# THE DEVELOPMENT OF AN INFRA-RED MONITORING AND DATA-LOGGING SYSTEM

By Juan Du Preez

Thesis submitted in partial fulfillment of the requirements for the Master Diploma in Technology to the Department of Electrical Engineering (light current) at the Cape Technikon.

CAPE TOWN
SOUTH AFRICA
MAY 1992

**DECLARATION**

I declare that the contents of this thesis represent my own work and the opinions contained here are my own. It has not been submitted before for any examination at this or any other institute.

**J Du Preez**

**(Signature)**

## TERMS OF REFERENCE

This thesis project was commissioned by Ms N. Louw, a Senior lecturer in the Department of Nature and Science Research Institute at the Cape Technikon, on 20 July 1990. The project is required for partial fulfillment of the requirements for the Masters Diploma in Technology.

Ms. Louw's specific instructions were:

1. To design an automatic monitoring system for pedestrian traffic (hikers).
2. To decide on a suitable design strategy by performing a thorough literature survey.
3. To make the system accurate and invisible to hikers.
4. To design a low current consumption unit.
5. To make the system low cost.
6. To prepare a detailed thesis report on the above system.

## ABSTRACT

This thesis describes the design and development of an infra-red monitoring and data logging system. The infra-red unit will be used to monitor certain hiking trails on Table Mountain, whilst the logging unit will be used to log the date and time of the monitored person into memory. This logged data can then be used to compute the statistics relating to that specific hiking trail.



## OPSOMMING

Hierdie verhaling beskryf die ontwerp van 'n infra-rooi monitor en data-opnemer stelsel. Die infra-rooi eenheid sal gebruik word om sekere wandel paaie op Tafelberg te monitor, terwyl die opnemer eenheid die datum en tyd van gemoniteerde persone stoor. Hierdie gestoorde inligting kan dan gebruik word om die statistieke van die wandel paaie te bereken.

## ACKNOWLEDGMENTS

I would especially like to thank Mr. J. P. Calitz for his friendly advice in helping with the software development.

I would like to thank my colleagues in the Cape Town Research and Development Centre of Telkom SA, especially Mr. M. Amerika, with whom many helpful discussions were held.

## TABLE OF CONTENTS

TABLE CONTENTS (continued) Page

**TABLE CONTENTS (continued)** **Page**

## LIST OF ILLUSTRATIONS

# 1. INTRODUCTION

This thesis involves an investigation into the design and implementation of an automatic infra-red detection system for hiking trails.

The vegetation on the Table Mountain is being destroyed by excessive hiking, causing severe soil erosion. To prevent escalation of this problem, the NATURE and SCIENCE RESEARCH INSTITUTE then commissioned this research project with the following **major objectives:**

(i) to design an automatic monitoring system for the traffic carrying capacity of these trails.
(ii) to determine where additional routes are required in order to provide an even traffic flow.

Most of the relevant information for the system designed was obtained from technical literature (data books, technical manuals, etc.) and at the Research and Development Centre (Telkom S.A.) in Cape Town under the supervision of Mr Tony Herbert.

Firstly, to fulfil the above requirements, the following systems were investigated:

(i) **Passive Infra-Red System:** This is a single-ended type system (receiver only) which is based on passive infra-red detection

techniques. In other words it reacts to the infra-red energy radiated by any living object that moves into the monitored area.

(ii) **Ultrasonic System:** This system comprises of a transmitter and a receiver and is based on frequency detection techniques. This system reacts to any moving object that will cause a change in the transmitted frequency.

(iii) **Infra-Red System:** This system comprises of an infra-red transmitter, infra-red receiver and a control unit. The unit is triggered by physically interrupting the infra-red beam.

Based upon the above survey, the system was designed and constructed by using the third strategy, i.e. an infra-red detection system. Infra-red sensors are therefore used in the detection circuitry. This circuit is interfaced to an 8051-based processor board for data (time and date) recording purposes. The results were then analyzed by using software programs written in high, and low level computer languages (Borland Turbo Pascal V5.0 package and 8051 assembler).

The **objectives** of this thesis report are therefore:

(i) to provide background information of the various detection systems that were considered.

(ii) to provide a detailed outline and analysis of the proposed

system.

The report commences with chapter 2 by providing background information about the various systems considered in the literature survey. This chapter also provides a basic description of the actual proposed system. A detailed description of the complete detection circuit, is fully outlined in chapters 3, 4 and 5. Chapters 6, 7, 8 and 9 describe the detailed designs of the digital interfacing (8051 controller, software, etc.) for data logging. The analysis of the data is fully described in chapter 11 (i.e. the software programs). Chapters 12 and 13 outline the problems that were encountered during the implementation of the circuit, as well as test results.

## 2. THE SYSTEMS INVESTIGATED

In this chapter the advantages and disadvantages of the systems, as described in the preceding introduction, will be outlined. These were obtained through a literature survey. A more detailed description of the proposed design system (infra-red system) is also provided.

### 2.1 Advantages and Disadvantages of the Systems Investigated

(i) **Passive Infra-Red System:** The advantage of this type of system is that only a single unit is required and the difficult alignment procedures associated with active systems are not required. However, for this particular application there are disadvantages: i.e. If a number of people are moving through the monitored area the system will not be able to determine exactly how many people passed through. The system will also detect the movement of animals in the surrounding area and this will result in inaccurate data being recorded. Hence, this type of system will not be appropriate.

(ii) **Ultrasonic System:** This system has the major disadvantage of being triggered easily by any change in transmitted frequency and is therefore mostly used in indoor electronic systems. Hence for our application

this system is not appropriate.

(iii) **Infra-Red System:** This system turned out to be the most feasible and efficient method for our application due to the fact that infra-red transmitters and receivers are more accurate, compact and reliable. External factors like different weather conditions and false triggering due to 'small' animals will not have an influence (small refers to animal height i.e. the sensors can be mounted at a certain height to avoid animals).

## 2.2 Proposed Design System

In this section, a basic description of the proposed design system is outlined. Fig 2.1 shows the basic arrangement of the final Infra-Red Monitoring and logging system.

The transmitter consists of an audio oscillator, driving four infra-red LEDs via a buffer amplifier which provides the high peak driving current required by these LEDs. The LEDs pulsing at this high operating current, provide bursts of infra-red radiation. In this way a strong peak output can be obtained from the transmitter. The LEDs used, are of an inexpensive and readily available type. The current consumed by each LED can be provided by a 6 to 9 Volt DC power supply. Infra-red LEDs produce no output in the visible light region.

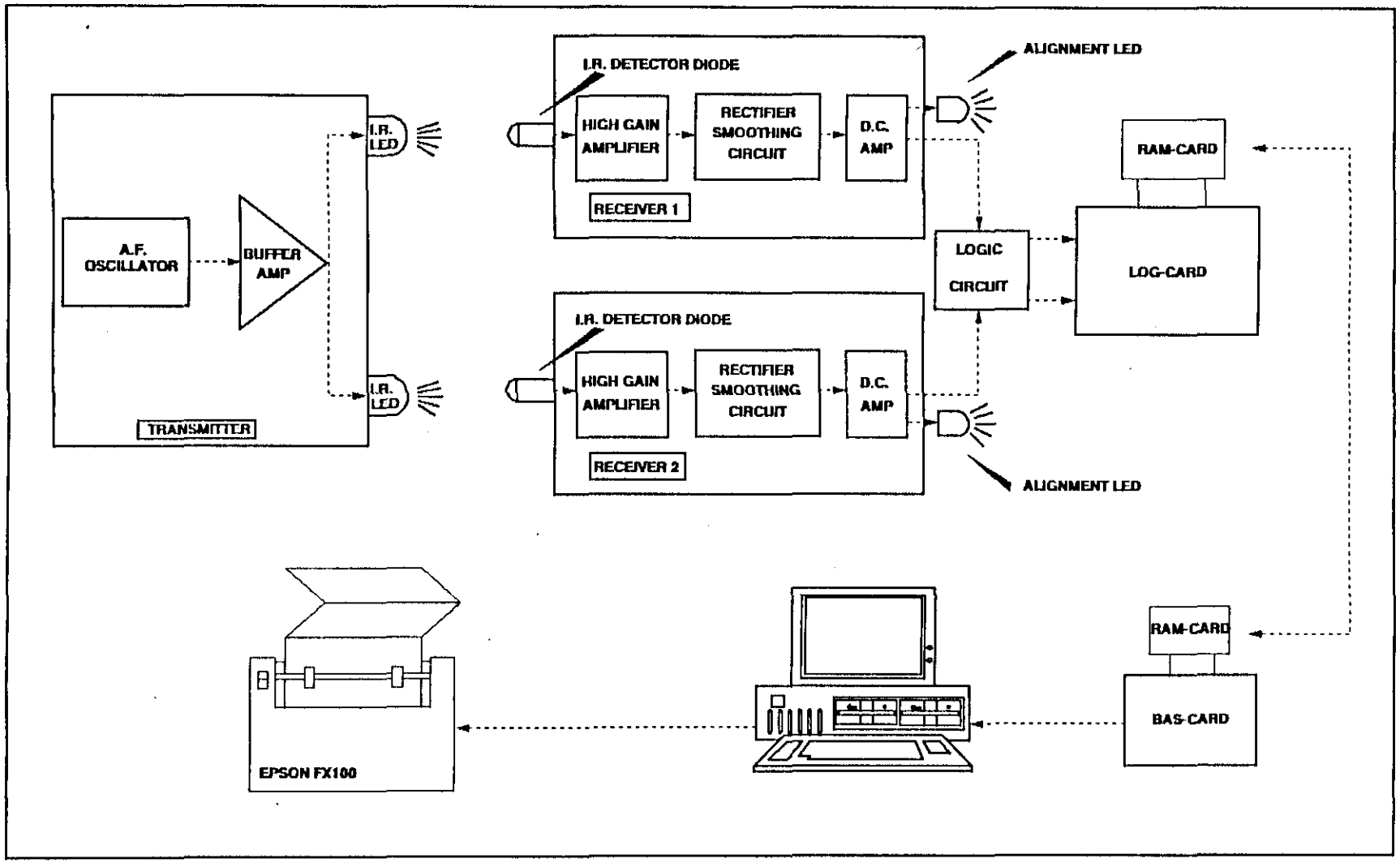



*FIGURE 2.1*

*BLOCK DIAGRAM SHOWING THE BASIC ARRANGEMENT*

*OF THE INFRA-RED MONITORING AND DATA-LOGGING SYSTEM*

The reasons for using a modulated beam rather than a continuous beam are:

(i) The operating current to drive the infra-red LEDs has to be high enough to obtain a strong peak output from the transmitter. The high operating current can damage the infra-red LEDs if it is continuous. Therefore a continuous beam system would need to have very low power output if it were to be constructed from inexpensive and readily available parts. This low power requirement would result in very little change in the infra-red detector diode when the beam is broken and restored again, making it difficult to obtain sensitivity, as well as good immunity to changes in the ambient infra-red level.

(ii) By using a modulated beam, the weak audio signal produced by the infra-red detector diode, as it responds to the infra-red pulses, can be considerably boosted by a high gain preamplifier. The output of the preamplifier is then rectified and smoothed to produce a d.c. signal that is roughly proportional to the strength of the received signal. When the beam is not interrupted, the received signal is strong and a strong d.c. output is produced by this part of the circuit. This signal is then used to drive a d.c. amplifier which in turn drives a monostable multivibrator. This makes the system immune

to most ambient infra-red radiation and is only affected by the modulated infra-red source.

In practice, apart from the transmitter the only source of modulated infra-red radiation is likely to be mains lighting, but provision is made for systems that are installed near mains lighting. Since the mains frequency is only 50 Hertz, the preamplifiers in the receiver circuits are designed to have a very poor response at this low frequency. The transmitter on the other hand is given a fairly high operating frequency, so that the preamplifiers in the receiver circuits exhibit their full gain.

In the idle state, both outputs (not-Q) of the Retriggerable Monostable Multivibrators are high. When any one of the two infra-red beams are broken, the outputs go low. This negative pulse at the output is used as an interrupt to the Log-card's microprocessor which then logs the "date-time" of the monitored person, into a RAM-card. The RAM-card is used to transfer the logged data to a PC via the Bas-card when required. The data provides statistics of the trails, which can then be displayed on the PC monitor or printed as required.

## 3. TRANSMITTER

Refer to drawing No. A1 Appendix A for the complete schematic diagram of the transmitter.

### 3.1 Hardware Design

The transmitter is comprised of an A.F. Oscillator, which consists of a low power CMOS 555 timer with its timing components and a Buffer Amplifier consisting of a VMOS transistor with a resistor to limit the peak current fed to the infra-red LEDs.

### 3.2 Theory of Operation

The timer U1 is used in the astable (free running) mode. Fig 3.1 shows the waveforms generated in this mode of operation. The circuit operates by first charging the timing capacitor C2 to two-thirds of the supply voltage VCC via resistor R2 and the bypassing diode D1. During this time the output, pin 3 of the timer, goes high. Capacitor C2 is then discharged by way of resistor R1 and an internal switching transistor of the timer until the charge falls to one-third of the supply voltage VCC. The output of the timer goes low during the discharge period. In this manner capacitor C2 will be continuously charged and discharged and will thus cause a

series of brief pulses on the output pin of the timer. A mark space ratio of approximately 1 in 10 is obtained. The specified values of these timing components results in an operating frequency of approximately 3 kHz. For calculating the frequency the following formula is used:

The charge time (output high) is given by :

$$t1 = 0,693\ (R2 \times C2)$$

And the discharge time (output low) by :

$$t2 = 0,693\ (R1 \times C2)$$

Thus the total period is :

$$T = t1 + t2 = 0,693\ (R1 + R2) \times C2$$

$$F = \frac{1}{T}$$

where F = required frequency.

Fig 3.2 may be used for quick determination of these RC values. This frequency is low enough to obtain a good efficiency from the infra-red detector diodes and the high gain required by the receiver circuits but is also high enough above 50 Hz not to be rejected from the receiver circuits.

When the output of the timer goes high, it will forward bias the VMOS transistor Q1 which then switches on the infra-

red LEDs D1-D4 via current limiting resistor R3. According to the calculated mark space ratio of 1 in 10, the output of the timer results in the infra-red LEDs being switched off for more than 50% of the time, reducing the current consumption of the circuit. A further reduction is obtained by the low current consumed by the oscillator circuit and the fact that Q1 requires no significant drive current.

The current drawn from the battery supply is therefore little more than the average LED current. This low level of power consumption makes battery operation a practical proposition for this project and it can be left operating for long periods of time.

TOP TRACE: OUTPUT 5V/DIV

BOTTOM TRACE: CAPACITOR VOLTAGE 1V/DIV

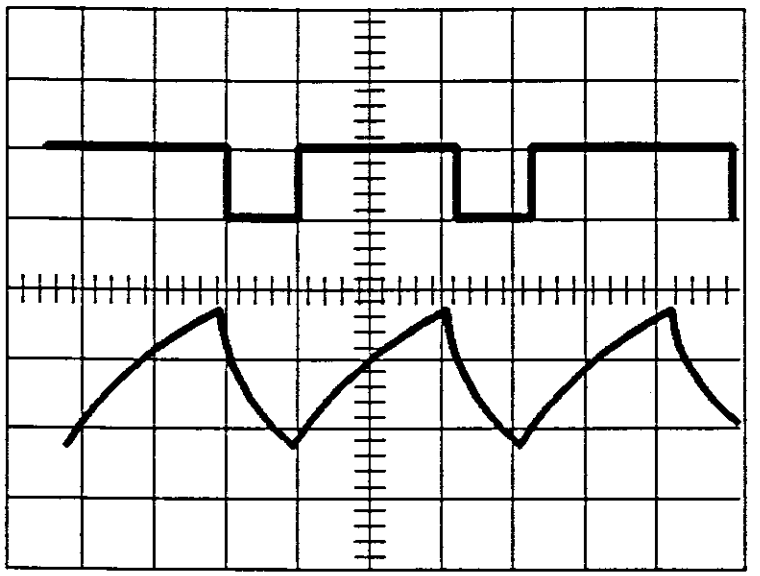


*FIGURE 3.1 ASTABLE WAVEFORMS*

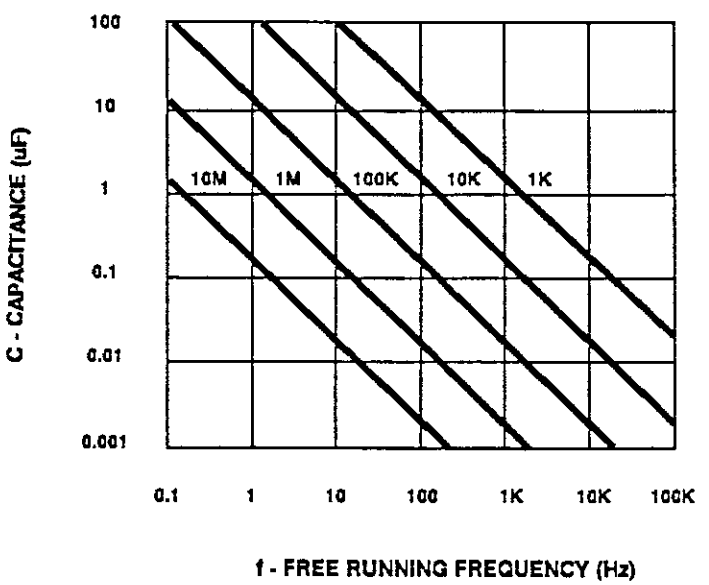



*FIGURE 3.2*
*FREE RUNNING FREQUENCY*

## 4. RECEIVER

Refer to drawing No. A2 Appendix A for the complete schematic diagram of the two identical receivers and logic circuit. The receivers are somewhat more complicated than the transmitter, as can be seen from the circuit diagrams.

### 4.1 Hardware Design

As seen from Fig 2.1 each receiver consists of the following stages:

(i) A detector stage, consisting of an infra-red detector diode. These diodes are high-speed PIN photo-diodes with high photosensitivity housed in a black infra-red transmissive moulding which reduces ambient white light interference.

(ii) An infra-red 4-stage differential preamplifier (SL486). The SL486 is a high gain preamplifier designed to form an interface between an infra-red receiving diode and the digital input of the receiving circuit.

(iii) A smoothing and rectifier stage.

(iv) A d.c. amplifier stage.

(v) A logic stage (Dual Retriggerable Monostable Multivibrator).

## 4.2 Theory of Operation

Seeing that the receivers are identical, the operation of only one will be discussed.

### 4.2.1 Detector Stage

The infra-red receiving diode is connected between pins 1 and 16 of the SL486. The input circuit is configured so as to reject signals common to both pins. This improves the stability of the device and greatly reduces the sensitivity to radiated electrical noise. The diode is reverse biased by a nominal 0.65 V.

### 4.2.2 Preamplifier Stage

The decoupling capacitor C2, rolls off the gain of the feedback loop, which balances the d.c. component of the infra-red diode current. The value of C2 is chosen to produce a low frequency cut-off characteristic below 2 kHz. Hence, C2 produces approximately 20 dB rejection at 100 Hz. The decoupling capacitor C2, must be connected between pin 2 and pin 4 of the SL486.

Capacitor C1 connected between pin 15 and earth decouples the signal from the non-inverting input of the first difference amplifier of the SL486. The capacitance of 15 nF is chosen to produce a 2 kHz low frequency roll-off.

Capacitor C3 connected between pin 5 and pin 4 decouples the signal from the non-inverting input of the second difference amplifier of the SL486. The capacitance of 33 nF is chosen to produce a 2 kHz low frequency roll-off.

Capacitor C4 connected between pin 6 and VCC decouples the signal from the non-inverting input of the fourth difference amplifier of the SL486. The capacitance of 4.7 nF is chosen to produce a 2 kHz low frequency roll-off.

Decoupling capacitor C7 connected between pin 8 and ground filters the pulsed input of the SL486 and the resultant level controls the gain of the first three difference amplifiers. The control level exhibits a fast-attack/slow-decay characteristic. When the infra-red pulses are detected, the gain is reduced so that any weaker noise pulses that are also received will not be seen at the output.

Capacitor C6 is connected between VCC and ground for decoupling purposes.

The output pin 9 of the SL486 will be low, pulsing high with a source impedance of a nominal 55 kΩ, for a received infra-red pulse. It is an amplification of the input and swings between output ground and output VCC.

### 4.2.3 Smoothing and Rectifier Stage

The output, pin 9 of the SL486 is connected to the d.c. amplifier stage via the smoothing and rectifier stage. A series of positive pulses are produced at the output of the rectifier and these are used to bias transistor Q1 into conduction.

### 4.2.4 D.C. Amplifier Stage

Capacitor C10 which is coupled between the collector terminal and the base terminal of Q1, integrates the positive signal pulses from the rectifier stage, so that Q1 is continuously biased into conduction and not rapidly pulsed on and off.

