV     A,#00000000B
        MOVX    @DPTR,A
        MOV     R7,#0DH
        MOV     DPTR,#RTC+13
        MOV     A,#00000001B
        MOVX    @DPTR,A
        MOV     DPTR,#RTC+13
        MOVX    A,@DPTR
        JB      ACC.1,SLAVE
        MOV     R0,#SECSRT
        MOV     DPTR,#RTC
RPT:    MOVX    A,@DPTR
        ANL     A,#0FH
        MOV     @R0,A
        INC     R0
        INC     DPTR
        DJNZ    R7,RPT
        MOV     DPTR,#RTC+13
        MOV     A,#00000000B
        MOVX    @DPTR,A

        CALL    CHK_TIME_CODE
        JB      FLDURLOADED,DAYAX
JUG:

        CALL    TIMEX
```

```
;-------------------------------------------------------
;PROCEDURE TO FIND TIME MATCH AND ACTIVATE BELLING
;-------------------------------------------------------
DAYAX:
        CALL    CHK_SEC
        JNB     FLSEC,COW
        JNB     FLK7,JUG
        CLR     FLK7
        JMP     DAYAX2
COW:
        JB      FLK7,JUG
        SETB    FLK7
DAYAX2:
        MOV     DPTR,#DURATION_BUFFER1
        MOVX    A,@DPTR
        JZ      DAYAX1
        DEC     A
        MOVX    @DPTR,A
        MOV     R0,#DURATION_BUFFER
        MOV     A,@R0
        JZ      JUG1
        CLR     P3.3
        DEC     A
        MOV     @R0,A
        JMP     JUG
JUG1:
        SETB    P3.3
        JMP     JUG

DAYAX1:
        MOV     A,#65
        MOVX    @DPTR,A
        CLR     FLDURLOADED
        JMP     JUG

CHK_SEC:
        MOV     R0,#SECSRT
        MOV     A,@R0
        JB      ACC.0,LRCC
        SETB    FLSEC
        RET
LRCC:
        CLR     FLSEC
        RET

SELECT_DIGIT:                           ;DIGIT TO BE ACTIVATED
        MOV     DPTR,#SEL_DIG
        MOV     B,#03
        MOV     A,R1
        MUL     AB
        JMP     @A+DPTR
SEL_DIG:
        LJMP    DISP6
        LJMP    DISP5
        LJMP    DISP4
        LJMP    DISP3
        LJMP    DISP2
        LJMP    DISP1
        LJMP    DISP0
        LJMP    DISP7
```

```
DISP0:
        CLR     P3.0
        CLR     P3.1
        CLR     P3.2
        RET

DISP1:
        SETB    P3.0
        CLR     P3.1
        CLR     P3.2
        RET

DISP2:
        CLR     P3.0
        SETB    P3.1
        CLR     P3.2
        RET

DISP3:
        SETB    P3.0
        SETB    P3.1
        CLR     P3.2
        RET

DISP4:
        CLR     P3.0
        CLR     P3.1
        SETB    P3.2
        RET
DISP5:
        SETB    P3.0
        CLR     P3.1
        SETB    P3.2
        RET
DISP6:
        CLR     P3.0
        SETB    P3.1
        SETB    P3.2
        RET

DISP7:
        SETB    P3.0
        SETB    P3.1
        SETB    P3.2
        RET

DISP_DIGIT:
        MOV     A,R1
        MOV     R0,#DISP
        ADD     A,R0
        MOV     R0,A
        MOV     A,@R0
        MOV     P1,A
        RET

DISP_BLK:
        MOV     A,#BLANK_CODE
        MOV     P1,A
        RET
```

```
KEY_SCAN:                    ;KEY BOARD SENCING WITH DEBOUNCE
          JNB       FLTEST1,TEST3

          JNB       FLTEST,TEST
          MOV       A,R2
          CJNE      A,R1$,TEST4

TEST:
          JB        P3.4,TEST1
          JMP       LOCATE

TEST1:
          CLR       FLTEST
          JMP       TEST4
TEST3:
          MOV       RO,#KEY_DEL
          DEC       @RO
          CJNE      @RO,#00,TEST4
          MOV       @RO,#KEY_DEL_VALUE
          SETB      FLTEST1
TEST4:
          RET

LOCATE:
          CPL       FLTEST
          JNB       FLTEST,TEST2
          CLR       FLTEST1
          MOV       2,R1
          JMP       TEST4
TEST2:
          SETB      FLTEST1
          MOV       DPTR,#CONTROL
          MOV       B,#03H
          MOV       A,R1
          MUL       AB
          JMP       @A+DPTR


;-------------------------------------------------------------------------
;PROCEDURE FOR FINDING CODE MATCH IN BUFFER & LOADS THE DURATION
;-------------------------------------------------------------------------

CHK_TIME_CODE:
          PUSH      5
          PUSH      6
          JB        FLDURLOADED,RTYYY

          MOV       DPTR,#SORT_COUNTA
          MOVX      A,@DPTR
          JZ        CTC3
          MOV       R5,A
;CTC1:
          DEC       R5
          CALL      ADDR_CALC
          MOVX      A,@DPTR
          MOV       R6,A
          MOV       RO,#HRSRT
          MOV       A,@RO
          ANL       A,#0FH
          MOV       B,A
```

```
            INC     R0
            MOV     A,@R0
            ANL     A,#0FH
            SWAP    A
            ORL     A,B
            CJNE    A,R6$,CTC2
            INC     DPTR
            MOVX    A,@DPTR
            MOV     R6,A
            MOV     R0,#MINRT
            MOV     A,@R0
            ANL     A,#0FH
            MOV     B,A
            INC     R0
            MOV     A,@R0
            ANL     A,#0FH
            SWAP    A
            ORL     A,B
            CJNE    A,R6$,CTC2

            INC     DPTR
            MOVX    A,@DPTR
            MOV     R0,#DURATION_BUFFER
            MOV     @R0,A

;           JMP     CHK_DAY_DATE                ;CHECK

            SETB    FLDURLOADED
RTYYY:
            POP     6
            POP     5

            RET
CTC2:
            INC     R5
            DJNZ    R5,CTC3
            CALL    L_SORT_C_A
            POP     6
            POP     5
            RET
CTC3:
            MOV     A,R5
            MOV     DPTR,#SORT_COUNTA
            MOVX    @DPTR,A
            POP     6
            POP     5

            RET
;-----------------
;COMPARES W AND C
;-----------------
CHK_DAY_DATE:

            MOV     DPTR,#STORE_WEEK            ;CHECK
            MOVX    A,@DPTR
            MOV     R6,A
            CALL    ABCD
            ANL     A,R6
            JZ      CHK_NEXT
```

```
        POP     6
        POP     5

        RET                                     ;CHECK

ABCD:
        MOV     DPTR,#FIND_EQUAL
        MOV     R0,#WD
        MOV     A,@R0
        MOV     B,#03
        MUL     AB
        JMP     @A+DPTR

FIND_EQUAL:
        LJMP    S1
        LJMP    M1
        LJMP    AT1
        LJMP    W1
        LJMP    H1
        LJMP    F1
        LJMP    XA1
        LJMP    S33                     ;SAFE

S1:
        MOV     A,#00000001B
        RET

M1:
        MOV     A,#00000010B
        RET

AT1:
        MOV     A,#00000100B
        RET

W1:
        MOV     A,#00001000B
        RET

H1:
        MOV     A,#00010000B
        RET

F1:
        MOV     A,00100000B
        RET

XA1:
        MOV     A,01000000B
        RET

S33:
        RET

CHK_NEXT:
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        JZ      CN21
        MOV     R5,A
```

```
CN2:
        DEC     R5
        CALL    ADDR_CALCY
        MOVX    A,@DPTR
        MOV     R6,A
        MOV     RO,#MONTRT
        MOV     A,@RO                                   ;CHECK\
        ANL     A,#OFH
        MOV     B,A
        INC     RO
        MOV     A,@RO
        ANL     A,#OFH
        SWAP    A
        ORL     A,B
        CJNE    A,R6$,CN1
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R6,A
        MOV     RO,#DATERT
        MOV     A,@RO                                   ;CHECK\
        ANL     A,#OFH
        MOV     B,A
        INC     RO
        MOV     A,@RO
        ANL     A,#OFH
        SWAP    A
        ORL     A,B
        CJNE    A,R6$,CN1

        POP     6
        POP     5

        RET

CN1:
        INC     R5
        DJNZ    R5,CN2

CN21:
        JMP     RTYYY

;-----------------------------------------------------------------
;LOAD DISPLAY FROM RTC DEPENDING UPON THE FLAG INPUT
;-----------------------------------------------------------------
TIMEX:

        JNB     FLK6,RTTO
RTT1:
        CALL    LOAD_RSEC
        CALL    LOAD_RMIN
        CALL    LOAD_RHOU
        CALL    LOAD_RWEEK
        RET

RTTO:
        CALL    LOAD_RYEAR
        CALL    LOAD_RMON
        CALL    LOAD_RDATE
        RET
```

```
;-----------------------------------------
;VARIOUS FUNCTIONS TO DISPLAY CALENDER
;-----------------------------------------

LOAD_RSEC:
          MOV       R0,#SECSRT
          MOV       A,@R0
          MOV       R0,#CHAR4+1
          JB        ACC.0,LRC
          MOV       @R0,#BLANK_CODE
          RET
LRC:
          MOV       @R0,#DASH_CODE
          RET