When the receiver and transmitter are aligned with each other Q1 will be switched on. The output on the collector terminal will be at earth potential and will be fed to the B-input of the logic circuit. This low level will cause D6 to be switched off to indicate that the receiver and transmitter are aligned. If not D 6 will be on.

The value of capacitor C9 is critical, because it controls the time taken for the d.c. signal to reach the point where the transistor switches off once the beam has been broken. This release time must be kept to a fraction of a second so that the transistor always switches off before the person breaking the beam moves on and the d.c. signal is restored.

### 4.2.5 Logic Stage

The Dual Retriggerable Monostable Multivibrator (74C123) is a micro CMOS type i.c., and it is used to achieve high operating speeds, low power consumption and high immunity to noise.

The inputs and outputs are connected as follows:

(i) Both A-inputs are connected to ground.

(ii) Both CLR-inputs are connected to VCC.

(iii) The B-inputs are connected to the outputs of the amplifier stages.

(iv) Both not-Q outputs are connected to the INT0 and INT1 of the microprocessor in the Log-Card.

The Multivibrator can be configured in three different ways to deliver negative going pulses on it's $\overline{Q}$-outputs (Refer to Fig 4.1). The transition from high to low, on the $\overline{Q}$-output of the Multivibrator interrupts the microprocessor in the Log-

Card. Fig 4.2 shows the typical output pulse width of the Multivibrator vs. the timing components R9, R10, C21 and C22. Tests with different values were made to get the correct pulse width. Too short a pulse width will cause the microprocessor to be interrupted more than once and thus may cause incorrect "Date-Time" data.

*FIGURE 4.1 TRUTH TABLE*

| INPUTS | | | OUTPUTS | |
|---|---|---|---|---|
| CLEAR | A | B | Q | NON-Q |
| L | X | X | L | H |
| X | H | X | L | H |
| X | X | L | L | H |
| H | L | ↑ | ⎍ | ⎍ |
| H | ↓ | H | ⎍ | ⎍ |
| ↑ | L | H | ⎍ | ⎍ |

H = HIGH LEVEL
L = LOW LEVEL
↑ = TRANSITION FROM LOW TO HIGH
↓ = TRANSITION FROM HIGH TO LOW

⎍ = ONE HIGH LEVEL PULSE
⎍ = ONE LOW LEVEL PULSE
X = IRRELEVANT

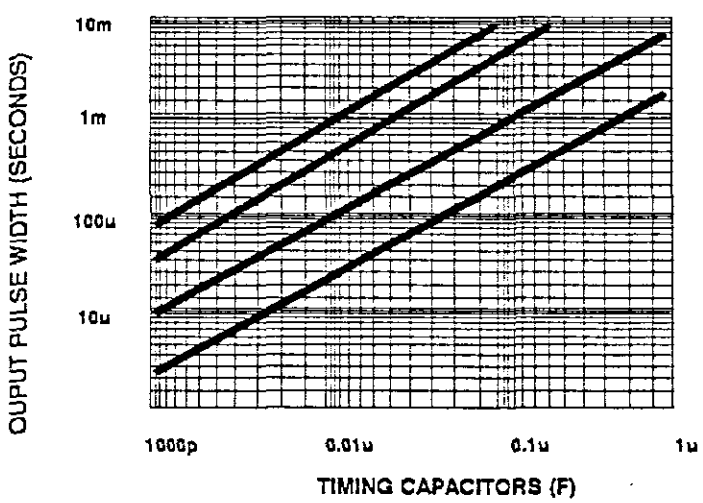



*FIGURE 4.2*
*TYPICAL OUTPUT PULSE WIDTH vs. TIMING COMPONENTS*

## 5. CONSTRUCTION OF THE INFRA-RED SYSTEM

### 5.1 Construction

The infra-red LEDs and the photo-diodes on the PCBs are placed apart to obtain a minimum distance between the infra-red beams. Before the final PCBs were constructed tests on prototype boards were made to determine the correct distance between the two beams. The best working result is obtained when the space between the LEDs as well as the space between the photo-diodes is 19.7 cm. It was found that when the LEDs and photo-diodes on the PCBs were spaced too closely, interference between the two beams occurred and thus caused false operation.

The leads of the infra-red LEDs and photo-diodes are trimmed quite short so that they do not protrude too far above the PC boards. This is important because the distance from the "lenses" to the LEDs and photo-diodes will be inadequate if the leads are left too long, giving a mediocre level of performance. These "lenses" mentioned, are two circular windows on the front of the plastic cases in which the PCBs are mounted. These windows are accurately positioned directly in front of the LEDs.

In practice without lens focussing there is some reduction in beam strength at the receiver side if the range between

receiver and transmitter units is increased, due to consequent spread of the beam and the fact that air is not totally transparent. Therefore, even with these simple and inexpensive type of "lenses" the infra-red beams from the transmitter can now be focused into narrow beams and the usual fall in strength as the range between the transmitter and receiver units increases, is avoided. Refer to Fig 5.1. If more infra-red LEDs are connected to the existing infra-red LEDs, an increase in the output signal of the transmitter will be caused.

Although the timer IC in the transmitter unit is a CMOS type device, it has a built in protection circuit which

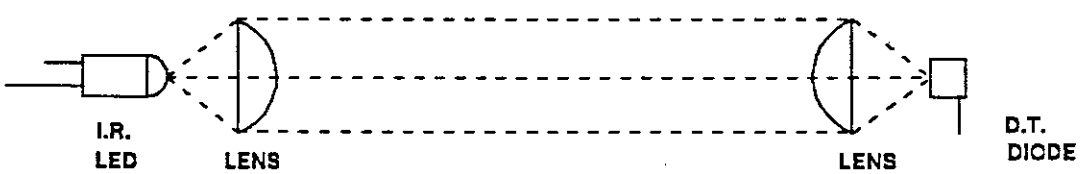



*FIGURE 5.1 SIMPLE LENS SYSTEM*

renders the usual anti-static handling precautions unnecessary. It is nevertheless, advisable to use sockets for all ICs.

The use of screening for the SL486 in the receiver circuits and associated components improves the performance and immunity to externally radiated noise. The screening method used must protect the sensitive front-end of the SL486. Provided that pin 1, pin 16, pin 2 and pin 15 are screened, it was found that for this application, the remaining circuitry need not be protected to the same degree.

As the infra-red beams are going to be used outdoors, it is essential that the transmitter and receiver circuits are completely waterproof. Special plastic cases, having outside dimensions of 6 x 5 x 5.5 cm were designed to accommodate the transmitter and receiver circuits. Each case has a removable lid at the back, onto which the units are mounted.

The circuit boards are mounted onto the lid with adhesive tape. A single hole is drilled on any side of each case to pass the power and signal leads through. After the leads are pulled through, the holes are sealed with silicone rubber. To make it more impenetrable to water, silicon is also placed right around the lids, before they are screwed to the cases.

The receiver case has two small window openings on the top, above the alignment LEDs. When the case is sealed, these windows allow the alignment LEDs to be seen.

## 5.2 Installation

Initially the units should be tested at a relatively short range, as they are then quite easily aligned. However, even at this short range the system is still highly directional and they will need to be aimed accurately in order to obtain proper operation. The system can easily be aligned with the two LEDs in the receiver unit.

Setting up the system over a distance of several meters or more is a little more difficult and the main problem is ensuring that the transmitters are aimed at the point where the receivers are situated. The receivers only have to be slightly outside the main beams in order to reduce the strength of the received signal to an inadequate level.

The best method is to aim the transmitters in the right general direction and then to move the receivers to a point where both alignment LEDs are off. The power and signal leads are made long enough so that the receiver and transmitter units can easily be situated far from the power supply.

# 6. LOG-CARD

## 6.1 Introduction

Refer to drawing No. A3, A4 Appendix A for the schematic diagrams of the Log-card. The Log-card, with the aid of the infra-red monitoring system, is used to log the date, time and direction of the monitored person, onto a RAM-card. This logged data is then loaded onto a PC's hard disk when required.

## 6.2 Hardware Design

### 6.2.1 Display

The LTN111 is a 5x7 dot, 16-character, 1-line dot matrix LCD module, with driver and controller LSIIC mounted on a single printed circuit board. Contrast is adjusted (R3) by varying the voltage between 0 and 5V. It has a built-in 160 character generator. The supply voltage must be between 4.75 and 5.25V.

### 6.2.2 Hex Keyboard Encoder

The 74C922 encoder has an internal debounce circuit that needs only a single external capacitor. The Data Available output is connected to port (P1.0) pin of the microprocessor to indicate when a key is pressed. A 3x4 keypad is used with

the encoder and therefore a look-up table is used, to scan for the correct key pressed.

### 7.2.3 Processor and Latch

A CMOS type (designated with "C" in the middle of the device name) microprocessor (87C51) is used. All CMOS devices have a low current consumption. The 87C51 has 4 ports and for this application the ports will be used as follows:

(i) Port 0 is an 8-bit open drain bi-directional I/O port. It is also the multiplexed low-order address and data bus during access to the RAM-card, Real-Time Clock and Keyboard Encoder.

(ii) Port 1 not used.

(iii) Port 2 is an 8-bit bi-directional I/O port. This port emits the high-order address byte during access to the RAM-card.

(iv) Port 3 is used as follows:

* P3.2 $\overline{\text{INT0}}$ (external interrupt 0)
* P3.3 $\overline{\text{INT1}}$ (external interrupt 1)
* P3.6 $\overline{\text{WR}}$ (external data memory write strobe)
* P3.7 $\overline{\text{RD}}$ (external data memory read strobe)

The reset input (RST), will reset the device if the RST pin is kept high for two machine cycles while the oscillator is running. The RST pin is connected to VCC via a 10 $\mu$F capacitor and to ground via a 8.2 k$\Omega$ resistor to reset the device automatically when power is applied.

The Address Latch Enable (ALE) output is connected to the C input of the 74HC573 latch. The pulse from the ALE output is used to latch the low byte of the address during accesses to external memory.

The External Access ($\overline{EA}$) pin must be strapped to VCC for internal program execution. The 87C51 contains 4K bytes of on-chip Program memory that can be electrically programmed and be erased by exposure to ultraviolet light.

For this application the 87C51 is using the on-chip oscillator with an 11.059 MHz crystal. The crystal is connected between the XTAL1 and XTAL2 pins.

### 6.2.4 Real-Time Clock (RTC)

The MM58274 Microprocessor Compatible RTC is used. The reason for using a real time clock is to log the correct date and time of the monitored person onto the RAM-card. The RTC also shares a part of the Address (A0 - A3) and Data Bus (D0 - D3) with the RAM. The RTC has a Read ($\overline{RD}$), Write ($\overline{WR}$) and Chip

Select ($\overline{CS}$) as its control inputs. The RTC has 15 internal registers to read from or to write to. The $\overline{WR}$ input are used to initialize the RTC internal registers, while the $\overline{RD}$ input is used to read the date and time from these registers. The RTC uses the on-chip oscillator with a 32.768 KHz crystal.

### 6.2.5 3-to-8 Line Decoder

The processor board is memory-mapped and selection of the various devices eg. Real-Time Clock, RAM, etc. is done by the 74HC138 under processor control. Three inputs (A, B and C) provides a selection of eight outputs. Only three outputs are used in this application.

# 7. BAS-CARD

## 7.1 Introduction

Refer to drawing No. A5 Appendix A for the schematic diagram of the Bas-card. The Bas-card is the interface between the PC and the RAM-card.

## 7.2 Hardware Design

### 7.2.1 Processor and Latch

An 87C51 processor is used to form the heart of the Bas-Card, as in the case with the Log-Card. Port 3 is used as follows:

* P3.0 RXD (serial input port)
* P3.1 TXD (serial output port)
* P3.6 $\overline{WR}$ (external data memory write strobe)
* P3.7 $\overline{RD}$ (external data memory read strobe)

The Address/Data is shared on Port 0 and the lower address-byte is latched by the 74HCT573.

### 7.2.2 MAX-232

The MAX 232 provides the required voltage levels for RS-232 serial communication between the Bas-Card and the PC. The MAX

232 has three sections: a dual transmitter, a dual receiver and a +5V to approximately 10V dual charge pump voltage converter. For this application only one receiver and transmitter is used.

The voltage converter is used to obtain the correct level on the serial port during the transmission and reception of data. Refer to any PC manual for the RS-232 connections.

The RS-232 serial port is full duplex, meaning it can transmit and receive simultaneously. The serial port can operate in 4 modes. For this application the port operates in mode 1 i.e. 10 bits are transmitted or received: a start bit, 8 data bits and a stop bit.

Timer 1 is used to generate a baud rate of 1200. The serial port transmits and receives the bits through a serial buffer, SBUF. The initialisation of the serial port, the transmission and reception of information, can be seen from the program listing in Appendix B.

## 8 RAM-CARD

Refer to drawing No. A6 Appendix A for the schematic diagram of the RAM-Card.

The 43256 Static RAM is used. The 43256 has three control inputs, Chip Select ($\overline{CE}$), Output Enable ($\overline{OE}$) and Write Enable ($\overline{WR}$). These inputs must be logically active in order to write data to the device or to obtain data from the device. The $\overline{OE}$ and $\overline{WR}$ is directly connected to the $\overline{RD}$ and $\overline{WR}$ outputs of the 87C51. With software addressing, Address A15 is used to select the RAM. The 43256 has a 32K x 8 memory capacity. The RAM can be interfaced with Bas-card or Log-card via an edge connector.

The RAM is permanently mounted on a DS1213C SmartSocket to provide a complete solution to problems associated with memory volatility. The SmartSocket monitors incoming VCC for an out-of-tolerance condition. When such a condition occurs, an internal lithium source is automatically switched on and write protection is unconditionally enabled to prevent invalid data.

# 9. SOFTWARE DEVELOPMENT

## 9.1 Program Development Tools

The INTEL ICE 5100/252 was used for the development of this project. The ICE in-circuit emulator is a high level, interactive debugging system that is used to develop and test the hardware and software of a target system based on the MCS-51 family of microcontrollers. Therefore the software and hardware can be system tested in real-time operation.

The 87C51's in the Log-card and Bas-card are programmed with programs that were written in assembly language. The programs were compiled with the ASM51 compiler. The advantage of writing in assembly language, is the reduction of execution time to a minimum.
Refer to Appendix B and Appendix C for the program listings and the flow diagrams.

A Turbo Pascal program was written to capture the logged data from the RAM-card via the serial port and Bas-card. This program can also be used for a screen display of the statistics of all the months of a specific path and year, or to print the monthly, daily and hourly statistics for a specific path and year. The Bas-card.EXE file was generated after the Pascal program was compiled.

Refer to Appendix D for the listing of this Turbo Pascal program.

## 9.2 Addressing Log-Card

The processor board is memory-mapped as follows:

### 9.2.1 Ram-Card

A15 selects the RAM-Card.

Starting Address: 0000-Hex

End Address : 7FFF-Hex

### 9.2.2 Real-Time Clock

Y4 selects the Real-Time Clock.

Starting Address: 8000-Hex

End Address : 800F-Hex

Refer to Fig 9.1 for the address decoding of the internal registers of the RTC.

There are three registers which control different operations of the RTC:

(i) The clock setting register, which is used for setting up:

a) the leap year counter

b) the AM/PM indicator

c) the 12/24 hour mode.

Refer to Fig 9.2 for the layout of the clock setting register.

(ii) The interrupt register, which can be programmed as follows:

a) to control the interrupt timer which generates interrupts at time intervals.

b) to select the required delay period.

c) to be a single or repeated interrupt timer.

Refer to Fig 9.3 for the listing of different time delays and the data words that select them in the interrupt register.

(iii) The control register. The control register is responsible for controlling the operation of the clock and supplying status information to the microprocessor. Refer to Fig 9.4 for the control register layout.

### 9.2.3 Hex Keyboard Encoder

Y5 selects the Encoder. The Encoder Address is A000-Hex. No control words are required to initialize the encoder.

### 9.2.4 Display

Y6 selects the LTN111 Display. The Display Address is C000-Hex. The LTN111 operates from an extensive instruction set

where the instructions are sent to the Control Word Address C000-Hex. Refer to Fig 9.5 for the instruction set.

## 9.3 Addressing Bas-Card

### 9.3.1 Max-232

No addressing is required. The serial port is set up as follows:

```
MOV TH1,#E8H ┐
             ├ This sets the Timer up for a baud rate of 1200.
MOV TL1,#00H ┘
MOV TMOD,#20H  Specify an 8-bit auto-reload timer.
MOV SCON,#50H  To set up the UART register.
SETB TR1       To start timer.
```

Information is sent to the serial port as follows:

```
MOV SBUF,A     Move the contents in the accumulator to the
               serial port.
JNB TI,$       Pause until byte has been sent.
```

Information is received on the serial port as follows:

```
JNB,RI,$       Pause until the byte has been received.
MOV A,SBUF     Move the contents from the serial port into
               the accumulator.
```

***FIGURE 9.1 ADDRESS DECODING OF REAL-TIME CLOCK INTERNAL REGISTERS***

| Register Selected | Address (Binary) | | | | ( HEX ) | ACCESS |
|---|---|---|---|---|---|---|
| | AD3 | AD2 | AD1 | AD0 | | |
| 0 Control Register | 0 | 0 | 0 | 0 | 9000 | SPLIT READ and WRITE |
| 1 Tenth of Seconds | 0 | 0 | 0 | 1 | 9001 | READ ONLY |
| 2 Units Seconds | 0 | 0 | 1 | 0 | 9002 | R/W |
| 3 Tens Seconds | 0 | 0 | 1 | 1 | 9003 | R/W |
| 4 Units Minutes | 0 | 1 | 0 | 0 | 9004 | R/W |
| 5 Tens Minutes | 0 | 1 | 0 | 1 | 9005 | R/W |
| 6 Units Hours | 0 | 1 | 1 | 0 | 9006 | R/W |
| 7 Tens Hours | 0 | 1 | 1 | 1 | 9007 | R/W |
| 8 Units Days | 1 | 0 | 0 | 0 | 9008 | R/W |
| 9 Tens Days | 1 | 0 | 0 | 1 | 9009 | R/W |
| 10 Units Months | 1 | 0 | 1 | 0 | 900A | R/W |
| 11 Tens Months | 1 | 0 | 1 | 1 | 900B | R/W |
| 12 Units Years | 1 | 1 | 0 | 0 | 900C | R/W |
| 13 Tens Years | 1 | 1 | 0 | 1 | 900D | R/W |
| 14 Day of Week | 1 | 1 | 1 | 0 | 900E | R/W |
| 15 Clock Setting/ Interrupt Registers | 1 | 1 | 1 | 1 | 900F | R/W |

***FIGURE 9.2 CLOCK SETTING REGISTER LAYOUT***

| Function | Data Bits Used | | | | Comments | Access |
|---|---|---|---|---|---|---|
| | DB3 | DB2 | DB1 | DB0 | | |
| Leap Year Counter | X | X | | | 0 indicates a Leap Year | R/W |
| AM/PM Indicator | | | X | | 0 = AM 1 = PM | R/W |
| 12/24-Hour Selected Bit | | | | X | 0 = 12-Hour 1 = 24-Hour Mode | R/W |

***FIGURE 9.3 INTERRUPT CONTROL REGISTER***

| Function | Comments | Control Word | | | |
|---|---|---|---|---|---|
| | | DB3 | DB2 | DB1 | DB0 |
| No Interrupt | Interrupt output cleared start/stop bit set to 1 | X | 0 | 0 | 0 |
| 0.1 Second | DB3 = 0 for single interrupt DB3 = 1 repeated interrupt | 0/1 | 0 | 0 | 1 |
| 0.5 Seconds | | 0/1 | 0 | 1 | 0 |
| 1 Seconds | | 0/1 | 0 | 1 | 1 |
| 5 Seconds | | 0/1 | 1 | 0 | 0 |
| 10 Seconds | | 0/1 | 1 | 0 | 1 |
| 30 Seconds | | 0/1 | 1 | 1 | 0 |
| 60 Seconds | | 0/1 | 1 | 1 | 1 |

***FIGURE 9.4 THE CONTROL REGISTER LAYOUT***

| Access (address 0) | DB3 | DB2 | DB1 | DB0 |
|---|---|---|---|---|
| Read From : | Data-Change Flag | 0 | 0 | Interrupt Flag |
| Write to : | Test<br>0 = Normal<br>1 = Test Mode | Clock Start/Stop<br>0 = Clock Run<br>1 = Clock Stop | Interrupt Select<br>0 = Clock Setting Register<br>1 = Interrupt Register | Interrupt Start/Stop<br>0 = Interrupt Run<br>1 = Interrupt Stop |

| INSTRUCTION \ ADDRESS | RS | R/W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|
| DISPLAY CLEAR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| CURSOR HOME | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | * |
| ENTRY MODE SET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S |
| DISPLAY ON/OFF CONTROL | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B |
| CURSOR DISPLAY SHIFT | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | * | * |
| FUNCTION SET | 0 | 0 | 0 | 0 | 1 | DL | 1 | 0 | * | * |
| CG RAM ADDRESS SET | 0 | 0 | 0 | 1 | ACG | | | | | |
| DD RAM ADDRESS SET | 0 | 0 | 1 | ADD | | | | | | |
| BUSY FLAG/ADDRESS READ | 0 | 1 | BF | AC | | | | | | |
| CG RAM/DD RAM DATA WRITE | 1 | 0 | WRITE DATA | | | | | | | |
| CG RAM/DD RAM DATA READ | 1 | 1 | READ DATA | | | | | | | |

NOTES: I/D = 1 : INCREMENT — I/D = 0 : DECREMENT

S = 1 : DISPLAY SHIFT — S = 0 : DISPLAY FREEZE

D = 1 : DISPLAY ON — D = 0 : DISPLAY OFF

C = 1 : CURSOR ON — C = 0 : CURSOR OFF

B = 1 : CHARACTER AT CURSOR POSITION BLINKS — B = 0 : CHARACTER AT CURSOR POSITION DOES NOT BLINK

S/C = 1 : DISPLAY SHIFT — S/C = 0 : CURSOR MOVE

R/L = 1 : RIGHT SHIFT — R/L = 0 : LEFT SHIFT

DL = 1 : 8 BITS — DL = 0 : 4 BITS

BF = 1 : DURING INTERNAL OPERATION — BF = 0 : END OF INTERNAL OPERATION

*FIGURE 9.5 INSTRUCTION SET*

## 10 POWER SUPPLY

Refer to drawing No. A7 Appendix A for the schematic diagram of the primary power supply. This power supply circuit is designed to supply the transmitter with 6 to 9 V DC, the receiver and the Log-Card with 5 V DC voltage.

The total current consumption is 27 mA and the voltage needed is 9 V, therefore a battery with a capacity of at least 20 Ah is needed to run the system for about a month.

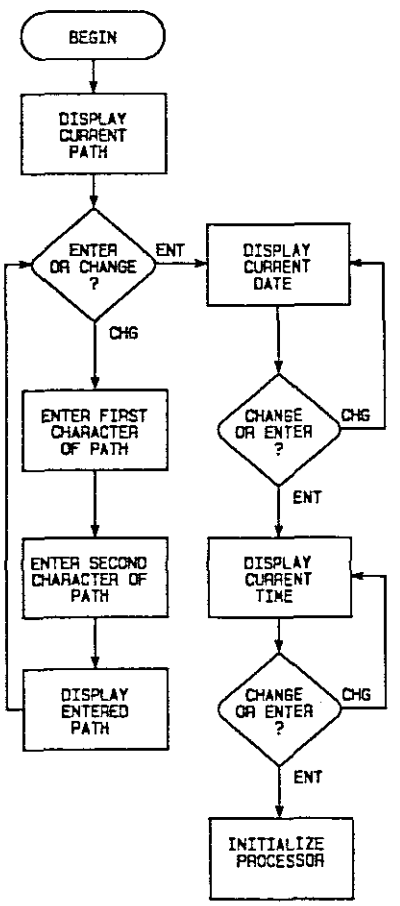
Refer to drawing No. A5 Appendix A for the schematic diagram of the primary power supply of the Bas-card. The power supply, designed to supply the Bas-card circuitry with the necessary power and to function correctly, is a very basic yet efficient supply. It is a simple rectifier with a regulator that converts the 220 V AC voltage to a +5 V DC.

# 11 OPERATING INSTRUCTIONS

## 11.1 Initialize Log-Card

Refer to Appendix B for the Log-Card program listing.

When both switches are on, the 87C51 microprocessor is reset and the program is initialized and executed in the following order:



The user will be prompted with PATH=..CHG/ENT. When the #-key (CHG) is pressed the following message is displayed:

ENTER PATHNUM:A_ --------1

Now the first path number can be entered. After the #-key (CHG) is pressed, an alphabetical letter A to Z is displayed. When the user is satisfied with the displayed letter, the *-key (ENT) can be pressed to place the first path number in the first address (0000H) location of the RAM-Card's memory. The following message is displayed:

ENTER PATHNUM:AA --------2

To enter the second path number the same procedure must be followed. When the *-key is pressed the second path number is placed in the second address (0001H) location of the RAM-Card's memory. Both Ram-card's path numbers are displayed eg:

PATH=CD CHG/ENT --------3

If the user is not satisfied with the path numbers the #-key (CHG) can be pressed and the program will return to procedure 1. If the *-key (ENT) is pressed the following message is displayed.

DATE: YY/MM/DD --------4

The Year, Month and Day can be entered. After the last digit has been entered the date is displayed eg:

92/02/02 CHG/ENT

If the date is incorrect the #-key (CHG) can be pressed to re-enter the date. The program will return to procedure 4. If correct the *-key (ENT) can be pressed to display the next message.

TIME: **:**:** --------5

The Hour, Minute and Seconds can be entered. After the last digit has been entered the time is displayed eg:

08:08:08 CHG/ENT

If the time is incorrect the #-key (CHG) can be pressed to re-enter the time. The program will return to procedure 5. If correct the *-key (ENT) can be pressed. The interrupt registers are enabled and the processor is forced into idle mode to minimize the total current consumption. INTERRUPT 0 and INTERRUPT1 are set up to be activated through edge triggering, meaning that the 87C51 will execute an interrupt routine when a negative going edge is received on the INT0 or INT1 inputs.

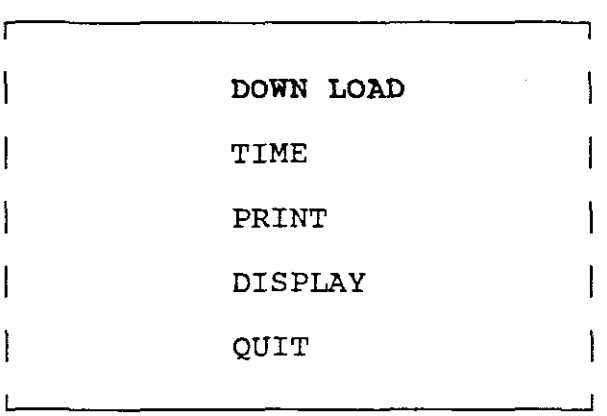
Every time an interrupt occurs, the date and time will be fetched from the Real-Time Clock registers and will be placed in the next RAM-Card's memory location.

**NB: Before removing the Ram-card from the Log-card the tagged switch must be switched off.**

## 11.2 Initialize Bas-Card

In order to down-load the data from the RAM-card, a serial cable, PC and a software package that can capture the data from the serial port are required. The PASCAL program, Bas-card.Pas (Refer to Appendix D), is used to capture the data from the Bas-card at a baud rate of 1200.

Connect the Bas-card to the PC's RS-232 port with a serial cable and switch the Bas-card on. The program, Bas-card.bin (Refer to Appendix C), automatically sets the Bas-card's microprocessor up for a 1200 baud rate. When the Bas-card.exe file is run, the following screen menu is displayed:



Each choice will be discussed individually.

### 11.2.1 Down Load

When the DOWN LOAD choice on the screen menu is entered the following screen menu is displayed:

```
┌──────────────────────────────────┐
│  ENTER PATH NO: AA - ZZ =        │
│  Press ESC to Exit to Menu       │
└──────────────────────────────────┘
```

When the path numbers entered are the same as the path numbers on the RAM-Card the following will happen:

(i) The Bas-card.EXE file will open a file with an extension the same as the path numbers entered.

(ii) The data is then loaded from the RAM-Card to the data file.

If the path numbers entered differ from the path numbers on the RAM-card the following menu is displayed:

```
┌──────────────────────────────────────────┐
│  !!! CARD NOT INSERTED OR WRONG CARD !!! │
│          Press ESC to Exit to Menu       │
└──────────────────────────────────────────┘
```

### 11.2.2 Print

When the PRINT choice is entered the following menu is displayed:

```
┌──────────────────────────────┐
│ ENTER PATH NO: AA - ZZ =     │
│ ENTER YEAR 9_                │
└──────────────────────────────┘
```

When the path numbers and year are entered, the following menu is displayed:

```
┌──────────────────────────┐
│ A = MONTHLY PRINTOUT     │
│ B = DAILY PRINTOUT       │
│ C = HOURLY PRINTOUT      │
└──────────────────────────┘
```

A : the statistics for each month of the year entered are printed.

B : the statistics for each day of the month entered are printed.

C : the statistics for each hour of the month entered are printed.

### 11.2.3 Display

When the DISPLAY choice is entered, the year and path numbers are required to display the monthly statistics. With these statistics it can then be determined if it is necessary to open alternative routes.

## 12 PROBLEMS ENCOUNTERED

(i) The major problem encountered throughout the duration of this project was interference caused by mains lighting and UV-radiation from the sun. Due to this problem it was extremely difficult to get reliable results from the receiver circuits. The problem was overcome by using a special type of infra-red photodiode and preamplifier (Sl486). The photodiode's photosensitive area is housed in a black infra-red transmissive moulding which reduces ambient white light interference. The differential inputs of the preamplifier (SL486) further reduce noise-pickup and improve stability. Screening the infra-red diode and the SL486 improves the performance and immunity to externally generated noise.

(ii) Due to the fact that the system is going to be used in the field and therefore powered by batteries, it will be necessary to keep the current consumption to a minimum. The total current consumption was minimized by using CMOS version IC's in all the circuits. The current consumption is further reduced by a power-reduction mode in the Log-card's microprocessor, viz., IDLE MODE. In this mode the oscillator continues to run and the Interrupt, Serial Port and Timer blocks

continue to be clocked but the clock signal is gated off to the CPU. An instruction that sets PCON.0 (ORL PCON,#1) causes that to be the last instruction executed before going into Idle mode.

(iii) A futher problem was the loss of data from the RAM-Card when VCC drops to below 2.0 V or when it is removed. The problem was overcome by using a SmartSocket with a normal static RAM. The SmartSocket has an embedded lithium energy source. The socket monitors the incoming VCC and when VCC falls below 4.75 V it automatically switches over to lithium energy source. With this method, VCC can be removed and the RAM-Card can now be transferred between the Log-card and the Bas-card.

## 13. TEST RESULTS

It was the objective to design and develop an infra-red monitoring and data logging system that could be used to monitor the traffic on Table Mountain. Due to the fact that the designed system is only a prototype it has not yet been tested in the targeted environment or in different weather conditions. The following important tests were done to determine if this prototype system would meet the original requirements as laid down at the beginning of the project.

(i) The monitoring part of the system was tested in direct sunlight to determine if the infra-red radiation from the sun would have an effect on the system. The result was that the sun's radiation had no effect on the system.

(ii) The accuracy of the system was tested by installing the system at a entrance of a room where traffic is heavy. The traffic was monitored in both directions from 07h00 on the 15th till 07h00 on the 16th of January 1992. From the information accumulated, daily (p13-2) and hourly (p13-3, p13-4) reports were produced. The results show an accuracy of 98%.

MONTHLY REPORT for YEAR 1992

Today is:Wednesday 1992/ 4/ 8
Time is: 7:43:33
TRAIL:AA

| MONTHS | UP | DOWN |
|---|---|---|
| JAN | 263 | 266 |
| FEB | 0 | 0 |
| MAR | 0 | 0 |
| APR | 0 | 0 |
| MAY | 0 | 0 |
| JUN | 0 | 0 |
| JUL | 0 | 0 |
| AUG | 0 | 0 |
| SEP | 0 | 0 |
| OCT | 0 | 0 |
| NOV | 0 | 0 |
| DEC | 0 | 0 |
| TOTAL = | 263 | 266 |

DAILY REPORT for JAN 1992

Today is:Wednesday 1992/ 4/ 8
Time is: 7:43:38
TRAIL:AA

| DAYS | UP | DOWN |
|---|---|---|
| 01 | 0 | 0 |
| 02 | 0 | 0 |
| 03 | 0 | 0 |
| 04 | 0 | 0 |
| 05 | 0 | 0 |
| 06 | 0 | 0 |
| 07 | 0 | 0 |
| 08 | 0 | 0 |
| 09 | 0 | 0 |
| 10 | 0 | 0 |
| 11 | 0 | 0 |
| 12 | 0 | 0 |
| 13 | 0 | 0 |
| 14 | 0 | 0 |
| 15 | 256 | 259 |
| 16 | 7 | 7 |
| 17 | 0 | 0 |
| 18 | 0 | 0 |
| 19 | 0 | 0 |
| 20 | 0 | 0 |
| 21 | 0 | 0 |
| 22 | 0 | 0 |
| 23 | 0 | 0 |
| 24 | 0 | 0 |
| 25 | 0 | 0 |
| 26 | 0 | 0 |
| 27 | 0 | 0 |
| 28 | 0 | 0 |
| 29 | 0 | 0 |
| 30 | 0 | 0 |
| 31 | 0 | 0 |
| TOTAL = | 263 | 266 |

HOURLY REPORT for JAN 1992

Today is: Wednesday 1992/ 4/ 8
Time is: 7:43:47
TRAIL: AA

UP

| HOURS / DAYS | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | TOTALS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 02 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 03 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 04 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 05 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 06 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 07 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 08 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 09 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 10 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 11 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 12 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 13 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 14 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 15 | . | . | . | . | . | . | . | 16 | 19 | 25 | 39 | 35 | 29 | 22 | 5 | 23 | 25 | 5 | 9 | 4 | . | . | . | . | 256 |
| 16 | . | . | . | . | . | . | 7 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 7 |
| 17 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 18 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 19 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 20 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 21 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 22 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 23 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 24 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 25 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 26 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 27 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 28 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 29 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 30 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 31 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |

HOURLY REPORT for JAN 1992

Today is: Wednesday 1992/ 4/ 8
Time is: 7:43:47
TRAIL: AA

DOWN

| HOURS / DAYS | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | TOTALS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 02 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 03 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 04 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 05 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 06 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 07 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 08 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 09 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 10 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 11 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 12 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 13 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 14 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 15 | . | . | . | . | . | . | . | 31 | 19 | 29 | 41 | 33 | 29 | 17 | 6 | 19 | 14 | 4 | 15 | 2 | . | . | . | . | 259 |
| 16 | . | . | . | . | . | . | 6 | 1 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 7 |
| 17 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 18 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 19 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 20 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 21 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 22 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 23 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 24 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 25 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 26 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 27 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 28 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 29 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 30 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |
| 31 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 0 |

## 14. GLOSSARY OF TERMS

**A.F.** - Audio Frequency

**ALE** - Address Latch Enable

**CMOS** - Complementary Metal-Oxide Semiconductor

**d.c.** - Direct Current

**I.C.** - Integrated Circuit

**ICE** - In Circuit Emulator

**I/O** - Input/Output

**LED** - Light-Emitting Diode

**PC** - Personal Computer

**PCB** - Printed Circuit Board

**RAM** - Random Access Memory

**VMOS** - Vertical Metal-Oxide Semiconductor

**$\mu$P** - Microprocessor

## 15. BIBLIOGRAPHY


Borland International. **Turbo Pascal 4.0 IBM Version.** Borland International, Inc.

Intel Coporation. 1989. **8-bit Embedded Controller Handbook.** Santa Clara, Intel Literature Sales.

Intel Coporation. 1990. **Memory Components Handbook.** Mt. Prospect, Intel Literature Sales.

Maxim. 1990. **Integrated Products Handbook.** Sunnyvale CA.

National Semiconductors Coporation. 1988. **Linear Data Book.** Santa Clara.

National Semiconductors Coporation. 1988. **CMOS Data Book.** Santa Clara.

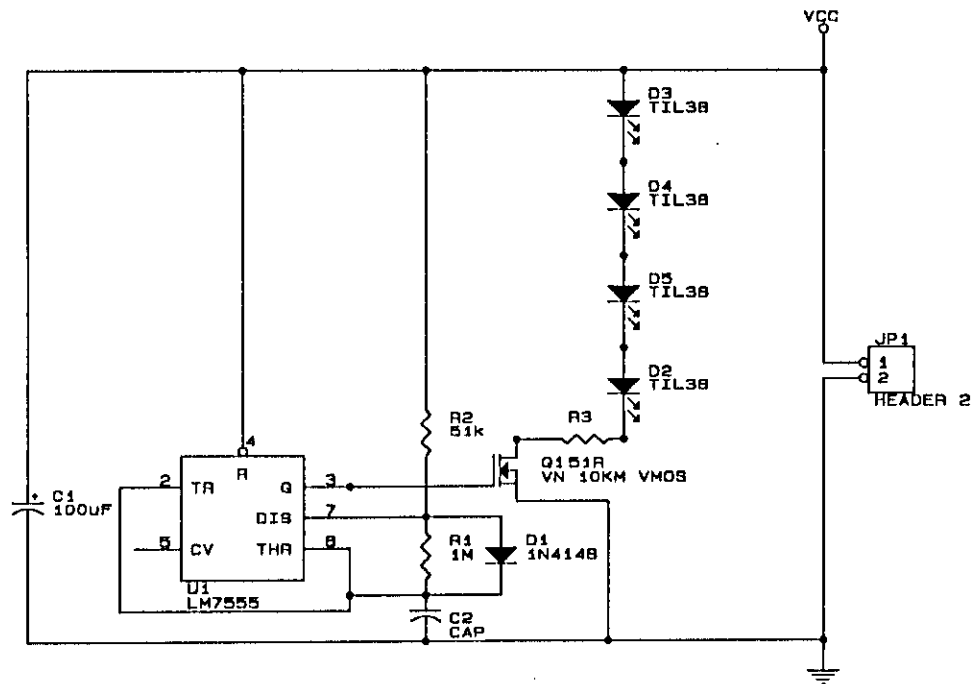
RS Export Centre. 1990. **RS Components Catalogue.** Corby Northants England.

Tarsus Technologies. 1990-1991. **Dallas Semiconductors.** Dallas Texas.