LOAD_RMIN:
          MOV       DPTR,#DISPL_TBL
          MOV       R0,#MINRT
          MOV       A,@R0
          MOVC      A,@A+DPTR
          MOV       R0,#DISP
          MOV       @R0,A
          MOV       R0,#MINRT+1
          MOV       A,@R0
          MOVC      A,@A+DPTR
          MOV       R0,#DISP+1
          MOV       @R0,A
          RET

LOAD_RHOU:
          MOV       DPTR,#DISPL_TBL
          MOV       R0,#HRSRT
          MOV       A,@R0
          MOVC      A,@A+DPTR
          MOV       R0,#CHAR2
          MOV       @R0,A
          MOV       R0,#HRSRT+1
          MOV       A,@R0
          MOVC      A,@A+DPTR
          MOV       R0,#CHAR2+1
          MOV       @R0,A
          RET

LOAD_RWEEK:
          MOV       DPTR,#WEEKJMP
          MOV       R0,#WD
          MOV       A,@R0
          MOV       B,#03
          MUL       AB
          JMP       @A+DPTR

WEEKJMP:
          LJMP      WEEK0
          LJMP      WEEK1
          LJMP      WEEK2
          LJMP      WEEK3
          LJMP      WEEK4
          LJMP      WEEK5
          LJMP      WEEK6
```

```
WEEK0:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#S_CODE
        RET

WEEK1:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#O_CODE
        INC     R0
        MOV     @R0,#M_CODE
        RET

WEEK2:
        MOV     R0,#CHAR3
        MOV     @R0,#E_CODE
        INC     R0
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#T_CODE
        RET

WEEK3:
        MOV     R0,#CHAR3
        MOV     @R0,#D_CODE
        INC     R0
        MOV     @R0,#E_CODE
        INC     R0
        MOV     @R0,#W_CODE
        RET

WEEK4:
        MOV     R0,#CHAR3
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#H_CODE
        INC     R0
        MOV     @R0,#T_CODE
        RET

WEEK5:
        MOV     R0,#CHAR3
        MOV     @R0,#I_CODE
        INC     R0
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#F_CODE
        RET

WEEK6:
        MOV     R0,#CHAR3
        MOV     @R0,#T_CODE
        INC     R0
```

```
        MOV     @R0,#A_CODE
        INC     R0
        MOV     @R0,#S_CODE
        RET

LOAD_RYEAR:
        MOV     DPTR,#DISPL_TBL
        MOV     R0,#YEARRT
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     R0,#DISP
        MOV     @R0,A
        MOV     R0,#YEARRT+1
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     R0,#DISP+1
        MOV     @R0,A
        RET

LOAD_RMON:
        MOV     DPTR,#DISPL_TBL
        MOV     R0,#MONTRT
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     R0,#CHAR2
        MOV     @R0,A
        MOV     R0,#MONTRT+1
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     R0,#CHAR2+1
        MOV     @R0,A
        RET

LOAD_RDATE:
        MOV     DPTR,#DISPL_TBL
        MOV     R0,#DATERT
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     R0,#CHAR3
        MOV     @R0,A
        MOV     R0,#DATERT+1
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     R0,#CHAR3+1
        MOV     @R0,A
        RET

;-------------------------------------------------------------------------------

CLFL:
        CLR     FLINC
        CLR     FLENT
        CLR     FLDO
        CLR     FLK5
        CLR     FLK6
        CLR     FLTEST
        CLR     FLDURLOADED
        CLR     FLWDS
        RET
```

```
;-----------------------------------------------
;TRANSFERS THE CONTROL TO THE REQUIRED ROUTINE
;-----------------------------------------------

CONTROL:
        LJMP    SFK0
        LJMP    SFK1
        LJMP    SFK2
        LJMP    SFK3
        LJMP    SFK4
        LJMP    SFK5
        LJMP    SFK6

SFK2:
        MOV     R7,#00
        CLR     FLTEST
        SETB    FLD0
        DJNZ    R3,T2
        MOV     R3,#08
T2:
        JMP     TEST4

SFK3:
        MOV     R7,#01H
        CLR     FLTEST
        SETB    FLD0
        SETB    FLK5
        JMP     TEST4
;       JMP     XSTART

SFK4:
        MOV     R7,#02H
        CLR     FLTEST
        SETB    FLD0
        SETB    FLINC
        JMP     TEST4
;       JMP     XSTART

SFK5:
        MOV     R7,#03H
        CLR     FLTEST
        SETB    FLD0
        SETB    FLENT
        JMP     TEST4
;       JMP     XSTART

SFK0:
        MOV     R7,#04H
        JMP     TEST4
        SJMP    $

SFK1:
        MOV     R7,#05H
        JMP     TEST4
        SJMP    $

SFK6:
        MOV     R7,#06H
        JMP     TEST4
        SJMP    $
```

```
SFK7:
        MOV     R7,#07H
        JMP     TEST4
        SJMP    $

;-------------------------------------------------------------------
;REGISTER R3 CONTAINS THE MODE AND REGISTER R4 CONTAINS SUBMODE
;R3 IS CONTROLLED BY KEY 1 (MAXI FUNCTION OF 7)
;R4 IS CONTROLLED BY KEY 2 (MAXI FUNCTION OF 7)
;-------------------------------------------------------------------

TINX:
        MOV     DPTR,#CNTMAIN
        MOV     B,#03H
        MOV     A,R3
        MUL     AB
        JMP     @A+DPTR

CNTMAIN:
        LJMP    RUN
        LJMP    CON7
        LJMP    CON6
        LJMP    CON5
        LJMP    CON4
        LJMP    CON3
        LJMP    CON2
        LJMP    CON1
        LJMP    RUN

RUN:
        MOV     R7,#0
        JNB     FLK5,RUN1
        CPL     FLK6
RUN1:
        CLR     FLD0
        CLR     FLK5
        JMP     X

CON1:
        MOV     R0,#CHAR2
        MOV     @R0,#S_CODE
        INC     R0
        MOV     @R0,#T_CODE
        JMP     SUBCNT

LOFUN1:
        MOV     R0,#FUN1
        MOV     A,@R0
        MOV     R4,A
        RET

SUBCNT:
        CALL    LOFUN1
        MOV     DPTR,#SUBCOOL
        MOV     B,#03H
        MOV     A,R4
        MUL     AB
        JMP     @A+DPTR
```

```
SUBCOOL:
        LJMP    SUCON6
        LJMP    SUCON5
        LJMP    SUCON4
        LJMP    SUCON3
        LJMP    SUCON2
        LJMP    SUCON1
        LJMP    SUCONO

SUCON6:
        CLR     FLDO
        JMP     X

SUCONO:
        MOV     RO,#CHAR3
        MOV     @RO,#R_CODE
        INC     RO
        MOV     @RO,#E_CODE
        INC     RO
        MOV     @RO,#Y_CODE
        MOV     DPTR,#BYEARX
        CALL    DOG
        JNB     FLINC,SUCO1
        CALL    YEAR
        CALL    DOG
SUCO1:
        JNB     FLENT,SUCO
        DJNZ    R4,SUCOX
        MOV     R4,#06
SUCOX:
        CALL    LOADY
        CALL    SAFUN1
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT
SUCO:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

SAFUN1:
        MOV     RO,#FUN1
        MOV     A,R4
        MOV     @RO,A
        RET

LOADY:
        MOV     DPTR,#BYEARX
        MOVX    A,@DPTR
        MOV     B,A
        ANL     A,#OFH
        MOV     DPTR,#RTC+10
        MOVX    @DPTR,A
        MOV     A,B
        SWAP    A
        ANL     A,#OFH
        INC     DPTR
        MOVX    @DPTR,A
        RET
```

```
DOG:
        MOVX    A,@DPTR
        MOV     R0,#BUFFER
        MOV     @R0,A
        CALL    FILL_NUM
        RET

SUCON1:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#O_CODE
        INC     R0
        MOV     @R0,#M_CODE
        MOV     DPTR,#BMONTH
        CALL    DOG
        JNB     FLINC,SUC11
        CALL    MON
        CALL    DOG
SUC11:
        JNB     FLENT,SUC1
        DJNZ    R4,SUC1X
        MOV     R4,#06
SUC1X:
        CALL    LOADMO
        CALL    SAFUN1
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT
SUC1:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

LOADMO:
        MOV     DPTR,#BMONTH
        MOVX    A,@DPTR
        MOV     B,A
        ANL     A,#0FH
        MOV     DPTR,#RTC+8
        MOVX    @DPTR,A
        MOV     A,B
        SWAP    A
        ANL     A,#0FH
        INC     DPTR
        MOVX    @DPTR,A
        RET
SUCON2:
        MOV     R0,#CHAR3
        MOV     @R0,#T_CODE
        INC     R0
        MOV     @R0,#A_CODE
        INC     R0
        MOV     @R0,#D_CODE
        MOV     DPTR,#BDAYX
        CALL    DOG
        JNB     FLINC,SUC21
        CALL    DAY
        CALL    DOG
```

```
SUC21:
        JNB     FLENT,SUC2
        DJNZ    R4,SUC2X
        MOV     R4,#06

SUC2X:
        CALL    LOADD
        CALL    SAFUN1
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT
SUC2:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

LOADD:
        MOV     DPTR,#BDAYX
        MOVX    A,@DPTR
        MOV     B,A
        ANL     A,#0FH
        MOV     DPTR,#RTC+6
        MOVX    @DPTR,A
        MOV     A,B
        SWAP    A
        ANL     A,#0FH
        INC     DPTR
        MOVX    @DPTR,A
        RET

SUCON3:
        MOV     R0,#CHAR3
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#0_CODE
        INC     R0
        MOV     @R0,#H_CODE
        MOV     DPTR,#BHOURS
        CALL    DOG
        JNB     FLINC,SUC31
        CALL    HOU
        CALL    DOG
SUC31:
        JNB     FLENT,SUC3
        DJNZ    R4,SUC3X
        MOV     R4,#06
SUC3X:
        CALL    LOADH
        CALL    SAFUN1
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT
SUC3:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X
```

```
LOADH:
        MOV     DPTR,#BHOURS
        MOVX    A,@DPTR
        MOV     B,A
        ANL     A,#0FH
        MOV     DPTR,#RTC+4
        MOVX    @DPTR,A
        MOV     A,B
        SWAP    A
        ANL     A,#0FH
        INC     DPTR
        MOVX    @DPTR,A
        RET

SUCON4:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#I_CODE
        INC     R0
        MOV     @R0,#M_CODE
        MOV     DPTR,#BMINS
        CALL    DOG
        JNB     FLINC,SUC41
        CALL    MIN
        CALL    DOG
SUC41:
        JNB     FLENT,SUC4
        DJNZ    R4,SUC4X
        MOV     R4,#06
SUC4X:
        CALL    LOADMI
        CALL    SAFUN1
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT
SUC4:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

LOADMI:
        MOV     DPTR,#BMINS
        MOVX    A,@DPTR
        MOV     B,A
        ANL     A,#0FH
        MOV     DPTR,#RTC+2
        MOVX    @DPTR,A
        MOV     A,B
        SWAP    A
        ANL     A,#0FH
        INC     DPTR
        MOVX    @DPTR,A
        RET
SUCON5:
        PUSH    5
        MOV     R0,#FUN3
        MOV     A,@R0
        MOV     R5,A
```

```
SUC5X:
        CALL    LOAD_WEEKX
        MOV     A,R5
        MOV     R0,#DISP
        MOV     DPTR,#DISPL_TBL
        MOVC    A,@A+DPTR
        MOV     @R0,A
        MOV     A,#00
        MOVC    A,@A+DPTR
        INC     R0
        MOV     @R0,A

        JNB     FLENT,SUC5X1
        CALL    LOADW
        DJNZ    R4,SUC5X4
        MOV     R4,#06
SUC5X4:
        CALL    SAFUN1
        CLR     FLENT
        CLR     FLINC
        POP     5
        JMP     SUBCNT
SUC5X1:
        JNB     FLINC,SUC5X2
        INC     R5
        CJNE    R5,#07,SUC5X3
        MOV     R5,#00
SUC5X3:
        CALL    SAFUN3
        CLR     FLENT
        CLR     FLINC
        POP     5
        JMP     SUBCNT
SUC5X2:
        CALL    SAFUN1
        CALL    SAFUN3
        CLR     FLDO
        CLR     FLENT
        CLR     FLINC
        POP     5

        JMP     X

SAFUN3:
        MOV     R0,#FUN3
        MOV     A,R5
        MOV     @R0,A
        RET
LOADW:
        MOV     A,R5
        ANL     A,#0FH
        MOV     DPTR,#RTC+12
        MOVX    @DPTR,A
        RET
LOAD_WEEKX:
        MOV     DPTR,#WEEKJMP
        MOV     A,R5
        MOV     B,#03
        MUL     AB
        JMP     @A+DPTR
```

```
FIND_CODES:
        MOV     DPTR,#DISPL_TBL
        MOVC    A,@A+DPTR
        MOV     R0,R4$
        MOV     @R0,A
        RET
MIN:
        PUSH    5
        CLR     C
        MOV     DPTR,#MINS
        MOVX    A,@DPTR
        INC     A
        MOV     B,A
        MOV     A,#59
        SUBB    A,B
        JNC     TOT
        MOV     B,#0
TOT:
        MOV     A,B
        MOVX    @DPTR,A
        MOV     R5,A
        CALL    BINTIN
        MOV     DPTR,#BMINS
        MOV     A,R5
        MOVX    @DPTR,A
        POP     5
        RET
HOU:
        PUSH    5
        CLR     C
        MOV     DPTR,#HOURS
        MOVX    A,@DPTR
        INC     A
        MOV     B,A
        MOV     A,#23
        SUBB    A,B
        JNC     TOT1
        MOV     B,#0
TOT1:
        MOV     A,B
        MOVX    @DPTR,A
        MOV     R5,A
        CALL    BINTIN
        MOV     DPTR,#BHOURS
        MOV     A,R5
        MOVX    @DPTR,A
        POP     5
        RET
MON:
        PUSH    5
        CLR     C
        MOV     DPTR,#MONTH
        MOVX    A,@DPTR
        INC     A
        MOV     B,A
        MOV     A,#12

        SUBB    A,B
        JNC     TOT2
        MOV     B,#1
```

```
TOT2:
        MOV     A,B
        MOVX    @DPTR,A
        MOV     R5,A
        CALL    BINTIN
        MOV     DPTR,#BMONTH
        MOV     A,R5
        MOVX    @DPTR,A
        POP     5
        RET

DAY:
        PUSH    5
        CLR     C
        MOV     DPTR,#DAYX
        MOVX    A,@DPTR
        INC     A
        MOV     B,A
        MOV     A,#31
        SUBB    A,B
        JNC     TOT3
        MOV     B,#1

TOT3:
        MOV     A,B
        MOVX    @DPTR,A
        MOV     R5,A
        CALL    BINTIN
        MOV     DPTR,#BDAYX
        MOV     A,R5
        MOVX    @DPTR,A
        POP     5
        RET

YEAR:
        PUSH    5
        CLR     C
        MOV     DPTR,#YEARX
        MOVX    A,@DPTR
        INC     A
        MOV     B,A
        MOV     A,#99
        SUBB    A,B
        JNC     TOT4
        MOV     B,#1

TOT4:
        MOV     A,B
        MOVX    @DPTR,A
        MOV     R5,A
        CALL    BINTIN
        MOV     DPTR,#BYEARX
        MOV     A,R5
        MOVX    @DPTR,A
        POP     5

        RET
```

```
DUX:
        PUSH    5
        CLR     C
        MOV     DPTR,#DURX
        MOVX    A,@DPTR
        INC     A
        MOV     B,A
        MOV     A,#59
        SUBB    A,B
        JNC     TOT5
        MOV     B,#0
TOT5:
        MOV     A,B
        MOVX    @DPTR,A
        MOV     R5,A
        CALL    BINTIN
        MOV     DPTR,#BDURX
        MOV     A,R5
        MOVX    @DPTR,A
        POP     5
        RET

BINTIN:
        PUSH    R3$
        PUSH    R4$
        MOV     R3,#0
        MOV     R4,#8H
BIN2BCD2:
        CLR     C
        MOV     A,R5
        RLC     A
        MOV     R5,A
        MOV     A,R3
        ADDC    A,R3
        DA      A
        MOV     R3,A
        DBNZ    R4,BIN2BCD2
        MOV     R5,3
        POP     R4$
        POP     R3$
        RET

FILL_NUM:
        PUSH    R3$
        PUSH    R4$
        MOV     R4,#DISP
        MOV     R3,#BUFFER
        MOV     A,#0FH
        MOV     R0,R3$
        ANL     A,@R0
        CALL    FIND_CODES
        INC     R4
        MOV     R0,R3$
        MOV     A,@R0
        SWAP    A
        ANL     A,#0FH
        CALL    FIND_CODES
        POP     R4$
        POP     R3$
        RET
```

```
;-------------------------------------------------------------------------
;CONTROL FUNCTION TWO SETS
;-------------------------------------------------------------------------

CON2:
        MOV     R4,#06
        CALL    SAFUN1
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,#MAX_TIME,D_FULL1
        CALL    D_FULL
        JMP     TUC3

D_FULL1:
        CALL    DISP_COUNT
        JNB     FLK5,X1X
        JMP     SOR_START

X1X:
        JMP     SUBCNT2

LOFUN2:
        MOV     R0,#FUN2
        MOV     A,@R0
        MOV     R4,A
        RET

SUBCNT2:
        CALL    DISP_COUNT              ;
        CALL    LOFUN2
        MOV     DPTR,#SUBCOOL2
        MOV     B,#03H
        MOV     A,R4
        MUL     AB
        JMP     @A+DPTR

SUBCOOL2:
        LJMP    TUCON3
        LJMP    TUCON2
        LJMP    TUCON1
        LJMP    TUCON0

TUCON0:
        MOV     R0,#CHAR3
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#O_CODE
        INC     R0
        MOV     @R0,#H_CODE
        MOV     DPTR,#BHOURS
        CALL    DOG
        JNB     FLINC,TUC31
        CALL    HOU
        CALL    DOG

TUC31:
        JNB     FLENT,TUC3
        DJNZ    R4,TUC3X
        MOV     R4,#03
```

```
TUC3X:
        CALL    SAFUN2
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT2
TUC3:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