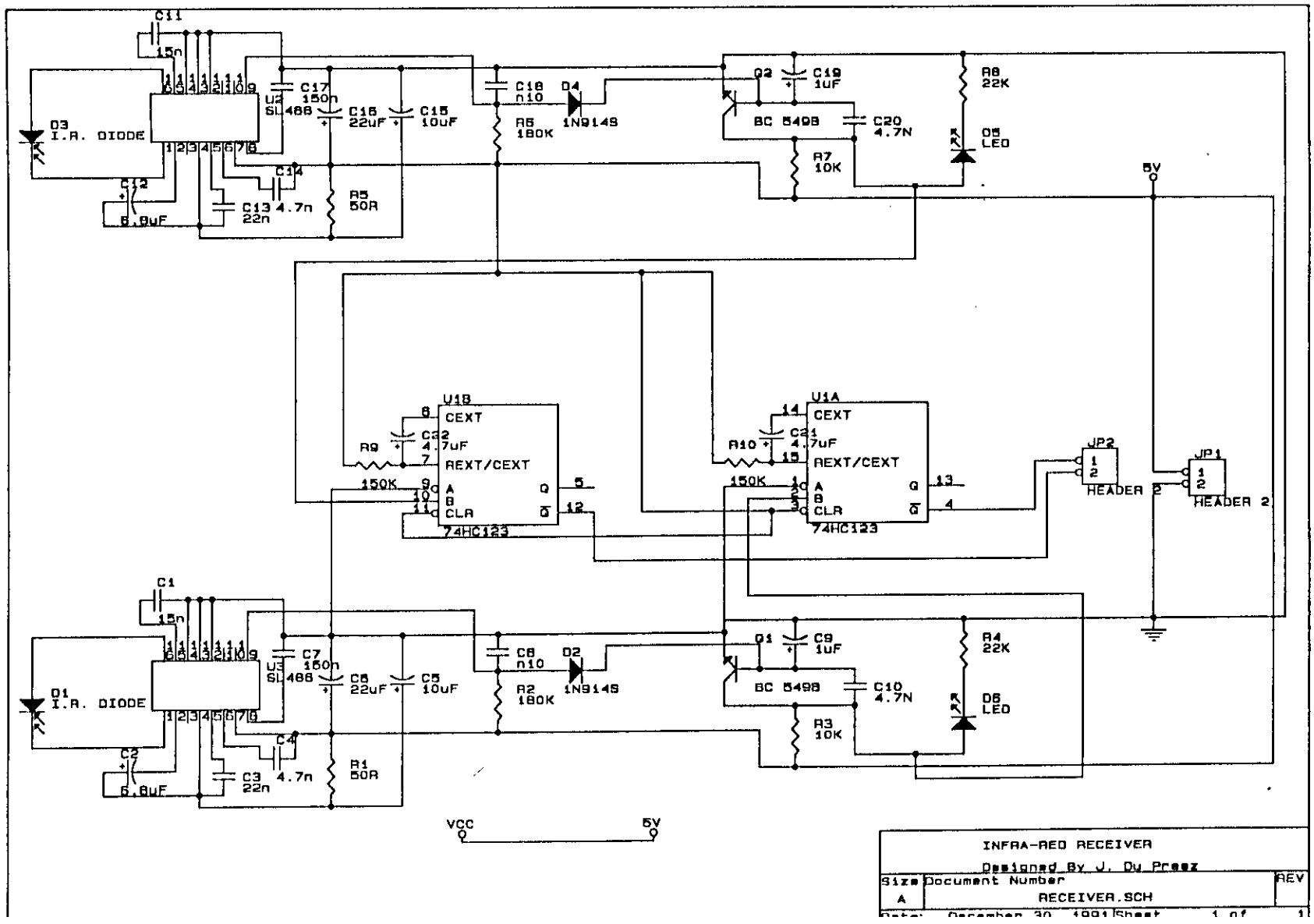
# APPENDIX A

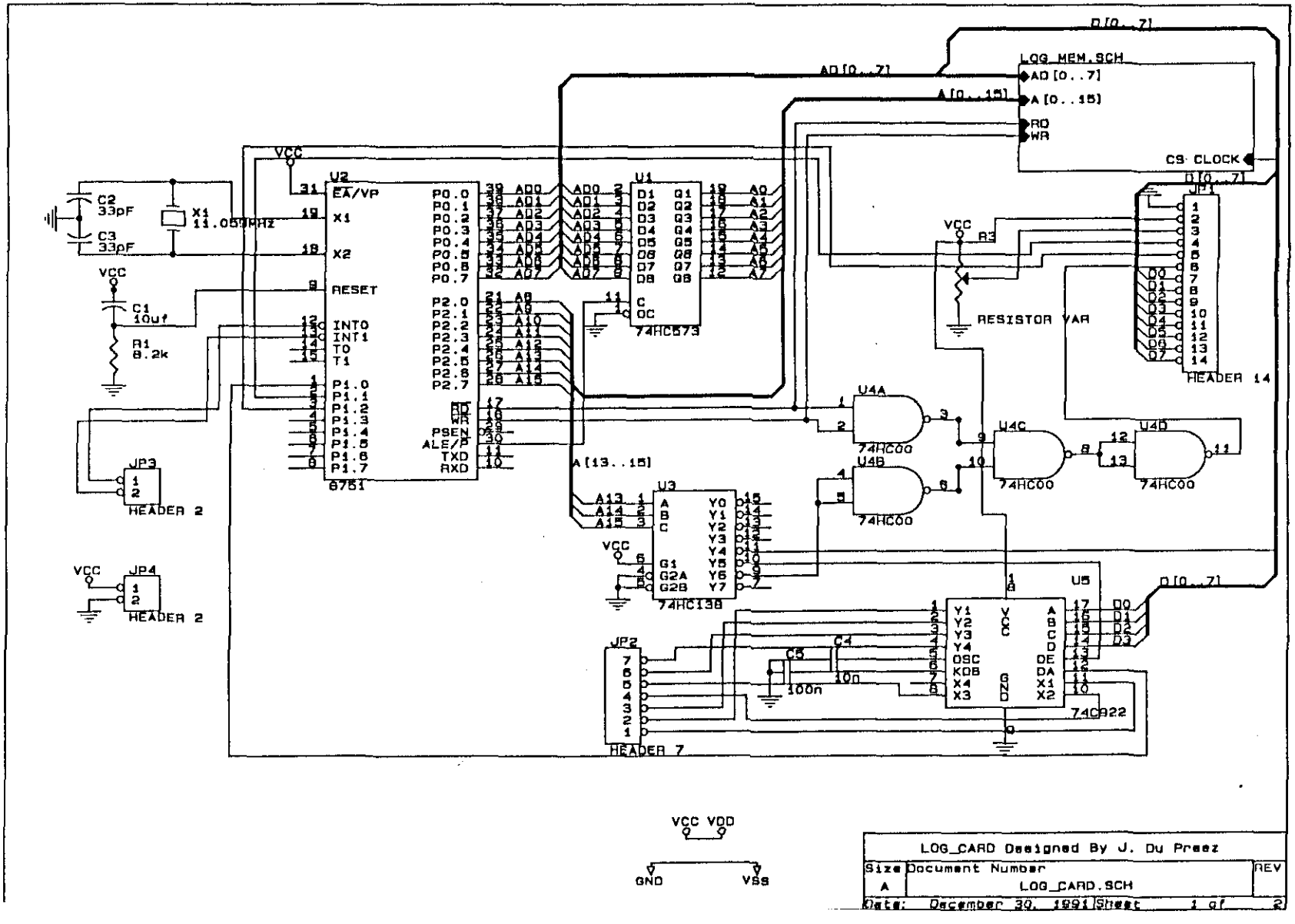
## SCHEMATIC DIAGRAMS OF THE INFRA-RED MONITORING AND DATA LOGGING SYSTEM

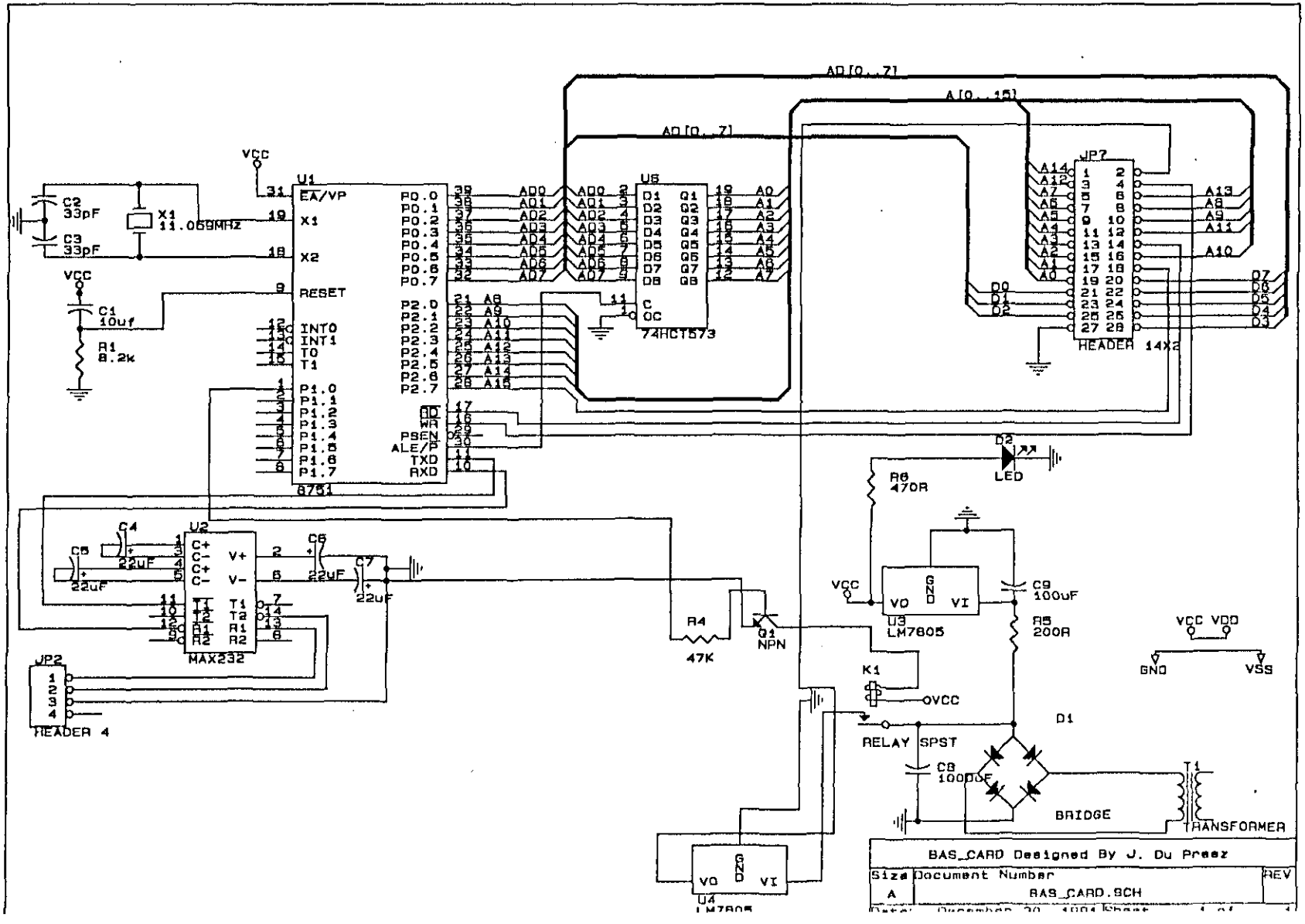
**DRAWINGS** **Page**

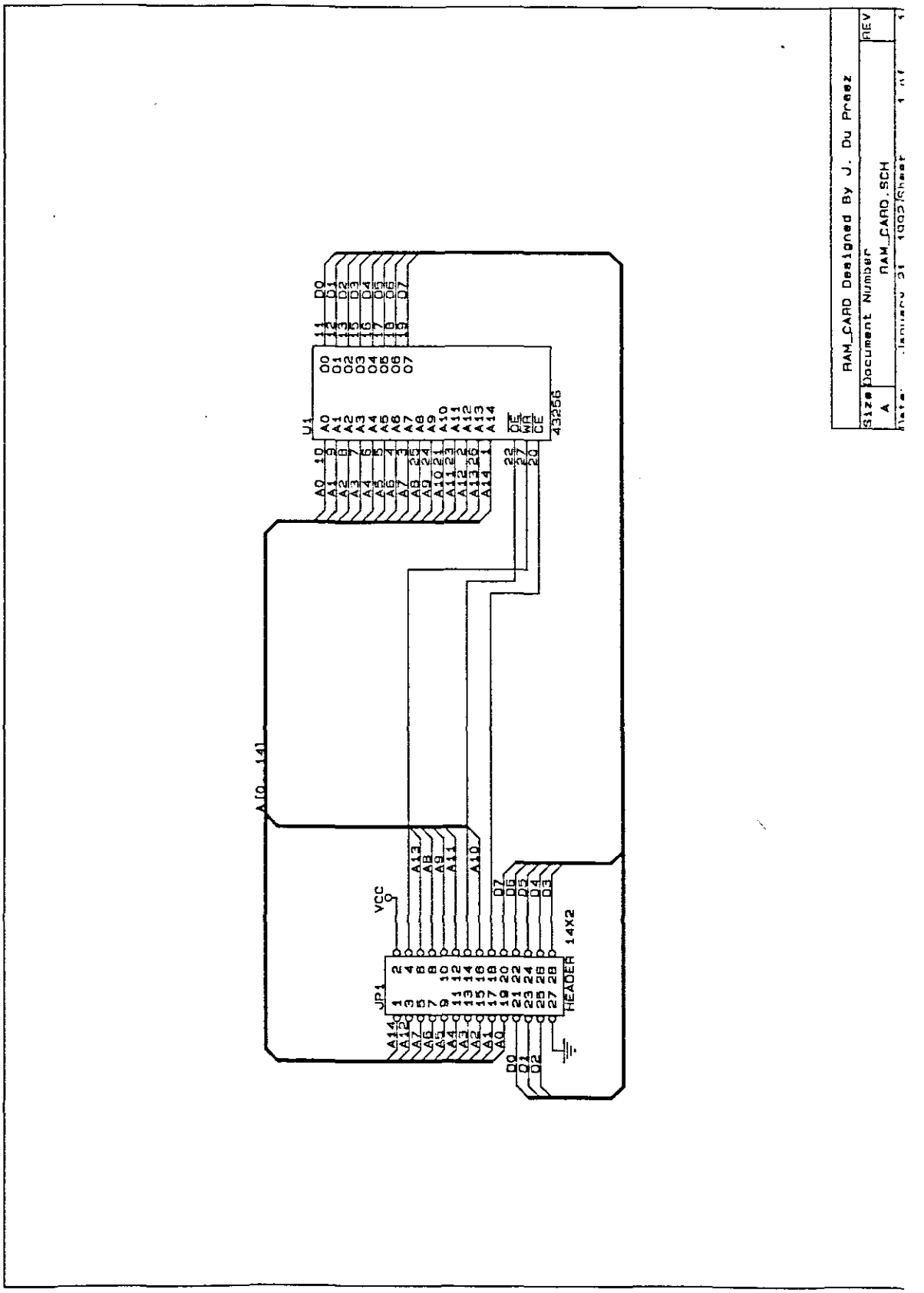


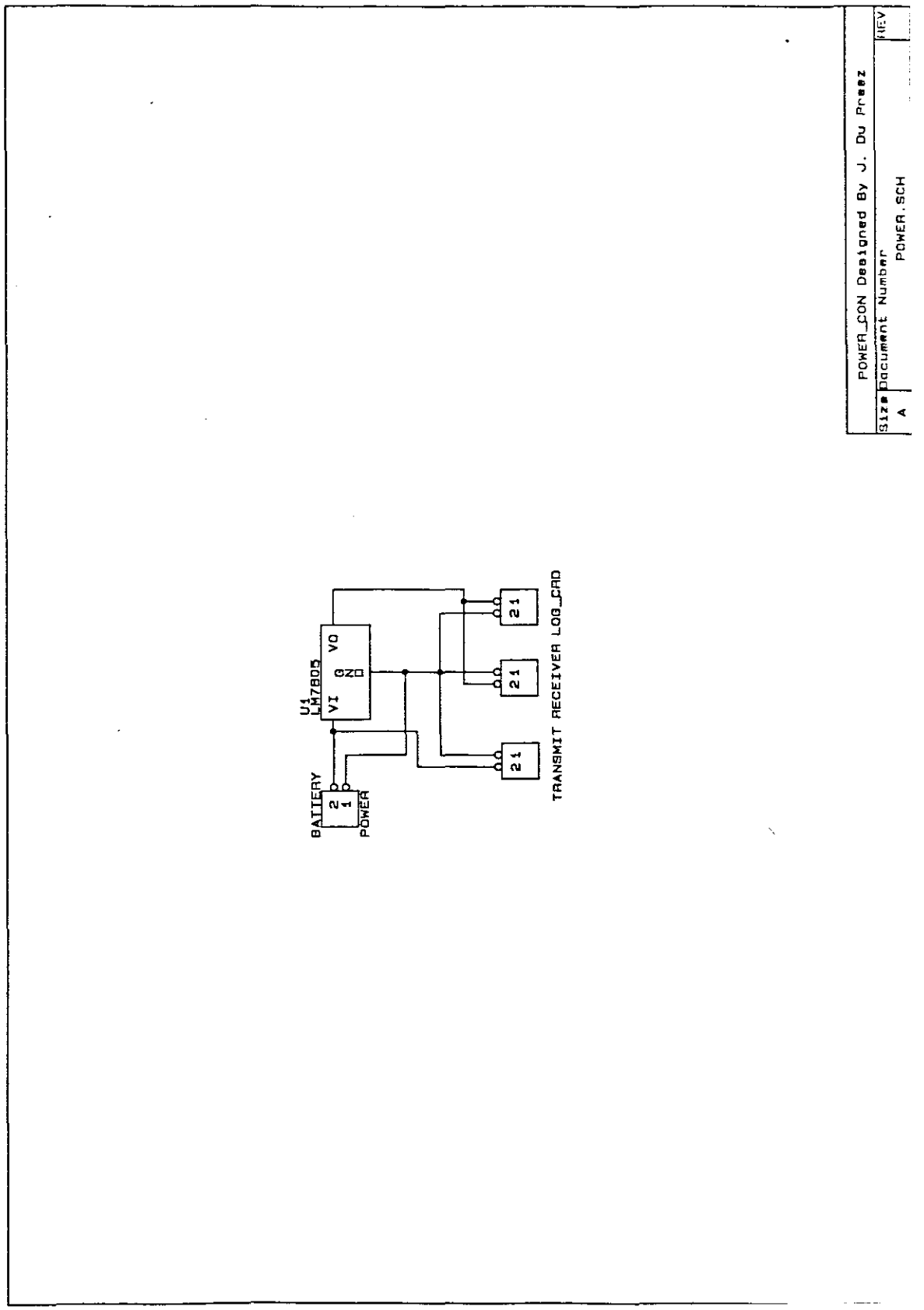

| INFRA-RED TRANSMITTER | | |
|---|---|---|
| Designed By J. Du Preez | | |
| Size A | Document Number TRANSMIT.SCH | REV |
| Date: December 30, 1991 | Sheet 1 of 1 | |


C11
15p
U3
SL486
C17
150n
C16
22uF
C15
10uF
C18
n10
D4
1N914S
R6
180K
Q2
C19
1uF
BC 549B
C20
4.7N
R8
22K
D5
LED
R7
10K
D3
I.R. DIODE
C12
6.8uF
C14
C13
22n
4.7n
R5
50R
5V
U1B
CEXT
C22
4.7uF
R9
150K
REXT/CEXT
A
B
CLR
Q
Q
74HC123
U1A
CEXT
C21
4.7uF
R10
150K
REXT/CEXT
A
B
CLR
Q
Q
74HC123
JP2
HEADER 2
JP1
HEADER 2
C1
15p
U3
SL486
C7
150n
C6
22uF
C5
10uF
C8
n10
D2
1N914S
R2
180K
Q1
C9
1uF
BC 549B
C10
4.7N
R4
22K
D6
LED
R3
10K
D1
I.R. DIODE
C2
6.8uF
C4
C3
22n
4.7n
R1
50R
VCC
5V
INFRA-RED RECEIVER
Designed By J. Du Preez
Size
A
Document Number
RECEIVER.SCH
REV
Date: December 30, 1991 Sheet 1 of 1


LOG_MEM.SCH
AD[0..7]
A[0..15]
RD
WR
CS CLOCK
D[0..7]
VCC
U2
8751
EA/VP
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
PSEN
ALE/P
TXD
RXD
X1
X2
C2
33pF
C3
33pF
X1
11.059MHz
C1
10uf
R1
8.2k
U1
74HC573
U3
74HC138
U4A
U4B
U4C
U4D
74HC00
U5
74C922
R3
RESISTOR VAR
JP1
HEADER 14
JP2
HEADER 7
JP3
HEADER 2
JP4
HEADER 2
C4
10n
C5
100n
A[13..15]
VCC VDD
GND VSS
LOG_CARD Designed By J. Du Preez
Size A
Document Number
LOG_CARD.SCH
REV
Date: December 30, 1991 Sheet 1 of 2


AD[0..7]
A[0..15]
AD[0..7]
VCC
U1
EA/VP
X1
X2
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
RD
WR
PSEN
ALE/P
TXD
RXD
8751
C2
33pF
C3
33pF
X1
11.059MHz
C1
10uf
R1
8.2k
U6
74HCT573
JP7
HEADER 14X2
D2
LED
R6
470R
U2
MAX232
C4
22uF
C5
22uF
C6
22uF
C7
22uF
JP2
HEADER 4
R4
47K
Q1
NPN
U3
LM7805
C9
100uF
R5
200R
K1
RELAY SPST
D1
BRIDGE
C8
1000uF
T1
TRANSFORMER
U4
LM7805
GND
VSS
VDD
BAS_CARD Designed By J. Du Preez
Size
A
Document Number
BAS_CARD.SCH
REV


RAM_CARD Designed By J. Du Preez
Size
A
Document Number
RAM_CARD.SCH
REV
U1
43256
JP1
HEADER 14X2
VCC
A[0..14]


POWER_CON Designed By J. Du Preez
Size Document Number REV
A POWER.SCH
U1
LM7805
VI
G
VO
BATTERY
2
1
POWER
21
21
21
TRANSMIT RECEIVER LOG_CRD

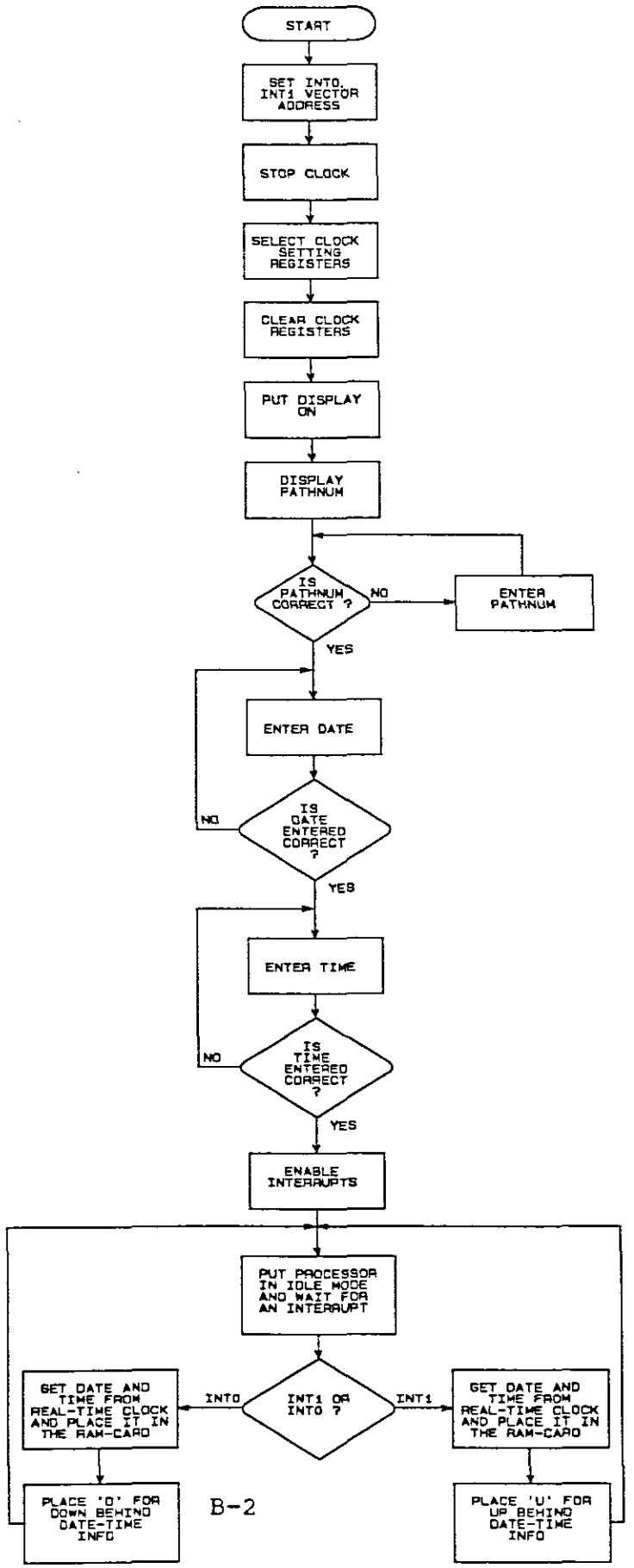
# APPENDIX B

## LOG-CARD PROGRAM LISTING

The assembler listing of the LOG-CARD program follows on page B3. It is inevitable that this program will be changed or added to, to correct any weaknesses found when the card is used more extensively.