TUCON1:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#I_CODE
        INC     R0
        MOV     @R0,#M_CODE
        MOV     DPTR,#BMINS
        CALL    DOG
        JNB     FLINC,TUC41
        CALL    MIN
        CALL    DOG
TUC41:
        JNB     FLENT,TUC4
        DJNZ    R4,TUC4X
        MOV     R4,#03
TUC4X:
        CALL    SAFUN2
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT2
TUC4:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X
D_FULL:
        MOV     R0,#DISP
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#L_CODE
        INC     R0
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#F_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        RET
SAFUN2:
        MOV     R0,#FUN2
        MOV     A,R4
        MOV     @R0,A
        RET
```

```
TUCON2:
        MOV     R0,#CHAR3
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#D_CODE
        MOV     DPTR,#BDURX
        CALL    DOG
        JNB     FLINC,TUC31X
        CALL    DUX
        CALL    DOG
TUC31X:
        JNB     FLENT,TUC3X1
        DJNZ    R4,TUC3XX
        MOV     R4,#03
TUC3XX:
        CALL    SAFUN2
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT2
TUC3X1:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

TUCON3:
        CLR     FLINC
        CLR     FLENT
        CLR     FLDO
        JMP     X


FILL_NUM1:
        PUSH    R3$
        PUSH    R4$
        MOV     R4,#CHAR2
        MOV     R3,#BUFFER
        MOV     A,#0FH
        MOV     R0,R3$
        ANL     A,@R0
        CALL    FIND_CODES
        INC     R4
        MOV     R0,R3$
        MOV     A,@R0
        SWAP    A
        ANL     A,#0FH
        CALL    FIND_CODES
        POP     R4$
        POP     R3$
        RET

DISP_COUNT:
        PUSH    5
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        MOV     R5,A
        CALL    BINTIN
        MOV     R0,#BUFFER
```

```
        MOV     @R0,5
        CALL    FILL_NUM1
        POP     5
        RET

;-------------------------------------------------------------------------
;THIS PROCEDURE SORTS THE GIVEN VALUE IMMEDEATLY AFTER KEYING IN
;R5 KEEPS TRACK OF THE COUNT OF THE VALUE
;-------------------------------------------------------------------------

SOR_START:
        PUSH    3
        PUSH    4
        PUSH    5

        MOV     R0,#HOU_BUFF
        MOV     DPTR,#BHOURS
        MOVX    A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     DPTR,#BMINS
        MOVX    A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     DPTR,#BDURX
        MOVX    A,@DPTR
        MOV     @R0,A

        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        MOV     R5,A

        CJNE    R5,#0,CONTINUE
        CALL    HOU_CHK
        MOV     DPTR,#ST_TIME_BUFF
        MOVX    @DPTR,A
        CALL    MIN_CHK
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        CALL    DURATION
        MOVX    @DPTR,A
        CALL    PUT_COUNT
        POP     5
        POP     4
        POP     3
        JMP     XX

PUT_COUNT:
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        INC     A
        MOVX    @DPTR,A
        RET

HOU_CHK:
        MOV     R0,#HOU_BUFF
        MOV     A,@R0
        RET
```

```
MIN_CHK:
        MOV     R0,#MIN_BUFF
        MOV     A,@R0
        RET

DURATION:
        MOV     R0,#DU_BUFF
        MOV     A,@R0
        RET

CONTINUE:
        JMP     CHK_EQU
CONTINUE2:
        CLR     C
        DJNZ    R5,HITLER
        SETB    FLLER
HITLER:
        CALL    HOU_CHK
        MOV     R3,A
        CALL    ADDR_CHK_HOU
        MOV     B,A
        SUBB    A,R3
        JC      MINUTES1
        MOV     A,B
        CJNE    A,R3$,CONTINUE1

        CALL    ADDR_CHK_MIN
        MOV     R4,A
        CALL    MIN_CHK
        CJNE    A,R4$,MINUTES
        POP     5
        POP     4
        POP     3

        JMP     XX

MINUTES1:
        INC     R5
        CALL    ADDR_CALC
        CALL    HOU_CHK
        MOVX    @DPTR,A
        INC     DPTR
        CALL    MIN_CHK
        MOVX    @DPTR,A
        INC     DPTR
        CALL    DURATION
        MOVX    @DPTR,A
        CALL    PUT_COUNT
        CLR     FLLER
        POP     5
        POP     4
        POP     3

        JMP     XX

MINUTES2:
        CALL    ADDR_CALC
        CALL    HOU_CHK
        MOVX    @DPTR,A
        INC     DPTR
```

p 207

```
        CALL    MIN_CHK
        MOVX    @DPTR,A
        INC     DPTR
        CALL    DURATION
        MOVX    @DPTR,A
        CALL    PUT_COUNT
        CLR     FLLER
        RET

CONTINUE1:
        CALL    EXCHANGE
        JNB     FLLER,HITLER1
        CALL    MINUTES2
HITLER1:
        INC     R5
        DJNZ    R5,CONTINUE2
        POP     5
        POP     4
        POP     3

        JMP     XX

MINUTES:
        CLR     C
        CALL    MIN_CHK
        MOV     R4,A
        CALL    ADDR_CHK_MIN
        SUBB    A,R4
        JC      MINUTES1
        CALL    EXCHANGE
        JNB     FLLER,HITLER2
        CALL    MINUTES2
HITLER2:
        INC     R5
        DJNZ    R5,HITLER3
        POP     5
        POP     4
        POP     3

        JMP     XX

HITLER3:
        JMP     CONTINUE2

ADDR_CHK_HOU:
        CALL    ADDR_CALC
        MOVX    A,@DPTR
        RET
ADDR_CHK_MIN:
        CALL    ADDR_CALC
        INC     DPTR
        MOVX    A,@DPTR
        RET
ADDR_CHK_DUR:
        CALL    ADDR_CALC
        INC     DPTR
        INC     DPTR
        MOVX    A,@DPTR
        RET
```

```
EXCHANGE:
          MOV       R0,#TEMP_HOU
          CALL      ADDR_CHK_HOU
          MOV       @R0,A
          INC       R0
          CALL      ADDR_CHK_MIN
          MOV       @R0,A
          INC       R0
          CALL      ADDR_CHK_DUR
          MOV       @R0,A
          INC       R5
          CALL      ADDR_CALC
          MOV       R0,#TEMP_HOU
          MOV       A,@R0
          MOVX      @DPTR,A
          INC       R0
          INC       DPTR
          MOV       A,@R0
          MOVX      @DPTR,A
          INC       R0
          INC       DPTR
          MOV       A,@R0
          MOVX      @DPTR,A
          DEC       R5
          RET

ADDR_CALC:
          CLR       C
          MOV       DPTR,#ST_TIME_BUFF
          MOV       B,#03
          MOV       A,R5
          MUL       AB
          ADD       A,DPL
          JC        ADDR_CALC1
          MOV       DPL,A
          RET

ADDR_CALC1:
          MOV       DPL,A
          INC       DPH
          RET

CHK_EQU:
          CALL      HOU_CHK
          MOV       R3,A
          CALL      MIN_CHK
          MOV       R4,A

CHK_EQU1:
          DEC       R5
          CALL      ADDR_CHK_HOU
          CJNE      A,R3$,ZORROR
          CALL      ADDR_CHK_MIN
          CJNE      A,R4$,ZORROR
          POP       5
          POP       4
          POP       3
          JMP       XX
```

```
ZORROR:
        INC     R5
        DJNZ    R5,CHK_EQU1
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        MOV     R5,A
        JMP     CONTINUE2

XX:
        CLR     FLD0
        CLR     FLK5
        CALL    DISP_COUNT

        JMP     X

;-----------------------------------------------
;PROCEDURE TO DELETE THE VALUES IN THE BUFFER
;-----------------------------------------------

CON3:
        MOV     R4,#03
        CALL    SAFUN2
        CALL    L_SORT_C_A

        PUSH    5
        PUSH    6

        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,#00,STARX
        CALL    FILL_ZERO
        JMP     XTR2

STARX:
        MOV     DPTR,#SORT_COUNT1
        MOVX    A,@DPTR
        MOV     R5,A
        INC     A
        MOV     R6,A
        JZ      VINMGA

VINMGA:
        JNB     FLK5,VINMGA1

        MOV     DPTR,#DEL_DATA
        INC     R5
        MOV     A,R5
        MOVX    @DPTR,A
        DEC     R5
        CALL    START_DEL
        CALL    LOAD_SORT_COUNT2
        MOV     DPTR,#SORT_COUNT1
        MOVX    A,@DPTR
        MOV     R5,A
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        JZ      VINMGA11
```

```
VINMGA1:
        CALL    ADDR_CALC
        MOVX    A,@DPTR
        MOV     R0,#BUFFER
        INC     R5
        MOV     @R0,A
        DEC     R5
        CALL    FILL_NUM1
        CALL    ADDR_CALC
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R0,#BUFFER
        MOV     @R0,A
        CALL    FILL_NUM
        CALL    ADDR_CALC
        INC     DPTR
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R0,#BUFFER
        MOV     @R0,A
        CALL    FILL_NUM2
        MOV     R0,#CHAR4
        MOV     @R0,#D_CODE


        JNB     FLENT,XTR2

        DEC     R6
        CJNE    R6,#00,XTR3
        CALL    LOAD_SORT_COUNT1
        JMP     XTR2

XTR3:
        DEC     R5
        MOV     A,R5
        MOV     DPTR,#SORT_COUNT1
        MOVX    @DPTR,A

XTR2:
        CLR     FLD0
        CLR     FLENT
        CLR     FLK5

        POP     6
        POP     5

        JMP     X

VINMGA11:
        CALL    FILL_ZERO
        JMP     XTR2

LOAD_SORT_COUNT2:
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,#01,DEE1
        JMP     LOAD_SORT_COUNT1
```

```
DEE1:
        MOVX    A,@DPTR
        JZ      DEE
        DEC     A
        DEC     A
DEE:
        INC     DPTR
        MOVX    @DPTR,A
        RET

LOAD_SORT_COUNT1:
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        JZ      LSC1
        DEC     A
LSC1:
        INC     DPTR
        MOVX    @DPTR,A
        RET

FILL_NUM2:
        PUSH    R3$
        PUSH    R4$
        MOV     R4,#CHAR3
        MOV     R3,#BUFFER
        MOV     A,#0FH
        MOV     R0,R3$
        ANL     A,@R0
        CALL    FIND_CODES
        INC     R4
        MOV     R0,R3$
        MOV     A,@R0
        SWAP    A
        ANL     A,#0FH
        CALL    FIND_CODES
        POP     R4$
        POP     R3$
        RET

FILL_ZERO:
        MOV     R0,#DISP
        MOV     @R0,#Y_CODE
        INC     R0
        MOV     @R0,#T_CODE
        INC     R0
        MOV     @R0,#P_CODE
        INC     R0
        MOV     @R0,#M_CODE
        INC     R0
        MOV     @R0,#E_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        RET
```

```
;-------------------------------------------------------------------
;THIS PROCEDURE REMOVES THE POINTED DATA FROM THE BUFFER AND
;REORDERS THE BUFFER. THE COUNT OF DATA POINTED SHOULD BE PASSED
;THROUGH THE PROCEDURE REGISTER R5
;-------------------------------------------------------------------

START_DEL:
        PUSH    5
        MOV     DPTR,#DEL_DATA
        MOVX    A,@DPTR
        MOV     R5,A
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,R5$,STAR
        MOV     R5,#00

STAR2:
        CALL    LOAD_BUFF_DEL
        CALL    DIVE_BUFF_DEL
        INC     R5
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,R5$,STAR2
        JMP     STAR1

STAR:
        CALL    LOAD_BUFF_DEL
        CALL    DIVE_BUFF_DEL
        INC     R5
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,R5$,STAR

STAR1:
        CALL    DEC_COUNT
        POP     5
        RET

DEC_COUNT:
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        JZ      DC1
        DEC     A

DC1:
        MOVX    @DPTR,A
        RET
LOAD_BUFF_DEL:
        INC     R5
        MOV     R0,#HOU_BUFF
        CALL    ADDR_CHK_HOU
        MOV     @R0,A
        INC     R0
        CALL    ADDR_CHK_MIN
        MOV     @R0,A
        INC     R0
        CALL    ADDR_CHK_DUR
        MOV     @R0,A
        RET
```

```
DIVE_BUFF_DEL:
        DEC     R5
        CALL    ADDR_CALC
        MOV     R0,#HOU_BUFF
        MOV     A,@R0
        MOVX    @DPTR,A
        INC     R0
        INC     DPTR
        MOV     A,@R0
        MOVX    @DPTR,A
        INC     R0
        INC     DPTR
        MOV     A,@R0
        MOVX    @DPTR,A
        RET

;-----------------------------------------------------------------------
;CONTROL FUNCTION FOUR SETS
;-----------------------------------------------------------------------

CON4:
        CALL    L_SORT_C_A
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        CJNE    A,#MAX_MON,D_FULL2
        CALL    D_FULL
        JMP     SUC1Z

D_FULL2:
        CALL    DISP_COUNTX
        JNB     FLK5,X1XX
        JMP     SOR_STARTZ

X1XX:
        JMP     SUBCNT3

LOFUN4:
        MOV     R0,#FUN4
        MOV     A,@R0
        MOV     R4,A
        RET

SUBCNT3:
        CALL    LOFUN4
        MOV     DPTR,#SUBCOOL3
        MOV     B,#03H
        MOV     A,R4
        MUL     AB
        JMP     @A+DPTR

SUBCOOL3:
        LJMP    TUCONX2
        LJMP    TUCONX1
        LJMP    TUCONX0

TUCONX2:
        CLR     FLD0
        JMP     X
```

```
SAFUN4:
        MOV     RO,#FUN4
        MOV     A,R4
        MOV     @RO,A
        RET

TUCONXO:
        MOV     RO,#CHAR3
        MOV     @RO,#N_CODE
        INC     RO
        MOV     @RO,#O_CODE
        INC     RO
        MOV     @RO,#M_CODE
        MOV     DPTR,#BMONTH
        CALL    DOG
        JNB     FLINC,SUC11XZ
        CALL    MON
        CALL    DOG
SUC11XZ:
        JNB     FLENT,SUC1Z
        DJNZ    R4,SUC1XZ
        MOV     R4,#02
SUC1XZ:
        CALL    SAFUN4
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT3
SUC1Z:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

TUCONX1:
        MOV     RO,#CHAR3
        MOV     @RO4,#T_CODE
        INC     RO
        MOV     @RO,#A_CODE
        INC     RO
        MOV     @RO,#D_CODE
        MOV     DPTR,#BDAYX
        CALL    DOG
        JNB     FLINC,SUC21XZ
        CALL    DAY
        CALL    DOG
SUC21XZ:
        JNB     FLENT,SUC2Z
        DJNZ    R4,SUC2XZ
        MOV     R4,#02
SUC2XZ:
        CALL    SAFUN4
        CLR     FLINC
        CLR     FLENT
        JMP     SUBCNT3
SUC2Z:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X
```

```
DISP_COUNTX:
        PUSH    5
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        MOV     R5,A
        CALL    BINTIN
        MOV     R0,#BUFFER
        MOV     @R0,5
        CALL    FILL_NUM1
        POP     5
        RET

;-------------------------------------------------------------
;PROCEDURE TO SORT
;-------------------------------------------------------------
SOR_STARTZ:
        PUSH    3
        PUSH    4
        PUSH    5

        MOV     R0,#HOU_BUFFX
        MOV     DPTR,#BMONTH
        MOVX    A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     DPTR,#BDAYX
        MOVX    A,@DPTR
        MOV     @R0,A

        MOV     DPTR,#SORT_COUNTX   ;previously sorted value
        MOVX    A,@DPTR         ;(count) is loaded into register r5
        MOV     R5,A

        CJNE    R5,#0,CONTINUEY     ;check count for 0 if so write
        CALL    HOU_CHKY            ;the value to the buffer
        MOV     DPTR,#ST_TIME_BUFFX
        MOVX    @DPTR,A
        CALL    MIN_CHKY
        INC     DPTR
        MOVX    @DPTR,A
;       INC     DPTR
;       CALL    DURATION
;       MOVX    @DPTR,A
        CALL    PUT_COUNTY          ;put the inc count value in
        POP     5                   ;the storage space
        POP     4
        POP     3
        JMP     XZ

PUT_COUNTY:
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        INC     A
        MOVX    @DPTR,A
        RET

HOU_CHKY:
        MOV     R0,#HOU_BUFFX
        MOV     A,@R0
        RET
```

```
MIN_CHKY:
        MOV     R0,#MIN_BUFFX
        MOV     A,@R0
        RET

;DURATION:
;       MOV     R0,#DU_BUFF
;       MOV     A,@R0
;       RET

CONTINUEY:
        JMP     CHK_EQUY        ;check in the buffer,for equal value

CONTINUEY2:                     ;if found so discard it
        CLR     C
        DJNZ    R5,HITLERY
        SETB    FLLERX

HITLERY:                ;check for equality of hour (month) part
        CALL    HOU_CHKY
        MOV     R3,A
        CALL    ADDR_CHK_HOUY
        MOV     B,A
        SUBB    A,R3
        JC      MINUTESY1       ;if the carry occurs  place the data
        MOV     A,B             ;in that place
        CJNE    A,R3$,CONTINUEY1
                                ;if they are equal sort min (day)
        CALL    ADDR_CHK_MINY
        MOV     R4,A
        CALL    MIN_CHKY
        CJNE    A,R4$,MINUTESY
        POP     5
        POP     4
        POP     3

        JMP     XZ

MINUTESY1:
        INC     R5
        CALL    ADDR_CALCY
        CALL    HOU_CHKY
        MOVX    @DPTR,A
        INC     DPTR
        CALL    MIN_CHKY
        MOVX    @DPTR,A
;       INC     DPTR
;       CALL    DURATION
;       MOVX    @DPTR,A
        CALL    PUT_COUNTY
        CLR     FLLERX
        POP     5
        POP     4
        POP     3

        JMP     XZ
```

```
MINUTESY2:
          CALL      ADDR_CALCY
          CALL      HOU_CHKY
          MOVX      @DPTR,A

          INC       DPTR
          CALL      MIN_CHKY
          MOVX      @DPTR,A
;         INC       DPTR
;         CALL      DURATION
;         MOVX      @DPTR,A
          CALL      PUT_COUNTY
          CLR       FLLERX
          RET