### Flow Diagrams

The flow diagram of the LOG-CARD program is also included and is shown on page B2.


START
SET INT0, INT1 VECTOR ADDRESS
STOP CLOCK
SELECT CLOCK SETTING REGISTERS
CLEAR CLOCK REGISTERS
PUT DISPLAY ON
DISPLAY PATHNUM
IS PATHNUM CORRECT ?
NO
ENTER PATHNUM
YES
ENTER DATE
IS DATE ENTERED CORRECT ?
NO
YES
ENTER TIME
IS TIME ENTERED CORRECT ?
NO
YES
ENABLE INTERRUPTS
PUT PROCESSOR IN IDLE MODE AND WAIT FOR AN INTERRUPT
INT1 OR INT0 ?
INT0
INT1
GET DATE AND TIME FROM REAL-TIME CLOCK AND PLACE IT IN THE RAM-CARD
GET DATE AND TIME FROM REAL-TIME CLOCK AND PLACE IT IN THE RAM-CARD
PLACE 'D' FOR DOWN BEHIND DATE-TIME INFO
PLACE 'U' FOR UP BEHIND DATE-TIME INFO

MCS-51 MACRO ASSEMBLER LOG-CARD 23/12/91

```
LOC  OBJ       LINE                SOURCE

                1                  NAME    LOG-CARD
REG             2                  ACCUM   EQU A
                3
0087            4                  PCON    DATA 87H
0020            5                  CNTR    DATA 20H
0021            6                  CNTR1   DATA 21H
                7
0020            8                  IDLE BIT  20H
                9
0000           10                  ORG     00H             ;START PROGRAM
0000 020030    11                  LJMP    START
               12
0003           13                  ORG     03H             ;VECTOR ADDRESS INT 0
0003 C0D0      14                  PUSH    PSW
0005 020360    15                  LJMP    INTERP0
               16
0013           17                  ORG     13H             ;VECTOR ADDRESS INT 1
0013 C0D0      18                  PUSH    PSW
0015 02038A    19                  LJMP    INTERP1
               20
0030           21                  ORG     30H
               22
0030 753000    23     START:       MOV     30H,#00H        ;STARTING ADDRESS TO STORE DATA
0033 753102    24                  MOV     31H,#02H
0036 753280    25                  MOV     32H,#80H        ;REAL_TIME CLOCK ADDRESS
0039 753302    26                  MOV     33H,#02H        ;REAL_TIME CLOCK ADDRESS
003C 75340D    27                  MOV     34H,#0DH        ;REAL_TIME CLOCK ADDRESS
003F 753500    28                  MOV     35H,#00H        ;FIRST ID ADDRESS
0042 753600    29                  MOV     36H,#00H        ;SECOND ID ADDRESS
               30
0045 740F      31                  MOV     A,#0FH          ;STOP CLOCK
0047 908000    32                  MOV     DPTR,#8000H
004A F0        33                  MOVX    @DPTR,A
               34
004B 7400      35                  MOV     A,#00H          ;TO ENSURE NO INTERRUPTS ARE PROGRAMMED
004D 90800F    36                  MOV     DPTR,#800FH
0050 F0        37                  MOVX    @DPTR,A
               38
0051 7405      39                  MOV     A,#05H          ;SELECT CLOCK SETTING REGISTERS
0053 908000    40                  MOV     DPTR,#8000H
0056 F0        41                  MOVX    @DPTR,A
               42
0057 7401      43                  MOV     A,#01H          ;SELECT 24HOUR MODE
0059 90800F    44                  MOV     DPTR,#800FH
005C F0        45                  MOVX    @DPTR,A
               46
005D 780D      47                  MOV     R0,#0DH         ;LOAD CLOCK REGS WITH 00
```

```
LOC  OBJ      LINE              SOURCE

005F 7400      48               MOV     A,#00H
0061 853283    49               MOV     DPH,32H
0064 853382    50               MOV     DPL,33H
0067 F0        51    B1:        MOVX    @DPTR,A
0068 A3        52               INC     DPTR
0069 D8FC      53               DJNZ    R0,B1
               54
006B 908000    55               MOV     DPTR,#8000H     ;START CLOCK
006E F0        56               MOVX    @DPTR,A
               57
006F 7438      58               MOV     A,#38H          ;FUNCTION SET_UP FOR DISPLAY
0071 1202AD    59               LCALL   LCD
               60
0074 7406      61               MOV     A,#06H          ;SET ENTRY MODE FOR DISPLAY
0076 1202AD    62               LCALL   LCD
               63
0079 740E      64               MOV     A,#0EH          ;PUT DISPLAY ON WITH CHARACTERS AT
007B 1202AD    65               LCALL   LCD             ;CURSOR NOT BLINKING
               66
007E 7401      67               MOV     A,#01H          ;CLEAR DISPLAY
0080 1202AD    68               LCALL   LCD
0083 900426    69               MOV     DPTR,#MSG4      ;DISPLAY " PATH= "
0086 12027A    70               LCALL   MSG_POINTER
0089 7485      71               MOV     A,#85H          ;PLACE CURSOR
008B 1202AD    72               LCALL   LCD
               73
008E 853582    74               MOV     DPL,35H
0091 853683    75               MOV     DPH,36H
0094 E0        76               MOVX    A,@DPTR
0095 A3        77               INC     DPTR
0096 858336    78               MOV     36H,DPH
0099 858235    79               MOV     35H,DPL
009C 120299    80               LCALL   DISPLAY         ;DISPLAY FIRST ID CHARACTER
               81
009F 7486      82               MOV     A,#86H          ;PLACE CURSOR
00A1 1202AD    83               LCALL   LCD
00A4 853582    84               MOV     DPL,35H
00A7 853683    85               MOV     DPH,36H
00AA E0        86               MOVX    A,@DPTR
00AB 858336    87               MOV     36H,DPH
00AE 858235    88               MOV     35H,DPL
00B1 120299    89               LCALL   DISPLAY         ;DISPLAY SECOND ID CHARACTER
00B4 74C1      90               MOV     A,#0C1H         ;PLACE CURSOR
00B6 1202AD    91               LCALL   LCD
00B9 90040F    92               MOV     DPTR,#MSG2      ;DISPLAY " CHG \ ENT "
00BC 12027A    93               LCALL   MSG_POINTER
               94
00BF 1202C1    95    D20:       LCALL   KEY
00C2 B43B11    96               CJNE    A,#3BH,D10
00C5 7401      97               MOV     A,#01H          ;CLEAR DISPLAY
```

```
LOC  OBJ      LINE              SOURCE

00C7 1202AD    98               LCALL   LCD
00CA 90042C    99               MOV     DPTR,#MSG5      ;DISPLAY " !!PLEASE WAIT!!
00CD 12027A   100               LCALL   MSG_POINTER
00D0 120331   101               LCALL   ERASE           ;ERASE RAM-CARD
00D3 0200E1   102               LJMP    PATH
00D6 B43AE6   103   D10:        CJNE    A,#3AH,D20
00D9 740F     104               MOV     A,#0FH          ;CHARACTERS BLINKING AT CURSOR
00DB 1202AD   105               LCALL   LCD
00DE 020187   106               LJMP    DATE
              107
00E1 753500   108   PATH:       MOV     35H,#00H        ;ID ADDRESS IN FLASH MEMORY
00E4 753600   109               MOV     36H,#00H
00E7 7401     110               MOV     A,#01H          ;CLEAR DISPLAY
00E9 1202AD   111               LCALL   LCD
00EC 9003F1   112               MOV     DPTR,#MSG       ;DISPLAY " ENTER PATHNUM "
00EF 12027A   113               LCALL   MSG_POINTER
              114
00F2 752102   115               MOV     CNTR1,#02H
00F5 75201B   116   A50:        MOV     CNTR,#1BH
00F8 900448   117               MOV     DPTR,#LOOK_UP
00FB E4       118   A30:        CLR      A
00FC 93       119               MOVC    A,@A+DPTR
00FD F9       120               MOV     R1,A
00FE A3       121               INC     DPTR
00FF D52003   122               DJNZ    CNTR,A40
0102 0200F5   123               LJMP    A50
0105 120299   124   A40:        LCALL   DISPLAY
0108 7410     125               MOV     A,#10H          ;MOV CURSOR TO THE LEFT
010A 1202AD   126               LCALL   LCD
010D 1202C1   127   A20:        LCALL   KEY             ;ENTER PATH ID
0110 B43B03   128               CJNE    A,#3BH,A10
0113 0200FB   129               LJMP    A30
0116 B43AF4   130   A10:        CJNE    A,#3AH,A20
0119 853582   131               MOV     DPL,35H
011C 853683   132               MOV     DPH,36H
011F 1203EE   133               LCALL   MEMWR
0122 A3       134               INC     DPTR
0123 858336   135               MOV     36H,DPH
0126 858235   136               MOV     35H,DPL
0129 7414     137               MOV     A,#14H          ;SHIFT CURSOR TO THE RIGHT
012B 1202AD   138               LCALL   LCD
012E D521C4   139               DJNZ    CNTR1,A50
0131 7401     140               MOV     A,#01H          ;CLEAR DISPLAY
0133 1202AD   141               LCALL   LCD
0136 900426   142               MOV     DPTR,#MSG4      ;DISPLAY " PATH= "
0139 12027A   143               LCALL   MSG_POINTER
013C 7485     144               MOV     A,#85H          ;PLACE CURSOR
013E 1202AD   145               LCALL   LCD
0141 853582   146               MOV     DPL,35H
0144 853683   147               MOV     DPH,36H
```

```
LOC  OBJ      LINE                SOURCE

0147 1582     148                 DEC     DPL
0149 1582     149                 DEC     DPL
014B E0       150                 MOVX    A,@DPTR
014C A3       151                 INC     DPTR
014D 858336   152                 MOV     36H,DPH
0150 858235   153                 MOV     35H,DPL
0153 120299   154                 LCALL   DISPLAY         ;DISPLAY FIRST ID ENTERED
              155
0156 7486     156                 MOV     A,#86H
0158 1202AD   157                 LCALL   LCD
015B 853582   158                 MOV     DPL,35H
015E 853683   159                 MOV     DPH,36H
0161 E0       160                 MOVX    A,@DPTR
0162 858336   161                 MOV     36H,DPH
0165 858235   162                 MOV     35H,DPL
0168 120299   163                 LCALL   DISPLAY         ;DISPLAY SECOND ID ENTERED
              164
016B 74C1     165                 MOV     A,#0C1H         ;PLACE CURSOR
016D 1202AD   166                 LCALL   LCD
0170 90040F   167                 MOV     DPTR,#MSG2      ;DISPLAY " CHG \ ENT "
0173 12027A   168                 LCALL   MSG_POINTER
0176 1202C1   169   A60:          LCALL   KEY
0179 B43B03   170                 CJNE    A,#3BH,A70
017C 0200E1   171                 LJMP    PATH
017F B43AF4   172   A70:          CJNE    A,#3AH,A60
0182 740F     173                 MOV     A,#0FH          ;CHARACTERS BLINKING AT CURSOR
0184 1202AD   174                 LCALL   LCD
              175
0187 7401     176   DATE:         MOV     A,#01H          ;CLEAR DISPLAY
0189 1202AD   177                 LCALL   LCD
018C 900400   178                 MOV     DPTR,#MSG1      ;DISPLAY " DATE: YY/MM/DD "
018F 12027A   179                 LCALL   MSG_POINTER
              180
0192 7405     181                 MOV     A,#05H          ;STOP CLOCK FOR SETTING
0194 908000   182                 MOV     DPTR,#8000H
0197 F0       183                 MOVX    @DPTR,A
              184
0198 7806     185                 MOV     R0,#06H
019A 7902     186                 MOV     R1,#02H
019C 7A04     187                 MOV     R2,#04H
019E 90800D   188                 MOV     DPTR,#800DH
01A1 7486     189                 MOV     A,#86H          ;PLACE CURSOR AT FIRST Y-ENTRY
01A3 1202AD   190                 LCALL   LCD
01A6 1202C1   191   NEW_D:        LCALL   KEY             ;WAIT FOR CHARACTER TO BE ENTERED
01A9 B43003   192                 CJNE    A,#30H,NOT_EQ1  ;IF CHARACTER ENTERED < 0 ENTER AGAIN
01AC 0201BB   193                 JMP     CORR1
01AF 40F5     194   NOT_EQ1:      JC      NEW_D
01B1 B43903   195                 CJNE    A,#39H,NOT_EQ2  ;IF CHARACTER ENTERED > 9 ENTER AGAIN
01B4 0201BB   196                 JMP     CORR1
01B7 4002     197   NOT_EQ2:      JC      CORR1
```

```
LOC  OBJ      LINE                SOURCE

01B9 80EB     198                 JMP     NEW_D
01BB F0       199     CORR1:      MOVX    @DPTR,A         ;IF CHARACTER IS VALLID THEN DISPLAY
01BC 120299   200                 LCALL   DISPLAY
01BF 1582     201                 DEC     DPL
01C1 D905     202                 DJNZ    R1,B10
01C3 74C1     203                 MOV     A,#0C1H         ;PLACE CURSOR ON FIRST M-ENTRY
01C5 1202AD   204                 LCALL   LCD
01C8 DA05     205     B10:        DJNZ    R2,B20
01CA 74C4     206                 MOV     A,#0C4H         ;PLACE CURSOR ON FIRST D-ENTRY
01CC 1202AD   207                 LCALL   LCD
01CF D8D5     208     B20:        DJNZ    R0,NEW_D
              209
01D1 7401     210                 MOV     A,#01H          ;CLEAR DISPLAY
01D3 1202AD   211                 LCALL   LCD
01D6 90800D   212                 MOV     DPTR,#800DH
01D9 1202EF   213                 LCALL   READ_DATE       ;READ DATE FOR VERIFICATION
01DC 74C1     214                 MOV     A,#0C1H         ;PLACE CURSOR
01DE 1202AD   215                 LCALL   LCD
01E1 90040F   216                 MOV     DPTR,#MSG2      ;DISPLAY " CHG / ENT "
01E4 12027A   217                 LCALL   MSG_POINTER
01E7 1202C1   218     B40:        LCALL   KEY             ;VERIFY IF 'E' OR 'C' IS ENTERED
01EA B43803   219                 CJNE    A,#3BH,B30      ;IF 'C' THEN ENTER DATE AGAIN
01ED 020187   220                 LJMP    DATE
01F0 B43AF4   221     B30:        CJNE    A,#3AH,B40      ;IF 'E' THEN ENTER TIME
              222
01F3 7401     223     TIME:       MOV     A,#01H          ;CLEAR DISPLAY
01F5 1202AD   224                 LCALL   LCD
01F8 900417   225                 MOV     DPTR,#MSG3      ;DISPLAY " TIME: **/**/** "
01FB 12027A   226                 LCALL   MSG_POINTER
              227
01FE 7806     228                 MOV     R0,#06H
0200 7902     229                 MOV     R1,#02H
0202 7A04     230                 MOV     R2,#04H
0204 908007   231                 MOV     DPTR,#8007H
0207 7486     232                 MOV     A,#86H          ;PLACE CURSOR AT FIRST *-ENTRY
0209 1202AD   233                 LCALL   LCD
020C 1202C1   234     NEW_T:      LCALL   KEY             ;WAIT FOR CHARACTER TO BE ENTERED
020F B43003   235                 CJNE    A,#30H,NOT_EQ3  ;IF CHARACTER ENTERED < 1 ENTER AGAIN
0212 020221   236                 JMP     CORR2
0215 40F5     237     NOT_EQ3:    JC      NEW_T
0217 B43903   238                 CJNE    A,#39H,NOT_EQ4  ;IF CHARACTER ENTERED > 9 ENTER AGAIN
021A 020221   239                 JMP     CORR2
021D 4002     240     NOT_EQ4:    JC      CORR2
021F 80EB     241                 JMP     NEW_T
0221 F0       242     CORR2:      MOVX    @DPTR,A         ;IF CHARACTER IS VALLID THEN DISPLAY
0222 120299   243                 LCALL   DISPLAY
0225 1582     244                 DEC     DPL
0227 D905     245                 DJNZ    R1,B50
0229 74C1     246                 MOV     A,#0C1H         ;PLACE CURSOR ON THIRD *-ENTRY
022B 1202AD   247                 LCALL   LCD
```

```
LOC  OBJ      LINE                SOURCE

022E DA05     248    B50:         DJNZ    R2,B60
0230 74C4     249                 MOV     A,#0C4H         ;PLACE CURSOR ON FIFTH *-ENTRY
0232 1202AD   250                 LCALL   LCD
0235 D8D5     251    B60:         DJNZ    R0,NEW_T
              252
0237 7401     253                 MOV     A,#01H          ;CLEAR DISPLAY
0239 1202AD   254                 LCALL   LCD
023C 908007   255                 MOV     DPTR,#8007H
023F 120310   256                 LCALL   READ_TIME       ;READ TIME FOR VERIFICATION
0242 74C1     257                 MOV     A,#0C1H         ;PLACE CURSOR
0244 1202AD   258                 LCALL   LCD
0247 90040F   259                 MOV     DPTR,#MSG2      ;DISPLAY " CHG / ENT "
024A 12027A   260                 LCALL   MSG_POINTER
024D 1202C1   261    B70:         LCALL   KEY             ;VERIFY IF 'E' OR 'C' IS ENTERED
0250 B43B03   262                 CJNE    A,#3BH,B80      ;IF 'C' THEN ENTER TIME AGAIN
0253 0201F3   263                 LJMP    TIME
0256 B43AF4   264    B80:         CJNE    A,#3AH,B70      ;IF 'E' THEN ENTER IDLE MODE
              265
0259 7401     266                 MOV     A,#01H          ;CLEAR DISPLAY
025B 1202AD   267                 LCALL   LCD
              268
025E 7408     269                 MOV     A,#08H          ;SWITCH DISPLAY OFF
0260 1202AD   270                 LCALL   LCD
              271
0263 7400     272                 MOV     A,#00H          ;START CLOCK
0265 908000   273                 MOV     DPTR,#8000H
0268 F0       274                 MOVX    @DPTR,A
              275
0269 D2AF     276                 SETB    EA              ;ENABLE INTERRUPT REG
026B D2A8     277                 SETB    EX0             ;ENABLE INT 0
026D D2AA     278                 SETB    EX1             ;ENABLE INT 1
026F D2B2     279                 SETB    P3.2            ;ENABLE INT 0 PIN
0271 D2B3     280                 SETB    P3.3            ;ENABLE INT 1 PIN
0273 D288     281                 SETB    IT0             ;INT 0 IS EDGE ACTIVATED
0275 D28A     282                 SETB    IT1             ;INT 1 IS EDGE ACTIVATED
              283
              284
              285
              286
              287
              288
0277 020352   289                 LJMP    BEGIN
              290
027A 7809     291    MSG_POINTER: MOV     R0,#09H         ;DISPLAY MESSAGES ON THE LCD_DISPLAY
027C E4       292    MESSAGE:     CLR     A
027D 93       293                 MOVC    A,@A+DPTR
027E 6018     294                 JZ      EXIT
0280 C082     295                 PUSH    DPL
0282 C083     296                 PUSH    DPH
0284 C0E0     297                 PUSH    ACC
```

```
LOC  OBJ       LINE              SOURCE

0286 D805      298               DJNZ    R0,B100
0288 74C0      299               MOV     A,#0C0H         ;PLACE CURSOR AT THE BEGINNING
028A 1202AD    300               LCALL   LCD             ;OF SECOND DISPLAY
028D D0E0      301   B100:       POP     ACC
028F 5199      302               ACALL   DISPLAY
0291 D083      303               POP     DPH
0293 D082      304               POP     DPL
0295 A3        305               INC     DPTR
0296 417C      306               AJMP    MESSAGE
0298 22        307   EXIT:       RET
               308
0299 C083      309   DISPLAY:    PUSH    DPH             ;DISPLAY CHARACTERS ON THE LCD_DISPLAY
029B C082      310               PUSH    DPL
029D 90C000    311               MOV     DPTR,#0C000H
02A0 D291      312               SETB    P1.1
02A2 C292      313               CLR     P1.2
02A4 F0        314               MOVX    @DPTR,A
02A5 12033E    315               LCALL   DELAY
02A8 D082      316               POP     DPL
02AA D083      317               POP     DPH
02AC 22        318               RET
               319
02AD C083      320    LCD:       PUSH    DPH
02AF C082      321               PUSH    DPL
02B1 90C000    322               MOV     DPTR,#0C000H    ;PLACE CURSOR AT AN ADDRESS POSITION
02B4 C291      323               CLR     P1.1
02B6 C292      324               CLR     P1.2
02B8 F0        325               MOVX    @DPTR,A
02B9 12033E    326               LCALL   DELAY
02BC D082      327               POP     DPL
02BE D083      328               POP     DPH
02C0 22        329               RET
               330
02C1 C083      331   KEY:        PUSH    DPH             ;ENTER CHARACTERS ON KEY_PAD
02C3 C082      332               PUSH    DPL
02C5 90A000    333               MOV     DPTR,#0A000H
02C8 3090FD    334               JNB     P1.0,$          ;WAIT TILL KEY IS PRESSED
02CB 120348    335               LCALL   DEBOUNCE        ;TO PREVENT KEY DEBOUNCING
02CE 2090FD    336               JB      P1.0,$          ;WAIT TILL KEY IS RELEASED
02D1 E0        337               MOVX    A,@DPTR
02D2 540F      338               ANL     A,#0FH          ;MASKED THE KEY PRESSED FOR DISPLAY
02D4 FE        339               MOV     R6,A            ;SEARCH IN LOOKUP TABLE FOR CHARACTER
02D5 7F00      340               MOV     R7,#00H         ;ENTERED
02D7 90043C    341               MOV     DPTR,#KEY_LOOKUP
02DA E4        342   LABEL_5:    CLR     A
02DB 93        343               MOVC    A,@A+DPTR
02DC B50603    344               CJNE    A,06H,LABEL_2
02DF 0202E7    345               LJMP    LABEL_6
02E2 0F        346   LABEL_2:    INC     R7
02E3 A3        347               INC     DPTR
```

```
LOC  OBJ       LINE                SOURCE

02E4 0202DA    348                 LJMP    LABEL_5
02E7 EF        349    LABEL_6:     MOV     A,R7
02E8 6430      350                 XRL     A,#30H
02EA D082      351                 POP     DPL
02EC D083      352                 POP     DPH
02EE 22        353                 RET
               354
02EF 7806      355    READ_DATE:   MOV     R0,#06H        ;READ DATE FROM REAL TIME_CLOCK
02F1 7902      356                 MOV     R1,#02H        ;FOR VERIFICATION
02F3 7A04      357                 MOV     R2,#04H        ;DISPLAY THE DATE ON THE LCD_DISPLAY
02F5 E0        358    B45:         MOVX    A,@DPTR
02F6 540F      359                 ANL     A,#0FH
02F8 6430      360                 XRL     A,#30H
02FA 120299    361                 LCALL   DISPLAY
02FD 1582      362                 DEC     DPL
02FF D905      363                 DJNZ    R1,B55
0301 742F      364                 MOV     A,#2FH
0303 120299    365                 LCALL   DISPLAY
0306 DA05      366    B55:         DJNZ    R2,B65
0308 742F      367                 MOV     A,#2FH
030A 120299    368                 LCALL   DISPLAY
030D D8E6      369    B65:         DJNZ    R0,B45
030F 22        370                 RET
               371
0310 7806      372    READ_TIME:   MOV     R0,#06H        ;READ TIME FROM REAL TIME_CLOCK
0312 7902      373                 MOV     R1,#02H        ;FOR VERIFICATION
0314 7A04      374                 MOV     R2,#04H        ;DISPLAY THE TIME ON THE LCD_DISPLAY
0316 E0        375    B75:         MOVX    A,@DPTR
0317 540F      376                 ANL     A,#0FH
0319 6430      377                 XRL     A,#30H
031B 120299    378                 LCALL   DISPLAY
031E 1582      379                 DEC     DPL
0320 D905      380                 DJNZ    R1,B85
0322 743A      381                 MOV     A,#3AH
0324 120299    382                 LCALL   DISPLAY
0327 DA05      383    B85:         DJNZ    R2,B95
0329 743A      384                 MOV     A,#3AH
032B 120299    385                 LCALL   DISPLAY
032E D8E6      386    B95:         DJNZ    R0,B75
0330 22        387                 RET
               388
0331 900000    389    ERASE:       MOV     DPTR,#0000H
334 742E       390    ER_MORE:     MOV     A,#2EH
0336 F0        391                 MOVX    @DPTR,A
0337 A3        392                 INC     DPTR
0338 E583      393                 MOV     A,DPH
033A B450F7    394                 CJNE    A,#80,ER_MORE
033D 22        395                 RET
               396
033E 7C40      397    DELAY:       MOV     R4,#40H        ;DELAY SUBROUTINE
```

```
LOC  OBJ      LINE                SOURCE

0340 74FF     398    AGAIN:       MOV     A,#0FFH
0342 14       399    PAUSE:       DEC     A
0343 70FD     400                 JNZ     PAUSE
0345 DCF9     401                 DJNZ    R4,AGAIN
0347 22       402                 RET
              403
0348 7C40     404    DEBOUNCE:    MOV     R4,#40H         ;DEBOUNCE SUBROUTINE
034A 74FF     405    DEB1:        MOV     A,#0FFH
034C 14       406    DEB2:        DEC     A
034D 70FD     407                 JNZ     DEB2
034F DCF9     408                 DJNZ    R4,DEB1
0351 22       409                 RET
              410
0352 7800     411    BEGIN:       MOV     R0,#00H
0354 7900     412                 MOV     R1,#00H
0356 D220     413    ID:          SETB    IDLE            ;MICROPROSESSOR INTO IDLE MODE
0358 438701   414                 ORL     PCON,#1
035B 2020FD   415                 JB      IDLE,$
035E 80F6     416                 JMP     ID
              417
0360 C2AF     418    INTERP0:     CLR     EA
0362 1203C1   419                 LCALL   COUNT
0365 853083   420                 MOV     DPH,30H
0368 853182   421                 MOV     DPL,31H
036B 7444     422                 MOV     A,#44H
036D F0       423                 MOVX    @DPTR,A
036E A3       424                 INC     DPTR
036F 858231   425                 MOV     31H,DPL
0372 858330   426                 MOV     30H,DPH
0375 D0D0     427                 POP     PSW
0377 C220     428                 CLR     IDLE
0379 E588     429    CHK0:        MOV     A,88H
037B 5402     430                 ANL     A,#02H
037D B400F9   431                 CJNE    A,#00H,CHK0
0380 1203B4   432                 LCALL   DELAY2
0383 C28B     433                 CLR     IE1
0385 C289     434                 CLR     IE0
0387 D2AF     435                 SETB    EA
0389 32       436                 RETI
              437
038A C2AF     438    INTERP1:     CLR     EA
038C 1203C1   439                 LCALL   COUNT
038F 853083   440                 MOV     DPH,30H
0392 853182   441                 MOV     DPL,31H
0395 7455     442                 MOV     A,#55H
0397 F0       443                 MOVX    @DPTR,A
0398 A3       444                 INC     DPTR
0399 858231   445                 MOV     31H,DPL
039C 858330   446                 MOV     30H,DPH
039F D0D0     447                 POP     PSW
```

```
LOC  OBJ       LINE                SOURCE

03A1 C220      448                 CLR     IDLE
03A3 E588      449     CHK1:       MOV     A,88H
03A5 5408      450                 ANL     A,#08H
03A7 B400F9    451                 CJNE    A,#00H,CHK1
03AA 120384    452                 LCALL   DELAY2
03AD C289      453                 CLR     IE0
03AF C28B      454                 CLR     IE1
03B1 D2AF      455                 SETB    EA
03B3 32        456                 RETI
               457
03B4 7C06      458     DELAY2:     MOV     R4,#06H         ;1,5 SEC DELAY
03B6 7DFF      459     PAUSE2:     MOV     R5,#0FFH
03B8 7EFF      460     AGAIN2:     MOV     R6,#0FFH
03BA DEFE      461                 DJNZ    R6,$
03BC DDFA      462                 DJNZ    R5,AGAIN2
03BE DCF6      463                 DJNZ    R4,PAUSE2
03C0 22        464                 RET
               465
03C1 7808      466     COUNT:      MOV     R0,#08H         ;READ DATE AND TIME FROM REAL TIME CLOCK
03C3 853283    467                 MOV     DPH,32H
03C6 853482    468                 MOV     DPL,34H
03C9 E0        469     B2:         MOVX    A,@DPTR
03CA 540F      470                 ANL     A,#0FH
03CC 6430      471                 XRL     A,#30H
03CE F9        472                 MOV     R1,A
03CF 1582      473                 DEC     DPL
03D1 C083      474                 PUSH    DPH
03D3 C082      475                 PUSH    DPL
03D5 853083    476                 MOV     DPH,30H
03D8 853182    477                 MOV     DPL,31H
03DB 71EE      478                 ACALL   MEMWR
03DD A3        479                 INC     DPTR
03DE 858330    480                 MOV     30H,DPH
03E1 858231    481                 MOV     31H,DPL
03E4 D082      482                 POP     DPL
03E6 D083      483                 POP     DPH
03E8 D8DF      484                 DJNZ    R0,B2
03EA 75340D    485                 MOV     34H,#0DH
03ED 22        486                 RET
               487
               488
03EE E9        489     MEMWR:      MOV     A,R1            ;WRITE BYTE INTO RAM-CARD
03EF F0        490                 MOVX    @DPTR,A
03F0 22        491                 RET
               492
03F1 454E5445493   MSG:            DB 'ENTER PATHNUM:',00H
03F5 52205041
03F9 54484E55
03FD 4D3A
03FF 00
```

```
LOC  OBJ       LINE                SOURCE

0400 44415445494    MSG1:          DB 'DATE: YY/MM/DD',00H
0404 3A205959
0408 2F4D4D2F
040C 4444
040E 00
040F 4348472F495    MSG2:          DB 'CHG/ENT',00H
0413 454E54
0416 00
0417 54494D45496    MSG3:          DB 'TIME: **:**:**',00H
041B 3A202A2A
041F 3A2A2A3A
0423 2A2A
0425 00
0426 50415448497    MSG4:          DB 'PATH=',00H
042A 3D
042B 00
042C 2121504C498    MSG5:          DB '!!PLEASE WAIT!!',00H
0430 45415345
0434 20574149
0438 542121
043B 00
043C 0D       499   KEY_LOOKUP:    DB 0DH,00H,01H,02H,04H,05H,06H,08H,09H,0AH,0CH,0EH
043D 00
043E 01
043F 02
0440 04
0441 05
0442 06
0443 08
0444 09
0445 0A
0446 0C
0447 0E
0448 41       500   LOOK_UP:       DB 'A','B','C','D','E','F','G','H','I','J','K','L','M'
0449 42
044A 43
044B 44
044C 45
044D 46
044E 47
044F 48
0450 49
0451 4A
0452 4B
0453 4C
0454 4D
0455 4E       501                  DB 'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'
0456 4F
0457 50
0458 51
```

```
LOC  OBJ      LINE                SOURCE

0459 52
045A 53
045B 54
045C 55
045D 56
045E 57
045F 58
0460 59
0461 5A
0462 00       502    STOP:        NOP
              503                 END

SYMBOL TABLE LISTING
------ ----- -------


N A M E          T Y P E   V A L U E      A T T R I B U T E S

A10 . . . . . .  C ADDR    0116H    A
A20 . . . . . .  C ADDR    010DH    A
A30 . . . . . .  C ADDR    00F8H    A
A40 . . . . . .  C ADDR    0105H    A
A50 . . . . . .  C ADDR    00F5H    A
A60 . . . . . .  C ADDR    0176H    A
A70 . . . . . .  C ADDR    017FH    A
ACC . . . . . .  D ADDR    00E0H    A
ACCUM . . . . .    REG     ACC
AGAIN . . . . .  C ADDR    0340H    A
AGAIN2. . . . .  C ADDR    0388H    A
B1. . . . . .    C ADDR    0067H    A
B10 . . . . . .  C ADDR    01C8H    A
B100. . . . . .  C ADDR    028DH    A
B2. . . . . .    C ADDR    03C9H    A
B20 . . . . . .  C ADDR    01CFH    A
B30 . . . . . .  C ADDR    01F0H    A
B40 . . . . . .  C ADDR    01E7H    A
B45 . . . . . .  C ADDR    02F5H    A
B50 . . . . . .  C ADDR    022EH    A
B55 . . . . . .  C ADDR    0306H    A
B60 . . . . . .  C ADDR    0235H    A
B65 . . . . . .  C ADDR    030DH    A
B70 . . . . . .  C ADDR    024DH    A
B75 . . . . . .  C ADDR    0316H    A
B80 . . . . . .  C ADDR    0256H    A
B85 . . . . . .  C ADDR    0327H    A
B95 . . . . . .  C ADDR    032EH    A
BEGIN . . . . .  C ADDR    0352H    A
CHK0. . . . . .  C ADDR    0379H    A
CHK1. . . . . .  C ADDR    03A3H    A
CNTR. . . . . .  D ADDR    0020H    A
```