CONTINUEY1:
          CALL      EXCHANGEZ
          JNB       FLLERX,HITLERY1
          CALL      MINUTESY2
HITLERY1:
          INC       R5
          DJNZ      R5,CONTINUEY2
          POP       5
          POP       4
          POP       3

          JMP       XZ

MINUTESY:
          CLR       C
          CALL      MIN_CHKY
          MOV       R4,A
          CALL      ADDR_CHK_MINY
          SUBB      A,R4
          JC        MINUTESY1
          CALL      EXCHANGEZ
          JNB       FLLERX,HITLERY2
          CALL      MINUTESY2
HITLERY2:
          INC       R5
          DJNZ      R5,HITLERY3
          POP       5
          POP       4
          POP       3

          JMP       XZ

HITLERY3:
          JMP       CONTINUEY2

ADDR_CHK_HOUY:
          CALL      ADDR_CALCY
          MOVX      A,@DPTR
          RET

ADDR_CHK_MINY:
          CALL      ADDR_CALCY
          INC       DPTR
          MOVX      A,@DPTR
          RET
```

```
;ADDR_CHK_DUR:
;         CALL    ADDR_CALCY
;         INC     DPTR
;         INC     DPTR
;         MOVX    A,@DPTR
;         RET
EXCHANGEZ:
          MOV     R0,#TEMP_HOUX
          CALL    ADDR_CHK_HOUY
          MOV     @R0,A
          INC     R0
          CALL    ADDR_CHK_MINY
          MOV     @R0,A
;         INC     R0
;         CALL    ADDR_CHK_DUR
;         MOV     @R0,A
          INC     R5
          CALL    ADDR_CALCY
          MOV     R0,#TEMP_HOUX
          MOV     A,@R0
          MOVX    @DPTR,A
          INC     R0
          INC     DPTR
          MOV     A,@R0
          MOVX    @DPTR,A
;         INC     R0
;         INC     DPTR
;         MOV     A,@R0
;         MOVX    @DPTR,A
          DEC     R5
          RET
ADDR_CALCY:
          CLR     C
          MOV     DPTR,#ST_TIME_BUFFX
          MOV     B,#02
          MOV     A,R5
          MUL     AB
          ADD     A,DPL
          JC      ADDR_CALCY1
          MOV     DPL,A
          RET
ADDR_CALCY1:
          MOV     DPL,A
          INC     DPH
          RET
CHK_EQUY:
          CALL    HOU_CHKY
          MOV     R3,A
          CALL    MIN_CHKY
          MOV     R4,A
CHK_EQUY1:
          DEC     R5
          CALL    ADDR_CHK_HOUY
          CJNE    A,R3$,ZORRORZ
          CALL    ADDR_CHK_MINY
          CJNE    A,R4$,ZORRORZ
          POP     5
          POP     4
          POP     3
          JMP     XZ
```

```
ZORRORZ:
        INC     R5
        DJNZ    R5,CHK_EQUY1
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        MOV     R5,A
        JMP     CONTINUEY2

XZ:
        CLR     FLD0
        CLR     FLK5
        CALL    DISP_COUNTX

        JMP     X
;-------------------------------------
;PROCEDURE DELETES TWO BYTES BUFFER
;-------------------------------------

CON5:
        PUSH    5
        PUSH    6

        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        CJNE    A,#00,STARXX
        CALL    FILL_ZERO
        JMP     XTR2X

STARXX:
        MOV     DPTR,#SORT_COUNTX1
        MOVX    A,@DPTR
        MOV     R5,A
        INC     A
        MOV     R6,A
        JZ      VINMGAX
VINMGAX:
        JNB     FLK5,VINMGA1X

        MOV     DPTR,#DEL_DATAX
        INC     R5
        MOV     A,R5
        MOVX    @DPTR,A
        DEC     R5
        CALL    START_DELX
        CALL    LOAD_SORT_COUNTX2
        MOV     DPTR,#SORT_COUNTX1
        MOVX    A,@DPTR
        MOV     R5,A
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        JZ      VINMGA11X
VINMGA1X:
        CALL    ADDR_CALCY
        MOVX    A,@DPTR
        MOV     R0,#BUFFER
        INC     R5
        MOV     @R0,A
        DEC     R5
        CALL    FILL_NUM1
        CALL    ADDR_CALCY
```

```
            INC     DPTR
            MOVX    A,@DPTR
            MOV     R0,#BUFFER
            MOV     @R0,A
            CALL    FILL_NUM
            MOV     R0,#CHAR3
            MOV     @R0,#L_CODE
            INC     R0
            MOV     @R0,#E_CODE
            INC     R0
            MOV     @R0,#D_CODE

            JNB     FLENT,XTR2X

            DEC     R6
            CJNE    R6,#00,XTR3X
            CALL    LOAD_SORT_COUNTX1
            JMP     XTR2
XTR3X:
            DEC     R5
            MOV     A,R5
            MOV     DPTR,#SORT_COUNTX1
            MOVX    @DPTR,A

XTR2X:
            CLR     FLD0
            CLR     FLENT
            CLR     FLK5

            POP     6
            POP     5

            JMP     X

VINMGA11X:
            CALL    FILL_ZERO
            JMP     XTR2X

LOAD_SORT_COUNTX2:
            MOV     DPTR,#SORT_COUNTX
            MOVX    A,@DPTR
            CJNE    A,#01,DEE1X
            JMP     LOAD_SORT_COUNTX1
DEE1X:
            MOVX    A,@DPTR
            JZ      DEEX
            DEC     A
            DEC     A
DEEX:
            INC     DPTR
            MOVX    @DPTR,A
            RET

LOAD_SORT_COUNTX1:
            MOV     DPTR,#SORT_COUNTX
            MOVX    A,@DPTR
            JZ      LSC1X
            DEC     A
```

```
LSC1X:
        INC     DPTR
        MOVX    @DPTR,A
        RET

;-------------------------------------------------------------------
;THIS PROCEDURE REMOVES THE POINTED DATA FROM THE BUFFER AND
;REORDERS THE BUFFER. THE COUNT OF DATA POINTED SHOULD BE PASSED
;THROUGH THE PROCEDURE REGISTER R5
;-------------------------------------------------------------------

START_DELX:
        PUSH    5
        MOV     DPTR,#DEL_DATAX
        MOVX    A,@DPTR
        MOV     R5,A
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        CJNE    A,R5$,STARX11
        MOV     R5,#00
STARX2:
        CALL    LOAD_BUFF_DELX
        CALL    DIVE_BUFF_DELX
        INC     R5
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        CJNE    A,R5$,STARX2
        JMP     STARX1
STARX11:
        CALL    LOAD_BUFF_DELX
        CALL    DIVE_BUFF_DELX
        INC     R5
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        CJNE    A,R5$,STARX11

STARX1:
        CALL    DEC_COUNTX
        POP     5
        RET

DEC_COUNTX:
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        JZ      DC1X
        DEC     A
DC1X:
        MOVX    @DPTR,A
        RET

LOAD_BUFF_DELX:
        INC     R5
        MOV     R0,#HOU_BUFFX
        CALL    ADDR_CHK_HOU
        MOV     @R0,A
        INC     R0
        CALL    ADDR_CHK_MIN
        MOV     @R0,A
        RET
```

```
DIVE_BUFF_DELX:
        DEC     R5
        CALL    ADDR_CALCY
        MOV     RO,#HOU_BUFFX
        MOV     A,@RO
        MOVX    @DPTR,A
        INC     RO
        INC     DPTR
        MOV     A,@RO
        MOVX    @DPTR,A
        RET

;-------------------------------------------------------------------------
;PROCEDURE CLEARS THE BUFFER
;PROGRAM ASKS FOR THE PASS-WORD & IF FOUND EQUAL CLEARS THE RAM
;-------------------------------------------------------------------------
CON6:
        MOV     DPTR,#SORT_COUNT
        MOVX    A,@DPTR
        CJNE    A,#00,ONE
        MOV     DPTR,#SORT_COUNTX
        MOVX    A,@DPTR
        CJNE    A,#00,ONE
        CALL    CC1
        CLR     FLDO
        CLR     FLENT
        CLR     FLINC

        JMP     X

ONE:
        MOV     RO,#CHAR2
        MOV     @RO,#BLANK_CODE
        INC     RO
        MOV     @RO,#S_CODE
        INC     RO
        MOV     @RO,#S_CODE
        INC     RO
        MOV     @RO,#A_CODE
        INC     RO,
        MOV     @RO,#P_CODE
        MOV     DPTR,#BYEARX
        CALL    DOG
        JNB     FLINC,SUCO1Z
        CALL    YEAR
        CALL    DOG
SUCO1Z:
        JNB     FLENT,SUCOZ
        CALL    CLEAR_CODE
SUCOZ:
        CLR     FLDO
        CLR     FLINC
        CLR     FLENT
        JMP     X

CLEAR_CODE:
        MOV     DPTR,#BYEARX
        MOVX    A,@DPTR
        CJNE    A,#99H,CC2
        JMP     CC1
```

```
CC2:
        MOV     R0,#CHAR2
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        MOV     R0,#CHAR3
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#E_CODE
        RET