| NAME | TYPE | VALUE | ATTRIBUTES |
|---|---|---|---|
| CNTR1 . . . . . | D ADDR | 0021H | A |
| CORR1 . . . . . | C ADDR | 01B8H | A |
| CORR2 . . . . . | C ADDR | 0221H | A |
| COUNT . . . . . | C ADDR | 03C1H | A |
| D10 . . . . . | C ADDR | 00D6H | A |
| D20 . . . . . | C ADDR | 008FH | A |
| DATE. . . . . | C ADDR | 0187H | A |
| DEB1. . . . . | C ADDR | 034AH | A |
| DEB2. . . . . | C ADDR | 034CH | A |
| DEBOUNCE. . . . | C ADDR | 0348H | A |
| DELAY . . . . . | C ADDR | 033EH | A |
| DELAY2. . . . . | C ADDR | 03B4H | A |
| DISPLAY . . . . | C ADDR | 0299H | A |
| DPH . . . . . | D ADDR | 0083H | A |
| DPL . . . . . | D ADDR | 0082H | A |
| EA. . . . . | B ADDR | 00A8H.7 | A |
| ER_MORE . . . . | C ADDR | 0334H | A |
| ERASE . . . . . | C ADDR | 0331H | A |
| EX0 . . . . . | B ADDR | 00A8H.0 | A |
| EX1 . . . . . | B ADDR | 00A8H.2 | A |
| EXIT. . . . . | C ADDR | 0298H | A |
| ID. . . . . | C ADDR | 0356H | A |
| IDLE. . . . . | B ADDR | 0024H.0 | A |
| IE0 . . . . . | B ADDR | 0088H.1 | A |
| IE1 . . . . . | B ADDR | 0088H.3 | A |
| INTERP0 . . . . | C ADDR | 0360H | A |
| INTERP1 . . . . | C ADDR | 038AH | A |
| IT0 . . . . . | B ADDR | 0088H.0 | A |
| IT1 . . . . . | B ADDR | 0088H.2 | A |
| KEY_LOOKUP. . . | C ADDR | 043CH | A |
| KEY . . . . . | C ADDR | 02C1H | A |
| LABEL_2 . . . . | C ADDR | 02E2H | A |
| LABEL_5 . . . . | C ADDR | 02DAH | A |
| LABEL_6 . . . . | C ADDR | 02E7H | A |
| LCD . . . . . | C ADDR | 02ADH | A |
| LOOK_UP . . . . | C ADDR | 0448H | A |
| MEMWR . . . . . | C ADDR | 03EEH | A |
| MESSAGE . . . . | C ADDR | 027CH | A |
| MSG_POINTER . . | C ADDR | 027AH | A |
| MSG . . . . . | C ADDR | 03F1H | A |
| MSG1. . . . . | C ADDR | 0400H | A |
| MSG2. . . . . | C ADDR | 040FH | A |
| MSG3. . . . . | C ADDR | 0417H | A |
| MSG4. . . . . | C ADDR | 0426H | A |
| MSG5. . . . . | C ADDR | 042CH | A |
| NEW_D . . . . . | C ADDR | 01A6H | A |
| NEW_T . . . . . | C ADDR | 020CH | A |
| NOT_EQ1 . . . . | C ADDR | 01AFH | A |
| NOT_EQ2 . . . . | C ADDR | 01B7H | A |
| NOT_EQ3 . . . . | C ADDR | 0215H | A |

```
NAME             TYPE  VALUE     ATTRIBUTES

NOT_EQ4 . . . .  C ADDR   021DH   A
P1. . . . . .    D ADDR   0090H   A
P3. . . . . .    D ADDR   0080H   A
PATH. . . . . .  C ADDR   00E1H   A
PAUSE . . . . .  C ADDR   0342H   A
PAUSE2. . . . .  C ADDR   0386H   A
PCON. . . . . .  D ADDR   0087H   A
PROGRAM_HANDLER    ----     ----
PSW . . . . . .  D ADDR   00D0H   A
READ_DATE . . .  C ADDR   02EFH   A
READ_TIME . . .  C ADDR   0310H   A
START . . . . .  C ADDR   0030H   A
STOP. . . . . .  C ADDR   0462H   A
TIME. . . . . .  C ADDR   01F3H   A


REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

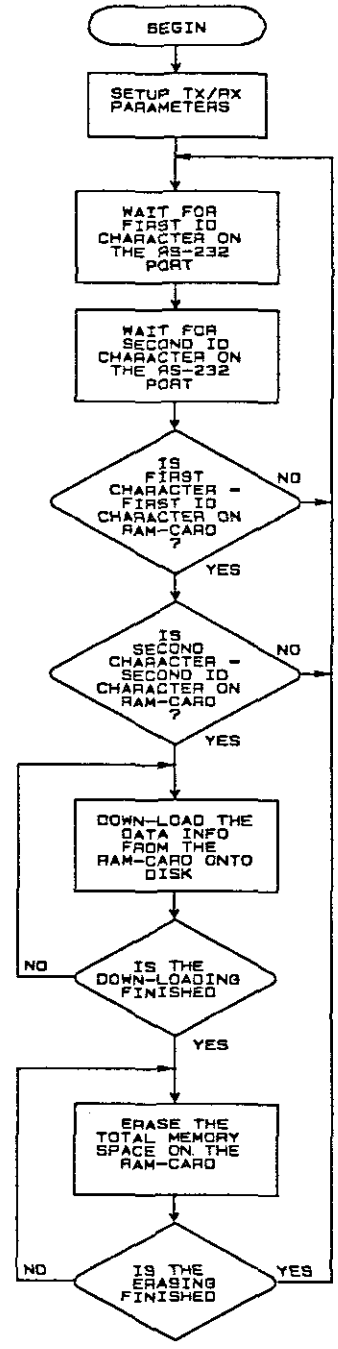
# APPENDIX C

## BAS-CARD PROGRAM LISTING

The assembler listing of the BAS-CARD program follows on page C3.

### Flow Diagrams

The flow diagram of the BAS-CARD program is also included and is shown on page C2.


BEGIN
SETUP TX/RX PARAMETERS
WAIT FOR FIRST ID CHARACTER ON THE RS-232 PORT
WAIT FOR SECOND ID CHARACTER ON THE RS-232 PORT
IS FIRST CHARACTER = FIRST ID CHARACTER ON RAM-CARD ?
NO
YES
IS SECOND CHARACTER = SECOND ID CHARACTER ON RAM-CARD ?
NO
YES
DOWN-LOAD THE DATA INFO FROM THE RAM-CARD ONTO DISK
IS THE DOWN-LOADING FINISHED
NO
YES
ERASE THE TOTAL MEMORY SPACE ON THE RAM-CARD
IS THE ERASING FINISHED
NO
YES