CC1:
        MOV     R0,#DISP
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        MOV     R0,#CHAR2
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#A_CODE
        INC     R0
        MOV     @R0,#E_CODE
        INC     R0
        MOV     @R0,#L_CODE
        INC     R0
        MOV     @R0,#C_CODE
        MOV     DPTR,#SORT_COUNT
        MOV     A,#00
        MOVX    @DPTR,A
        MOV     DPTR,#SORT_COUNTX
        MOVX    @DPTR,A
        MOV     DPTR,#STORE_WEEK
        MOVX    @DPTR,A
        RET

;-------------------------------------------------------------
;PROCEDURE KNOCKS OFF THE DAY
;-------------------------------------------------------------

CON7:
        PUSH    5
        CALL    LOAD_WE

CON7X:
        CALL    LOAD_FUN5

        JNB     FLENT,FOLLOW
        SETB    FLWDS

FOLLOW:
        CALL    WE_DA_SET

        JNB     FLINC,FOL1
        INC     R5
        CJNE    R5,#07,FOL2
        MOV     R5,#00
```

```
FOL2:
          CALL      SAVE_FUN5
          CLR       FLENT
          CLR       FLINC
          JMP       CON7X

FOL1:
          CLR       FLENT
          CLR       FLINC
          CLR       FLDO
          CALL      SAVE_WE
          POP       5

          JMP       X

LOAD_WE:
          MOV       DPTR,#STORE_WEEK
          MOVX      A,@DPTR
          MOV       R0,#FLSET_WE_DAY
          MOV       @R0,A
          RET

SAVE_WE:
          MOV       DPTR,#STORE_WEEK
          MOV       R0,#FLSET_WE_DAY
          MOV       A,@R0
          MOVX      @DPTR,A
          RET

LOAD_FUN5:
          MOV       R0,#FUN5
          MOV       A,@R0
          MOV       R5,A
          RET

SAVE_FUN5:
          MOV       R0,#FUN5
          MOV       A,R5
          MOV       @R0,A
          RET

WE_DA_SET:
          MOV       DPTR,#WEEKJMPX
          MOV       A,R5
          MOV       B,#03
          MUL       AB
          JMP       @A+DPTR

WEEKJMPX:
          LJMP      WEEK0X
          LJMP      WEEK1X
          LJMP      WEEK2X
          LJMP      WEEK3X
          LJMP      WEEK4X
          LJMP      WEEK5X
          LJMP      WEEK6X
```

```
WEEK0X:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#S_CODE

        JNB     FLWDS,WX01
        CPL     29H.0

WX01:
        JNB     29H.0,WX0
        CALL    SET_DAY
        CLR     FLWDS
        RET

WX0:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

WEEK1X:
        MOV     R0,#CHAR3
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#O_CODE
        INC     R0
        MOV     @R0,#M_CODE

        JNB     FLWDS,WX11
        CPL     29H.1

WX11:
        JNB     29H.1,WX1
        CALL    SET_DAY
        CLR     FLWDS
        RET
WX1:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

WEEK2X:
        MOV     R0,#CHAR3
        MOV     @R0,#E_CODE
        INC     R0
        MOV     @R0,#U_CODE
        INC     R0
        MOV     @R0,#T_CODE

        JNB     FLWDS,WX21
        CPL     29H.2

WX21:
        JNB     29H.2,WX2
        CALL    SET_DAY
        CLR     FLWDS
        RET
```

```
WX2:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

WEEK3X:
        MOV     R0,#CHAR3
        MOV     @R0,#D_CODE
        INC     R0
        MOV     @R0,#E_CODE
        INC     R0
        MOV     @R0,#W_CODE

        JNB     FLWDS,WX31
        CPL     29H.3

WX31:
        JNB     29H.3,WX3
        CALL    SET_DAY
        CLR     FLWDS
        RET

WX3:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

WEEK4X:
        MOV     R0,#CHAR3
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#H_CODE
        INC     R0
        MOV     @R0,#T_CODE

        JNB     FLWDS,WX41
        CPL     29H.4

WX41:
        JNB     29H.4,WX4
        CALL    SET_DAY
        CLR     FLWDS
        RET

WX4:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

WEEK5X:
        MOV     R0,#CHAR3
        MOV     @R0,#I_CODE
        INC     R0
        MOV     @R0,#R_CODE
        INC     R0
        MOV     @R0,#F_CODE

        JNB     FLWDS,WX51
        CPL     29H.5
```

```
WX51:
        JNB     29H.5,WX5
        CALL    SET_DAY
        CLR     FLWDS
        RET
WX5:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

WEEK6X:
        MOV     R0,#CHAR3
        MOV     @R0,#T_CODE
        INC     R0
        MOV     @R0,#A_CODE
        INC     R0
        MOV     @R0,#S_CODE

        JNB     FLWDS,WX61
        CPL     29H.6

WX61:
        JNB     29H.6,WX6
        CALL    SET_DAY
        CLR     FLWDS
        RET
WX6:
        CALL    RESET_DAY
        CLR     FLWDS
        RET

SET_DAY:
        MOV     R0,#DISP
        MOV     @R0,#N_CODE
        INC     R0
        MOV     @R0,#O_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        RET

RESET_DAY:
        MOV     R0,#DISP
        MOV     @R0,#F_CODE
        INC     R0
        MOV     @R0,#F_CODE
        INC     R0
        MOV     @R0,#O_CODE
        INC     R0
        MOV     @R0,#BLANK_CODE
        RET

DISPL_TBL:
;**** NUMERIC DISPLAY CODES ****
;****     h g f e d c b a     ****
        DB      03FH    ;0
        DB      006H    ;1
        DB      05BH    ;2
```

```
        DB      04FH    ;3
        DB      066H    ;4
        DB      06DH    ;5
        DB      07DH    ;6
        DB      007H    ;7
        DB      07FH    ;8
        DB      06FH    ;9
        DB      077H    ;A
        DB      07CH    ;B
        DB      039H    ;C
        DB      05EH    ;D
        DB      079H    ;E
        DB      071H    ;F

;** ALPHABET DISPLAY CODES **

ZERO_CODE       EQU     03FH
EQUAL_CODE      EQU     048H
ONE_CODE        EQU     006H
TWO_CODE        EQU     05BH
THREE_CODE      EQU     04FH
SIX_CODE        EQU     07DH
A_CODE          EQU     077H
B_CODE          EQU     07CH
C_CODE          EQU     039H
C1_CODE         EQU     058H
D_CODE          EQU     05EH
E_CODE          EQU     079H
F_CODE          EQU     071H
G_CODE          EQU     03DH
H_CODE          EQU     076H
I_CODE          EQU     030H
J_CODE          EQU     00EH
K_CODE          EQU     070H
L_CODE          EQU     038H
M_CODE          EQU     055H
N_CODE          EQU     054H
O_CODE          EQU     05CH
P_CODE          EQU     073H
R_CODE          EQU     050H
S_CODE          EQU     06DH
T_CODE          EQU     078H
U_CODE          EQU     01CH
W_CODE          EQU     01DH
Y_CODE          EQU     06EH
Z_CODE          EQU     05BH
BLANK_CODE      EQU     00OH
DASH_CODE       EQU     040H
DASH1_CODE      EQU     008H
STAR_CODE       EQU     07BH

        END
```

Conclusion...

CHAPTER VII

CONCLUSION

In this project a programmable timer has been designed, fabricated and tested successfully. A microcontroller has been used for controlling the programmable timer.

The software developed enables the unit to operate in the following 8 modes.

1. Time setting mode
2. Alarm setting mode
3. Alarm deleting mode
4. Holiday setting mode
5. Holiday deleting mode
6. Reset mode
7. Day setting mode
8. Display mode

The special feature of the project is that only **4 keys** are used for all operations.

In view of the multiple facilities offered by the timer it has a lot of applications in industries where time controlled operations are performed. Using this timer control can be done automatically.

It can be used effectively to ring bell at the end of the period in educational institutions automatically.

REFERENCES

1. RAMESH S. GOANKAR , "Microprocessor Architecture and Programming and Applications with 8085 and 8080, Wiley Eastern limited,New Delhi, 1991.

2. JOHN B. PEATMAN , "Design with Microcontroller" MCGRAW Hill International Edition, 1988

3. DOUGLAS. V. HALL , "Microprocessors and interfacing programming and Hardware", MCGRAW Hill International Limited, New York, 1986.

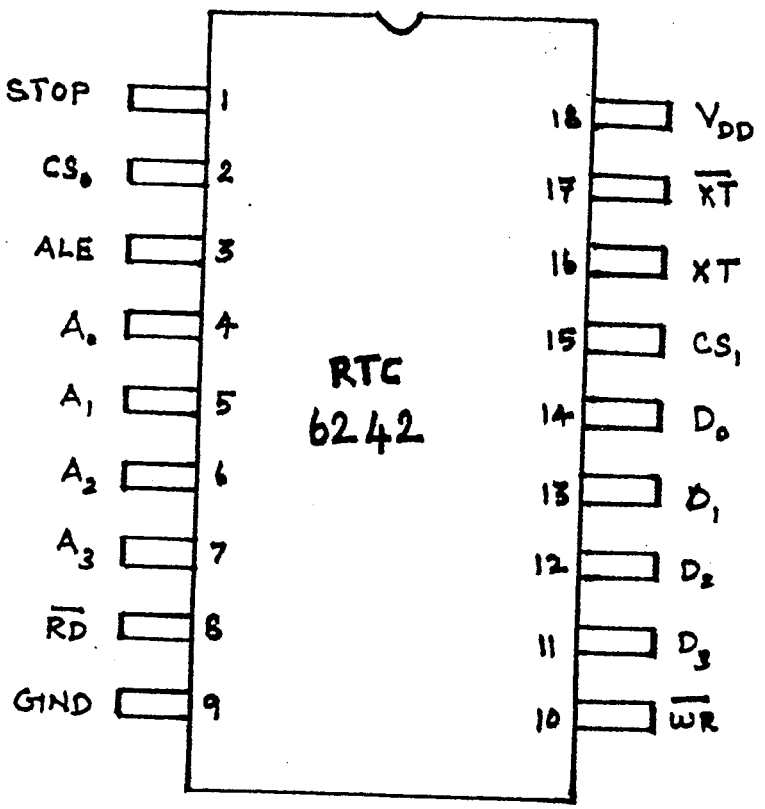
4. "Micro processor data Hand Book", BPB publications, New Delhi, 1992.

5. "TTL Logic Data book", Texas Instruments, U.S.A, March 1988.

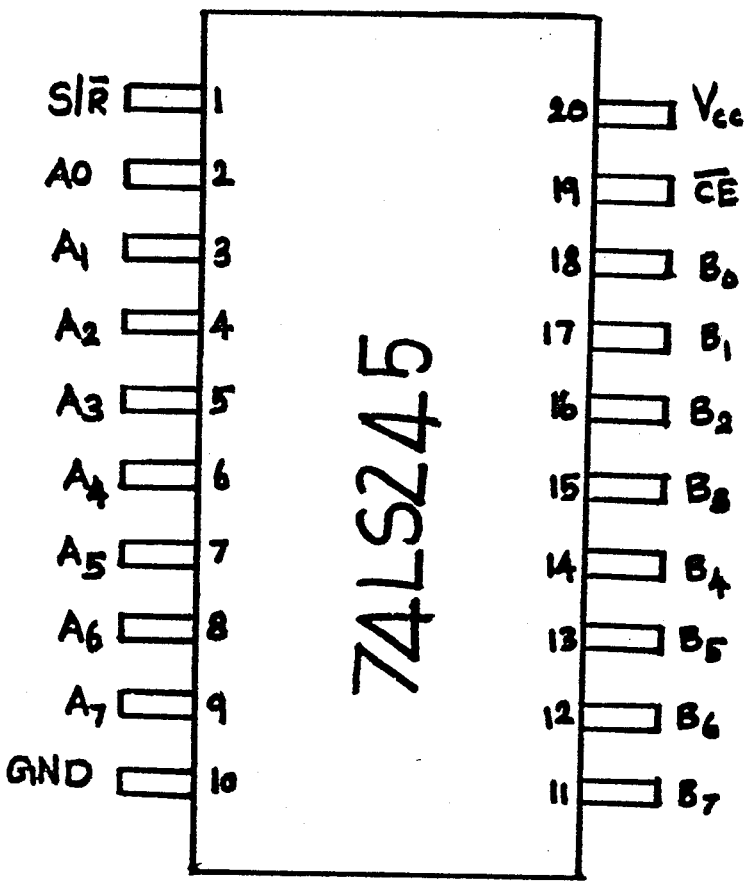
6. "Master selection Guide", Texas instruments, U.S.A, 1988.

7. "Embedded Microcontrollers and Processors", Vol I, Intel 1993.

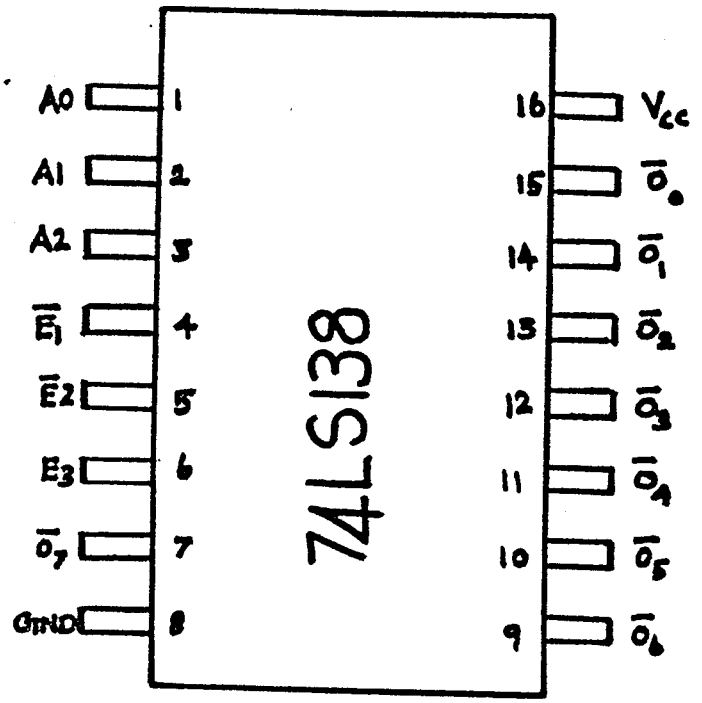
APPENDIX



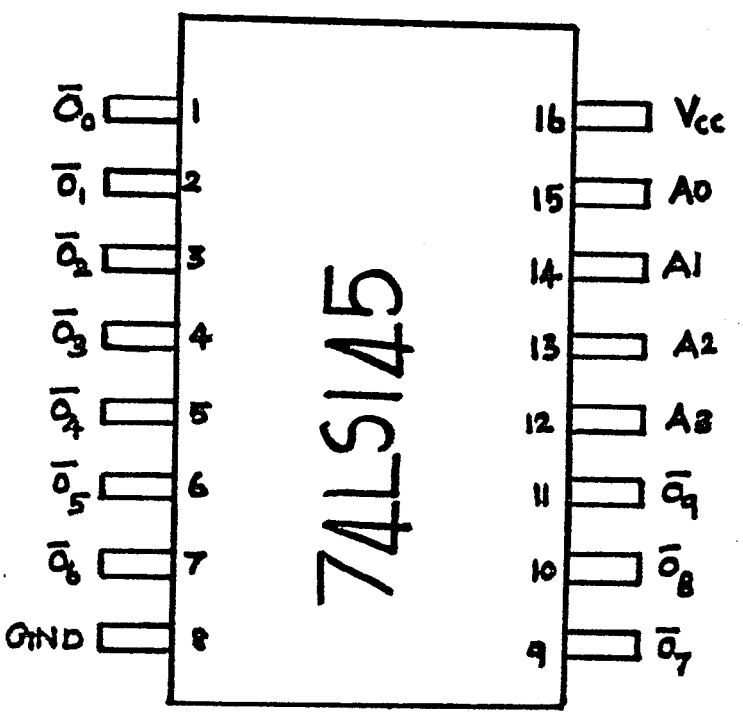
REAL TIMECLOCK CHIP



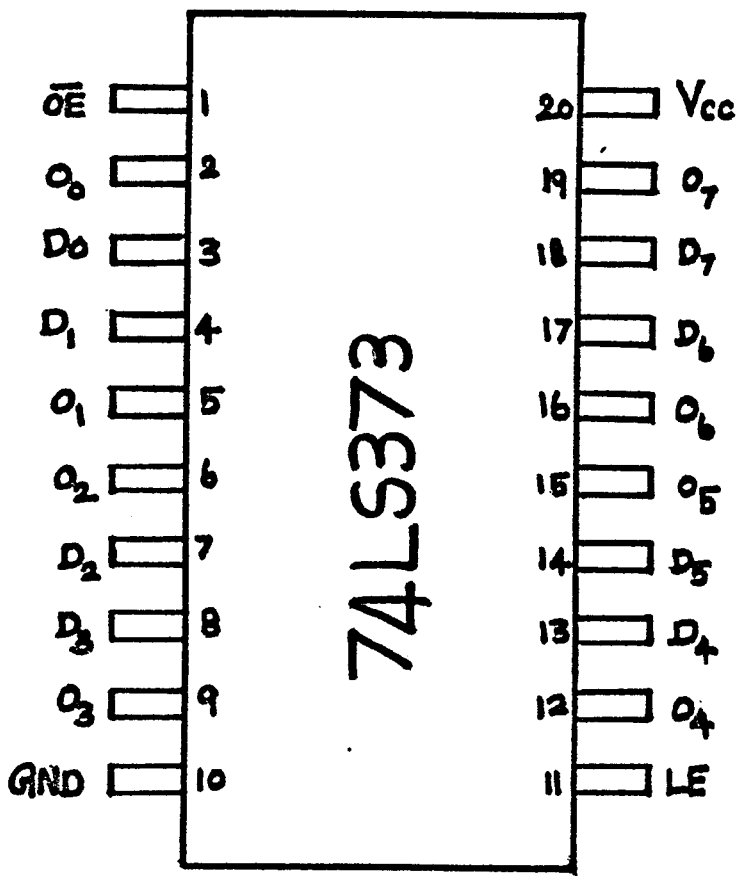
OCTAL TRANSCEIVER



3X8 DECODER



1 OF 10 DECODER



OCTAL LATCH/DEMULTIPLEXER