```
MCS-51 MACRO ASSEMBLER    BAS-CARD 18/12/91

LOC  OBJ       LINE                SOURCE

               1                   NAME    PROGRAM_HANDLER
REG            2                   ACCUM   EQU A
0087           3                   PCON    DATA 87H
               4
0000           5                   ORG     00H             ;START PROGRAM
0000 020030    6                   LJMP    START
               7
0030           8                   ORG     30H
               9
0030 C290      10   START:         CLR     P1.0
0032 753000    11                  MOV     30H,#00H        ;RAM-CARD ADDRESS
0035 753100    12                  MOV     31H,#00H
               13
0038 D2AF      14   INIT:          SETB    EA
003A 758DE8    15                  MOV     TH1,#0E8H       ;1200 BAUD TIMER 1
003D 758800    16                  MOV     TL1,#000H       ;TRANSMISSION SET UP
0040 53877F    17                  ANL     PCON,#7FH
0043 758920    18                  MOV     TMOD,#20H
0046 759850    19                  MOV     SCON,#50H
0049 D28E      20                  SETB    TR1
               21
004B C290      22   LOOP1:         CLR     P1.0            ;PUT RAM-CARD OFF
004D 3098FD    23                  JNB     RI,$            ;READ FIRST ID CHARACTER IN
0050 D290      24                  SETB    P1.0            ;PUT RAM-CARD ON
0052 1185      25                  ACALL   DELAY
0054 C298      26                  CLR     RI
0056 E599      27                  MOV     A,SBUF
0058 F9        28                  MOV     R1,A
               29
0059 3098FD    30                  JNB     RI,$            ;READ SECOND ID CHARACTER IN
005C 1185      31                  ACALL   DELAY
005E C298      32                  CLR     RI
0060 E599      33                  MOV     A,SBUF
0062 FA        34                  MOV     R2,A
               35
0063 900000    36                  MOV     DPTR,#0000H     ;COMPARE FIRST ID CHARACTER WITH RAM-CARD ID
0066 E0        37                  MOVX    A,@DPTR
0067 A3        38                  INC     DPTR
0068 69        39                  XRL     A,R1
0069 70E0      40                  JNZ     LOOP1
               41
006B E0        42                  MOVX    A,@DPTR         ;COMPARE SECOND ID CHARACTER WITH RAM-CARD ID
006C 6A        43                  XRL     A,R2
006D 70DC      44                  JNZ     LOOP1
006F 12008F    45                  LCALL   DOWN_L          ;IF ID'S ARE THE SAME THEN DO DOWN_LOADING
               46
0072 3098FD    47   LOOP2:         JNB     RI,$            ;AFTER DOWN_LOADING ERASE FLASH MEMORY
```

```
LOC  OBJ      LINE                SOURCE

0075 1185     48                  ACALL   DELAY
0077 C298     49                  CLR     RI
0079 E599     50                  MOV     A,SBUF
0078 6445     51                  XRL     A,#45H
007D 70F3     52                  JNZ     LOOP2
007F 120081   53                  LCALL   ERASE
0082 020048   54                  LJMP    LOOP1
              55
0085 7AFF     56    DELAY:        MOV     R2,#0FFH
0087 74FF     57    AGAIN:        MOV     A,#0FFH
0089 14       58    PAUSE:        DEC     A
008A 70FD     59                  JNZ     PAUSE
008C DAF9     60                  DJNZ    R2,AGAIN
008E 22       61                  RET
              62
008F 900002   63    DOWN_L:       MOV     DPTR,#0002H     ;DOWN-LOAD PROCEDURE
0092 E0       64    SEND:         MOVX    A,@DPTR         ;TRANSMIT MEMORY CONTENTS
0093 C0E0     65                  PUSH    ACC
0095 642E     66                  XRL     A,#2EH          ;END OF DATA INDICATION
0097 600C     67                  JZ      EXIT1
0099 D0E0     68                  POP     ACC
009B F599     69                  MOV     SBUF,A
009D 3099FD   70                  JNB     TI,$
00A0 C299     71                  CLR     TI
00A2 A3       72                  INC     DPTR
00A3 80ED     73                  JMP     SEND
00A5 D0E0     74    EXIT1:        POP     ACC
00A7 7446     75                  MOV     A,#46H          ;SENT 'F' TO INDICATE DOWN LOADING FINISHED
00A9 F599     76                  MOV     SBUF,A
00AB 3099FD   77                  JNB     TI,$
00AE C299     78                  CLR     TI
00B0 22       79                  RET
              80
              81
00B1 900000   82    ERASE:        MOV     DPTR,#0000H     ;ERASE PROCEDURE
00B4 742E     83    ER_MORE:      MOV     A,#2EH
00B6 F0       84                  MOVX    @DPTR,A
00B7 A3       85                  INC     DPTR
00B8 E583     86                  MOV     A,DPH
00BA B480F7   87                  CJNE    A,#80H,ER_MORE  ;END ADDRESS ?  NO! ERASE MORE
00BD 7446     88                  MOV     A,#46H          ;YES! QUIT
00BF F599     89                  MOV     SBUF,A
00C1 3099FD   90                  JNB     TI,$
00C4 C299     91                  CLR     TI
00C6 22       92                  RET
              93
              94                  END
```

```
SYMBOL TABLE LISTING
------ ----- -------

N A M E          T Y P E  V A L U E      A T T R I B U T E S

ACC . . . . . .  D ADDR   00E0H    A
ACCUM . . . . .    REG    ACC
AGAIN . . . . .  C ADDR   0087H    A
DELAY . . . . .  C ADDR   0085H    A
DOWN_L. . . . .  C ADDR   008FH    A
DPH . . . . . .  D ADDR   0083H    A
EA. . . . . .    B ADDR   00A8H.7  A
ER_MORE . . . .  C ADDR   00B4H    A
ERASE . . . . .  C ADDR   00B1H    A
EXIT1 . . . . .  C ADDR   00A5H    A
INIT. . . . . .  C ADDR   0038H    A
LOOP1 . . . . .  C ADDR   004BH    A
LOOP2 . . . . .  C ADDR   0072H    A
P1. . . . . .    D ADDR   0090H    A
PAUSE . . . . .  C ADDR   0089H    A
PCON. . . . . .  D ADDR   0087H    A
PROGRAM_HANDLER    ----   ----
RI. . . . . .    B ADDR   0098H.0  A
SBUF. . . . . .  D ADDR   0099H    A
SCON. . . . . .  D ADDR   0098H    A
SEND. . . . . .  C ADDR   0092H    A
START . . . . .  C ADDR   0030H    A
TH1 . . . . . .  D ADDR   008DH    A
TI. . . . . .    B ADDR   0098H.1  A
TL1 . . . . . .  D ADDR   008BH    A
TMOD. . . . . .  D ADDR   0089H    A
TR1 . . . . . .  B ADDR   0088H.6  A


REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

# APPENDIX D

## SCREEN-MONITOR PROGRAM LISTING

The listing of the Pascal program follows on page D2.

```
program BASE_CARD;
uses
  dos,crt,printer;

{$L COMMS}                                          {Link the following routines in COMMS.OBJ}

Procedure CommInit ( port : word; params : byte); EXTERNAL;
Function CharReady : Boolean; EXTERNAL;
Function GetByte   : Byte; EXTERNAL;
Procedure SendByte ( theByte : Byte) ; EXTERNAL;

Const baud1200   = $80;                             {Baudrate settings}

      noparity   = $00;                             {Parity checking}

      onestop    = $00;                             {Stop bits}

      len8       = $03;                             {Lentgh of charackter}

      Xon        = ^Q;                              {Flow control characters}
      Xoff       = ^S;

type
  Trail  = record
             Time : string[9];
           end;
const
   HXas : integer = 20;
   HYas : integer = 19;
   Xas  : integer = 33;
   Yas  : integer = 09;
   Item : array[1..4] of string[15] = (' DOWN LOAD ',
                                       ' PRINT     ',
                                       ' DISPLAY   ',
                                       ' QUIT      ');
   Help : array[1..4] of string[44]= ('        To store data on disk     ',
                                      '          To print data           ',
                                      '      To display data on screen   ',
                                      '          To exit to dos          ');

   Lyn  : array[1..6] of char = ('┌','┐','│','└','─','┘');

  Months: array[1..12] of string[5] = ('JAN = ','FEB = ','MAR = ','APR = ',
                                       'MAY = ','JUN = ','JUL = ','AUG = ',
                                       'SEP = ','OCT = ','NOV = ','DEC = ');

  MonthP: array[1..12] of string[4] = ('JAN ','FEB ','MAR ','APR ',
                                       'MAY ','JUN ','JUL ','AUG ',
                                       'SEP ','OCT ','NOV ','DEC ');

  DayP  : array[1..31] of string[2] = ('01','02','03','04',
                                       '05','06','07','08',
```

```
                                    '09','10','11','12',
                                    '13','14','15','16',
                                    '17','18','19','20',
                                    '21','22','23','24',
                                    '25','26','27','28',
                                    '29','30','31');

  DayPh : array[1..31] of string[7] = ('| 01  |','| 02  |','| 03  |','| 04  |',
                                    '| 05  |','| 06  |','| 07  |','| 08  |',
                                    '| 09  |','| 10  |','| 11  |','| 12  |',
                                    '| 13  |','| 14  |','| 15  |','| 16  |',
                                    '| 17  |','| 18  |','| 19  |','| 20  |',
                                    '| 21  |','| 22  |','| 23  |','| 24  |',
                                    '| 25  |','| 26  |','| 27  |','| 28  |',
                                    '| 29  |','| 30  |','| 31  |');


   Print: array[1..3] of string[20] = ('A=MONTHLY PRINTOUT ',
                                    'B=DAILY PRINTOUT',
                                    'C=HOURLY PRINTOUT  ');

   Error: array[0..3] of string[24] = ('!!! PRINTER OFF !!!',
                                    '!!! PRINTER OFF LINE !!!',
                                    '!!! PRINTER BUSY !!!',
                                    '!!! PAPER OUT !!!');

   Days: array[0..6] of string[9] = ('Sunday','Monday','Tuesday','Wednesday',
                                   'Thursday','Friday','Saturday');

  HourP: array[0..23] of string[5] = ('00 = ','01 = ','02 = ','03 = ',
                                    '04 = ','05 = ','06 = ','07 = ',
                                    '08 = ','09 = ','10 = ','11 = ',
                                    '12 = ','13 = ','14 = ','15 = ',
                                    '16 = ','17 = ','18 = ','19 = ',
                                    '20 = ','21 = ','22 = ','23 = ');

var
   TrailNum,DateNum,PathNum,YearNum,
   Ch      : char;
   Choice  : integer;
   Info    : Trail;
   Fil     : file of Trail;
   DataU   : array[1..12,1..31,0..23] of integer;
   DataD   : array[1..12,1..31,0..23] of integer;
   Year,
   Month,
   Day,
   Hour,
   Min,
   Code,
   MonthTotal_U,
   DayTotal_U,
```

```
   HourTotal_U,
   FinalTotal_U,
   MonthTotal_D,
   DayTotal_D,
   HourTotal_D,
   FinalTotal_D,
   MonthCount_U,
   MonthCount_D,
   MonthNum,
   I          : integer;
   Yr         : string[2];
   AvMonth_U  : real;
   AvMonth_D  : real;
   AvDay      : real;
   ErrorReturn : string[24];
   Save_Register,Reg : registers;
   HourComp,MinComp,SecComp,SecComp100,MonthComp,DayComp,YearComp,
   WeekDayComp : word;
   IOCode : integer;
(* ------------------------------------------------------------------------------------------------*)
Procedure KillCursor;                                    {to hide cursor}
Begin
  with Save_Register do
    begin
      BH := 01;                                          {01}
      AH := 03;                                          {03}
      intr($10,Save_Register)                            {save current cursor}
    end;
  with REG do
    Begin
      CH := $12;                                         {00}
      CL := $12;                                         {12}
      AH := 01;                                          {01}
      intr($10,REG)
    End
end;
(* ------------------------------------------------------------------- *)
Procedure RestoreCursor;                                 {to restore cursor}
begin
 Save_Register.AH := 01;
 intr($10,Save_Register)
end;
(* ------------------------------------------------------------------- *)
procedure Screen;                                        {choice menu}
var I : integer;
begin
  Clrscr;
  gotoXY(17,05);
  write('▄▄▄▄▄▄▄▄ MENU ▄▄▄▄▄▄▄▄▄');
  for I := 06 to 17 do
   begin
     gotoXY(17,I);write('▌ ');
```

```
    gotoXY(58,I);write(' ');
   end;
  gotoXY(17,18);
  write('[redacted]');
  for I := 1 to 4 do
  begin
    gotoXY(Xas,Yas + I);
    write(Item[I]);
  end;
end;
(* ---------------------------------------------------------------------- *)
procedure WriteChoice;                                  {to high_light choice}
begin
  gotoXY(Xas,Yas + Choice);
  textcolor(black);
  textbackground(white);
  write(Item[Choice]);
  gotoXY(HXas,HYas);
  write(Help[Choice]);
  textcolor(white);
  textbackground(black);
end;
(* ---------------------------------------------------------------------- *)
procedure Up;                                           {to select choice}
begin
  gotoXY(Xas,Yas + Choice);
  write(Item[Choice]);
  if Choice <> 1 then dec(Choice) else Choice := 4;
  WriteChoice;
end;
(* ---------------------------------------------------------------------- *)
procedure Down;                                         {to select choice}
begin
  gotoXY(Xas,Yas + Choice);
  write(Item[Choice]);
  if Choice <> 4 then inc(Choice) else Choice := 1;
  WriteChoice;
end;
(* ---------------------------------------------------------------------- *)
function Exist: boolean;                                {to see if file exist}
begin
  {$I-}
     reset(fil);
  {$I+}
  Exist := (IOresult = 0);
end;
(* ---------------------------------------------------------------------- *)
procedure OpenPort;                                     {to open RS232_port for communication}
begin
  CommInit(0,baud1200+noparity+onestop+len8);
end;
(* ---------------------------------------------------------------------- *)
```

```
procedure Download;                                    {to down load data from selected card}

var
  Net    : array[1..32000] of char;
  A,B    : integer;
  Ch     : char;
   I     : integer;
 Data    : integer;
 TimeOut:  boolean;
 label load;
 label erase;
 label again;
begin                                                  {to display downloading menu}
  for I := 1 to 32000 do Net[I] := ' ';
  clrscr;
  gotoXY(1,2);
  write('┌──────────────────────┤ DOWN LOADING PROCESS ├──────────',
        '──────────┐');
  for I := 3 to 23 do
   begin
     gotoXY(1,I); write('│');
     gotoXY(79,I);write('│');
   end;
  gotoXY(1,24);
  write('└──────────────────────────────────────────────────────',
        '──────────┘');
  gotoXY(60,09);
  write('┌──────────────┐');
  for I := 10 to 20 do
   begin
     gotoXY(60,I); write('│');
     gotoXY(76,I); write('│');
   end;
  gotoXY(60,21);
  write('└──────────────┘');
again:
  gotoXY(30,7);
  textcolor(black);
  textbackground(white);
  write(' ENTER PATH No:AA - ZZ =   ');
  textcolor(white);
  textbackground(black);
  gotoXY(3,23);
  write('Press ESC to Exit to Menu');
  OpenPort;                                            {open port for communication}
  if CharReady then                                    {clear port}
  Ch := chr(getbyte);
  RestoreCursor;
  gotoXY(55,7);
  repeat
    PathNum := readkey;                                {to enter correct card number}
    if PathNum = #0 then Ch := readkey;
```

```
  if PathNum = #27 then
  begin
    KillCursor;
    exit;
  end;
  PathNum := upcase(PathNum);
until PathNum in ['A'..'Z'];
textcolor(black);
textbackground(white);
write(PathNum);
sendbyte(ord(PathNum));                          {send first character to card}
textcolor(white);
textbackground(black);
repeat
  TrailNum := readkey;                           {to enter correct card number}
  if TrailNum = #0 then Ch := readkey;
  if TrailNum = #27 then
  begin
    KillCursor;
    exit;
  end;
  TrailNum := upcase(TrailNum);
until TrailNum in ['A'..'Z'];
textcolor(black);
textbackground(white);
write(TrailNum);
sendbyte(ord(TrailNum));                         {send second character to card}
textcolor(white);
textbackground(black);
KillCursor;
gotoXY(5,16);
textcolor(black + blink);
textbackground(white);
write('!!! DOWN LOADING IN PROGRESS !!!');
textcolor(white);
textbackground(black);
window(64,10,72,20);                             {display data in a window}
if CharReady then                                {clear port again}
Ch := chr(getbyte);
TimeOut := false;                                {reset timer to 0}
I := 0;
sendbyte(ord(TrailNum));                         {send character to card to start downloading}
repeat
  inc(I);                                        {start timer}
  if I= 30000 then TimeOut := true;
until CharReady or TimeOut;
if  TimeOut then                                 {if time expired}
begin
  window(1,1,80,25);
  gotoXY(5,16);
  write('!!! CARD NOT INSERTED OR WRONG CARD !!!');
  repeat
```

```
      Ch := readkey;
    until Ch = #27;
    gotoXY(5,16);
    write('                                        ');
    goto again
  end else
  begin
  I := 1;
    while (I < 32000) do
    begin
      if CharReady then                          {start reading characters from card}
      begin                                      {store data temporary in array}
        Ch := chr(getbyte);
        if Ch = 'F' then                         {do downloading until end_marker is found}
        begin
          window(1,1,80,25);
          gotoXY(5,16);
          write('!!! DOWN LOADING FINISHED !!!    ');
          goto erase
        end else
        net[I] := Ch;
        write(ch);
        inc(I);
      end;
    end;
  end;
erase:
  gotoXY(3,23);
  write('                          ');
  gotoXY(5,17);
  write('!!! PRESS E FOR ERASING MEMORY !!! ');
  repeat
    Ch := readkey;
    Ch :=upcase(Ch);
  until Ch = 'E';
  sendbyte(ord(Ch));
  gotoXY(5,17);
  write('                                        ');
  gotoXY(5,16);
  textcolor(black + blink);
  textbackground(white);
  write('!!! ERASING IN PROGRESS !!!   ');
  textcolor(white);
  textbackground(black);
  repeat
    if CharReady then Ch := chr(getbyte)
  until Ch = 'F';
  gotoXY(5,16);
  write('!!! ERASING FINISHED !!!      ');
  gotoXY(3,23);
  write('Press ESC to Exit to Menu');
  repeat
```

```
    Ch := readkey;
  until Ch = #27;
  assign(Fil,concat('c:\Data\trail',PathNum,TrailNum,'.dta'));
  if exist then                                          {open file to store data}
  begin
    reset(fil);
    seek(fil,filesize(fil));
  end else rewrite(fil);
  A := 1; B := 9;                                        {copy data from array to file}
  repeat
    Info.time := '';
    for I := A to B do Info.Time := Info.Time + Net[I];
    write(fil,info);
    inc(A,9);
    inc(B,9);
  until Info.time = '         ';
  close(fil);
  gotoXY(5,16);
  write('                              ');
  window(64,10,72,20);                                   {clear small window}
  clrscr;
  window(1,1,80,25);
  for I := 1 to 32000 do Net[I] := ' ';
  goto again
end;
(* ---------------------------------------------------------------------- *)
procedure Message3;                                      {to display message:'file empty'}
begin
  gotoXY(29,4);
  write('            ');
  gotoXY(31,14);
  textcolor(black + blink);
  textbackground(white);
  write(' !! FILE EMPTY !!');
  gotoXY(28,16);
  write('Press ESC to Exit to Menu');
  textcolor(white);
  textbackground(black);
  repeat
    Ch := readkey;
  until Ch = #27;
  gotoXY(31,14);
  textcolor(white);
  textbackground(black);
  write('                ');
  gotoXY(28,16);
  write('                          ');
end;
(* ---------------------------------------------------------------------- *)
procedure Message2;                                      {to display message:'file not found'}
begin
  gotoXY(29,4);
```

```
  write('            ');
  gotoXY(31,14);
  textcolor(black + blink);
  textbackground(white);
  write('!! FILE NOT FOUND !!');
  gotoXY(28,16);
  write('Press ESC to Exit to Menu');
  textcolor(white);
  textbackground(black);
  repeat
    Ch := readkey;
  until Ch = #27;
  gotoXY(31,14);
  textcolor(white);
  textbackground(black);
  write('                    ');
  gotoXY(28,16);
  write('                         ');
end;
(* ------------------------------------------------------------------ *)
procedure GetFileData;                                  {copy data from file to array}
var  I,J,K : integer;
begin
  for I := 1 to 12 do
    for J := 1 to 31 do
      for K := 0 to 23 do DataU[I,J,K] := 0;
  for I := 1 to 12 do
    for J := 1 to 31 do
      for K := 0 to 23 do DataD[I,J,K] := 0;
  assign(Fil,concat('c:\data\trail',PathNum,TrailNum,'.dta'));
  if exist then                                         {open file where data were stored}
  begin
    reset(Fil);
    if filesize(fil) <> 0 then
    begin
      repeat
        read(Fil,Info);
        if Yr = copy(Info.Time,1,2) then
        begin
          val(copy(Info.time,1,2),Year,Code);
          val(copy(Info.time,3,2),Month,Code);
          val(copy(Info.time,5,2),Day,Code);
          val(copy(Info.time,7,2),Hour,Code);
          if (Copy(Info.time,9,1) = 'U') then
            DataU[Month,Day,Hour] := DataU[Month,Day, Hour] + 1;

          if (Copy(Info.time,9,1) = 'D') then
            DataD[Month,Day,Hour] := DataD[Month,Day, Hour] + 1;
        end;
      until eof(Fil);
    end else Message3
  end else Message2;
```

```
end;
(* ------------------------------------------------------------------ *)
procedure DisplayScreen;                                (screen for monthly display)
var I : integer;
begin
  clrscr;
  for I := 2 to 78 do
  begin
    gotoXY(I,1);
    write(Lyn[5]);
  end;

  gotoXY(79,1);
  write(Lyn[2]);

  for I := 2 to 21 do
  begin
    gotoXY(79,I);
    write(Lyn[3]);
  end;

  gotoXY(79,22);
  write(Lyn[4]);

  for I := 78 downto 2 do
  begin
    gotoXY(I,22);
    write(Lyn[5]);
  end;

  gotoXY(1,22);
  write(Lyn[6]);

  for I := 21 downto 2 do
  begin
    gotoXY(1,I);
    write(Lyn[3]);
  end;
  gotoXY(1,1);
  write(Lyn[1]);
  gotoXY(2,2);
  write('                                MONTHLY REPORT  ');
  for I := 2 to 78 do
  begin
    gotoXY(I,3); write(Lyn[5]);
    gotoXY(I,5); write(Lyn[5]);
    gotoXY(I,7);write(Lyn[5]);
    gotoXY(1,20);write(Lyn[5]);
  end;
  for I := 1 to 12 do
  begin
    gotoXY(3,7 + I);
```

```
    write(Months[I]);
  end;
  gotoXY(3,21);
  write('Final Total = ');
  gotoXY(53,21);
  textcolor(black);
  textbackground(white);
  write('Press ESC to Exit to Menu');
  textcolor(white);
  textbackground(black);
end;
(* ---------------------------------------------------------------------- *)
procedure DisplayData;                                  {display data on screen}

var I,J,K : integer;
    DirInfo : SearchRec;
label start1;
begin
  gotoXY(45,4);                                         {write current date & time}
  GetDate(MonthComp,DayComp,YearComp,WeekDayComp);
  write('Today is : ',Days[WeekDayComp]:9,MonthComp:5,'/',DayComp,'/',YearComp);
  gotoXY(45,6);
  GetTime(HourComp,MinComp,SecComp,SecComp100);
  write('Time is : ',HourComp,':',MinComp,':',SecComp);
  gotoXY(22,6);
  write('UP      DOWN');
start1:
  RestoreCursor;
  gotoXY(3,4);
  write('ENTER PATHNUM AA - ZZ :');                     {enter trail to be displayed}
  repeat
    PathNum := readkey;
    if PathNum = #0 then Ch := readkey;
    if PathNum = #27 then
    begin
      KillCursor;
      exit;
    end;
    PathNum := upcase(PathNum);
  until PathNum in ['A'..'Z'];
  write(PathNum);
  repeat
    TrailNum := readkey;
    if TrailNum = #0 then Ch := readkey;
    if TrailNum = #27 then
    begin
      KillCursor;
      exit;
    end;
    TrailNum := upcase(TrailNum);
  until TrailNum in ['A'..'Z'];
  write(TrailNum);
```

```
for I := 1 to 12 do
begin
  gotoXY(22,7 + I);
  writeln('              ');
  gotoXY(22,21);
  write('          ');
end;
gotoXY(3,6);
write('ENTER YEAR :9');                          {enter year to be displayed}
repeat
  YearNum := readkey;
  if YearNum = #0 then Ch := readkey;
  if YearNum = #27 then
  begin
    KillCursor;
    exit;
  end;
until YearNum in ['0'..'9'];
write(YearNum);
KillCursor;
Yr := concat('9',YearNum);
gotoXY(29,4);
textcolor(black + blink);
textbackground(white);
write('PLEASE WAIT!');
textcolor(white);
textbackground(black);
GetFileData;                                     {get info from the array}
gotoXY(29,4);
textcolor(white);
textbackground(black);
write('            ');
if exist and (filesize(fil) <> 0) then
begin                                            {up}
  MonthTotal_U := 0;
  FinalTotal_U := 0;
  MonthCount_U := 0;
  for I := 1 to 12 do                            {calculate total for each month}
  begin
    for J := 1 to 31 do
      for K := 0 to 23 do MonthTotal_U := MonthTotal_U + DataU[I,J,K];
    gotoXY(22,7 + I);
    writeln('',Monthtotal_U);
    if MonthTotal_U <> 0 then                    {calculate final total}
    begin
      FinalTotal_U := FinalTotal_U + MonthTotal_U;        {do finaltotal}
    end;
    MonthTotal_U := 0;
  end;
  gotoXY(22,21);
  write(FinalTotal_U);
```

```
    MonthTotal_D := 0;                                    {down}
    FinalTotal_D := 0;
    MonthCount_D := 0;
    for I := 1 to 12 do                                   {calculate total for each month}
    begin
      for J := 1 to 31 do
        for K := 0 to 23 do MonthTotal_D := MonthTotal_D + DataD[I,J,K];
      gotoXY(30,7 + I);
      writeln('',Monthtotal_D);
      if MonthTotal_D <> 0 then                           {calculate final total}
      begin
        FinalTotal_D := FinalTotal_D + MonthTotal_D;      {do finaltotal}
      end;
      MonthTotal_D := 0;
    end;
    gotoXY(30,21);
    write(FinalTotal_D);
    goto start1
  end;
end;
(* ------------------------------------------------------------------ *)
procedure Display;
begin
  DisplayScreen;
  DisplayData;
end;
(* ------------------------------------------------------------------ *)
procedure digit;
Label out,out1;
Var  Ch : char;
     st2 : string[2];
     pass : boolean;
     code  : integer;

begin
  st2 := '';
  repeat
    out:
    Ch := readkey;
    pass := (Ch in[#48..#57]);
    If not pass
      then
        if Ch = #13
          then
            goto out1
          else
            goto out;
    st2 := st2 + Ch;
    GotoXY(54,11);
    Write(st2:1);
    out1:
  until Ch = #13;
```

```
  val(st2,I,code);
end;
(* ------------------------------------------------------------------ *)
procedure Monthly;                                    {print monthly data}
var
  I,J,K  : integer;
begin                                                 {print current date & time}
  GetDate(MonthComp,DayComp,YearComp,WeekDayComp);
  GetTime(HourComp,MinComp,SecComp,SecComp100);
  writeln(lst,'┌──────────────────────────────────────┐');
  writeln(lst,'│     MONTHLY REPORT for YEAR 199',YearNum,'     │');
  writeln(lst,'├──────────────────────────────────────┤');
  writeln(lst,'│Today is:',Days[WeekDayComp]:9,MonthComp:5,'/',DayComp:2,'/',YearComp:2,'         │');
  writeln(lst,'│Time is:',HourComp:2,':',MinComp:2,':',SecComp:2,'                    │');
  writeln(lst,'│TRAIL:',PathNum,TrailNum,'                              │');
  writeln(lst,'│MONTHS     UP DOWN                    │');
  writeln(lst,'├──────────────────────────────────────┤');
  MonthTotal_U := 0;                                  {calculate month totals & print}
  FinalTotal_U := 0;
  MonthTotal_D := 0;                                  {calculate month totals & print}
  FinalTotal_D := 0;
  for I := 1 to 12 do
  begin
    for J := 1 to 31 do
      for K := 0 to 23 do MonthTotal_U := MonthTotal_U + DataU[I,J,K];
        begin
          for J := 1 to 31 do
            for K := 0 to 23 do MonthTotal_D := MonthTotal_D + DataD[I,J,K];
            writeln(lst,'│',MonthP[I],'     ',Monthtotal_U:4,' ',MonthTotal_D:4,'                  │');
            if MonthTotal_U <> 0 then
            begin
              FinalTotal_U := FinalTotal_U + MonthTotal_U;
            end;
            if MonthTotal_D <> 0 then
            begin
              FinalTotal_D := FinalTotal_D + MonthTotal_D;
            end;
            MonthTotal_U := 0;
            MonthTotal_D := 0;
          end;
  end;
  writeln(lst,'├──────────────────────────────────────┤');
  writeln(lst,'│F/TOTAL =',FinalTotal_U:4,' ',FinalTotal_D:4,'                  │');
  writeln(lst,'└──────────────────────────────────────┘');
end;
(* ------------------------------------------------------------------ *)
procedure Dayly;                                      {print dayly data}
Var  Ch : char;
     st2 : string[2];
     pass : boolean;
     code  : integer;
     I,J,K : integer;
```

```
    label loop1,out,out1;
begin
  loop1:
  RestoreCursor;
  gotoXY(35,11);
  write('ENTER MONTH (1-12):');
  st2 := '';
out:
  repeat
    Ch := readkey;
    pass := (Ch in['1'..'9']);
    If not pass then
    If Ch = #13 then goto out1 else
    If Ch = #27 then
       begin
         KillCursor;
         gotoXY(35,11);
         write('                          ');
         exit;
       end else goto out;
    st2 := st2 + Ch;
    GotoXY(54,11);
    Write(st2:1);
  until Ch = #13;
out1:
  val(st2,I,code);
  if (I > 12) or (I < 1) then
    begin
      Sound(220);
      Delay(200);
      NoSound;
      gotoXY(35,11);
      write('                           ');
      goto loop1
    end else
  gotoXY(35,11);
  write('                    ');
  KillCursor;
  GetDate(MonthComp,DayComp,YearComp,WeekDayComp);
  GetTime(HourComp,MinComp,SecComp,SecComp100);
  writeln(lst,'┌──────────────────────────────────┐');
  writeln(lst,'│    DAILY REPORT for ',MonthP[I],'199',YearNum,'         │');
  writeln(lst,'├──────────────────────────────────┤');
  writeln(lst,'│Today is:',Days[WeekDayComp]:9,MonthComp:5,'/',DayComp:2,'/',YearComp:2,'         │');
  writeln(lst,'│Time is:',HourComp:2,':',MinComp:2,':',SecComp:2,'                    │');
  writeln(lst,'│TRAIL:',PathNum,TrailNum, '                          │');
  writeln(lst,'│DAYS       UP DOWN                │');
  writeln(lst,'├──────────────────────────────────┤');
  DayTotal_U := 0;                                    {calculate day totals & print}
  FinalTotal_U := 0;
  DayTotal_D := 0;                                    {calculate day totals & print}
  FinalTotal_D := 0;
```

```
  for J := 1 to 31 do
  begin
    for K := 0 to 23 do DayTotal_U := DayTotal_U + DataU[I,J,K];
      begin
        for K := 0 to 23 do DayTotal_D := DayTotal_D + DataD[I,J,K];
        writeln(lst,'|',DayP[J],'        ',DayTotal_U:3,'  ',DayTotal_D:3,'                    |');
      if DayTotal_U <> 0 then
        begin
          FinalTotal_U := FinalTotal_U + DayTotal_U;
        end;
        if DayTotal_D <> 0 then
        begin
          FinalTotal_D := FinalTotal_D + DayTotal_D;
        end;
        DayTotal_U := 0;
        DayTotal_D := 0;
      end;
  end;
  writeln(lst,'├──────────────────────────────────────┤');
  writeln(lst,'|F/TOTAL =',FinalTotal_U:4,' ',FinalTotal_D:4,'                    |');
  writeln(lst,'└──────────────────────────────────────┘');
write(lst,#12);
end;
(* ---------------------------------------------------------------------- *)
procedure Hourly;                                    {print hourly data}
Var  Ch : char;
     st2 : string[2];
     pass : boolean;
     code  : integer;
     I,J,K : integer;
     label loop1,out,out1;
begin
write(lst,#27,#15);                                  {set printer for condense mode}
loop1:
  RestoreCursor;
  gotoXY(35,12);
  write('ENTER MONTH (1-12):');
  st2 := '';
out:
  repeat
    Ch := readkey;
    pass := (Ch in['1'..'9']);
    If not pass then
    If Ch = #13 then goto out1 else
    If Ch = #27 then
       begin
         KillCursor;
         gotoXY(35,12);
         write('                          ');
         exit;
       end else goto out;
    st2 := st2 + Ch;
```

```
    GotoXY(54,12);
    Write(st2:1);
  until Ch = #13;
out1:
  val(st2,I,code);
  if (I > 12) or (I < 1) then
    begin
      Sound(220);
      Delay(200);
      NoSound;
      gotoXY(35,12);
      write('                              ');
      goto loop1
    end else
  gotoXY(35,12);
  write('                      ');
  KillCursor;
  GetDate(MonthComp,DayComp,YearComp,WeekDayComp);
  GetTime(HourComp,MinComp,SecComp,SecComp100);
  writeln(lst,'┌──────────────────────────────────────────────────────────────',
              '──────────────────────────────────────────────┐');
  writeln(lst,'│                                        HOURLY REPORT for ',MonthP[I],'199',YearNum,
                    '                                        │');
  writeln(lst,'├──────────────────────────────────────────────────────────────',
              '──────────────────────────────────────────────┤');
  writeln(lst,'│Today is: ',Days[WeekDayComp]:9,MonthComp:5,'/',DayComp:2,'/',YearComp:2,'
',
  '                                                                        │');
  writeln(lst,'│Time is: ',HourComp:2,':',MinComp:2,':',SecComp:2,'                                  ',
  '                                                                        │');
  writeln(lst,'│TRAIL: ',PathNum,TrailNum, '                                        ',
  '                                                                        │');
  writeln(lst,'│                                                       UP      ',
              '                                                          │');
  writeln(lst,'├──────────────────────────────────────────────────────────────',
              '──────────────────────────────────────────────┤');
  writeln(lst,'│HOURS│ 00  01  02  03  04  05  06  07  08  09  10  11  12',
              '  13  14  15  16  17  18  19  20  21  22  23 │TOTALS│');
  writeln(lst,'│DAYS ├──────────────────────────────────────────────────────',
              '──────────────────────────────────────────────┤');
 FinalTotal_U := 0;
  begin
    for J := 1 to 31 do                                    {do hourly printing}
    begin
      write(lst,DayPh[J]);
      for K := 0 to 23 do
      if DataU[I,J,K] <> 0 then
      begin
        FinalTotal_U := FinalTotal_U + DataU[I,J,K];
        write(lst,DataU[I,J,K]:4);
      end else
      write(lst,'.':4);
```

```
      write(lst,' |');
      writeln(lst,FinalTotal_U:3,'   |');
      FinalTotal_U := 0;
    end;
  end;
  writeln(lst,'└────┴──────────────────────────────────────────────────',
              '──────────┴────┘');
  write(lst,#12);
  writeln(lst,'┌──────────────────────────────────────────────────',
              '──────────┐');
  writeln(lst,'│                                        HOURLY REPORT for ',MonthP[I],'199',YearNum,
                '                                        │');
  writeln(lst,'├──────────────────────────────────────────────────',
              '──────────┤');
  writeln(lst,'│Today is: ',Days[WeekDayComp]:9,MonthComp:5,'/',DayComp:2,'/',YearComp:2,'
',
  '                                                        │');
  writeln(lst,'│Time is: ',HourComp:2,':',MinComp:2,':',SecComp:2,'                                ',
  '                                                        │');
  writeln(lst,'│TRAIL: ',PathNum,TrailNum, '                                      ',
  '                                                        │');
  writeln(lst,'│                                          DOWN      ',
              '                                            │');
  writeln(lst,'├────┬──────────────────────────────────────────────────',
              '──────────┤');
  writeln(lst,'│HOURS│  00  01  02  03  04  05  06  07  08  09  10  11  12',
              '  13  14  15  16  17  18  19  20  21  22  23 │TOTALS│');
  writeln(lst,'│DAYS ├──────────────────────────────────────────────────',
              '──────────┤');
  FinalTotal_D := 0;
   begin
     for J := 1 to 31 do                                  {do hourly printing}
     begin
       write(lst,DayPh[J]);
       for K := 0 to 23 do
       if DataD[I,J,K] <> 0 then
       begin
         FinalTotal_D := FinalTotal_D + DataD[I,J,K];
         write(lst,DataD[I,J,K]:4);
       end else
       write(lst,'.':4);
       write(lst,' |');
       writeln(lst,FinalTotal_D:3,'   |');
       FinalTotal_D := 0;
     end;
   end;
  writeln(lst,'└────┴──────────────────────────────────────────────────',
              '──────────┘');
  write(lst,#12);
  write(lst,#18);                                         {cancel condense mode}
end; .
(* ------------------------------------------------------------------ *)
```

```
procedure Message1;                                    {to display if printer is faulty}
begin
  gotoXY(20,14);
  textcolor(black + blink);
  textbackground(white);
  write('PRINTER ERROR:',ErrorReturn);
  gotoXY(28,16);
  write('Press ESC to Exit to Menu');
  textcolor(white);
  textbackground(black);
  repeat
    Ch := readkey;
  until Ch = #27;
  gotoXY(20,14);
  textcolor(white);
  textbackground(black);
  write('                                    ');
  gotoXY(28,16);
  write('Press ESC to Exit to Menu');
end;
(* ------------------------------------------------------------------ *)
function PrintReady: boolean;                          {to check printer status}
var
   Status      : byte;
   ErrorCode   : integer;
begin
  PrintReady := false;
  Status := port[$03BD];
  if Status in [135,120] then                          {printer off}
      ErrorCode := 0;
  if Status in [87,80] then                            {printer off line}
      ErrorCode := 1;
  if Status in [95] then                               {printer busy}
      ErrorCode := 2;
  if Status in [119] then                              {paper out}
      ErrorCode := 3;
  if Status in [223,146,216] then
      PrintReady := true;
  ErrorReturn := Error[ErrorCode];
end;
(* ------------------------------------------------------------------ *)
procedure Printing;                                    {menu to print data}
var
  Ch    : char;
  I     : integer;
begin
clrscr;
  gotoXY(11,3);
   write('┌──────────────────┤ PRINT MENU ├──────────────────┐');
  for I := 4 to 15 do
   begin
     gotoXY(11,I); write('│');
```

```
   gotoXY(68,I); write('│');
 end;
gotoXY(11,16);
write('└──────────┤Press ESC to Exit to Menu├──────────┘');
gotoXY(12,8);
write('──────────────────────────────────────────────────');
RestoreCursor;
gotoXY(29,5);
write('ENTER PATHNUM AA - ZZ :');                    {enter trail to be printed}
repeat
  PathNum := readkey;
  if PathNum = #0 then Ch := readkey;
  if PathNum = #27 then
  begin
    KillCursor;
    exit;
  end;
  PathNum := upcase(PathNum);
until PathNum in ['A'..'Z'];
write(PathNum);
repeat
  TrailNum := readkey;
  if TrailNum = #0 then Ch := readkey;
  if TrailNum = #27 then
  begin
    KillCursor;
    exit;
  end;
  TrailNum := upcase(TrailNum);
until TrailNum in ['A'..'Z'];
write(TrailNum);
gotoXY(33,6);
write('ENTER YEAR :9');                              {enter year to be printed}
repeat
  YearNum := readkey;
  if YearNum = #0 then Ch := readkey;
  if YearNum = #27 then
  begin
    KillCursor;
    exit;
  end;
until YearNum in ['0'..'9'];
write(YearNum);
KillCursor;
Yr := concat('9',YearNum);
gotoXY(28,16);
textcolor(black + blink);
textbackground(white);
write('      PLEASE WAIT!       ');
textcolor(white);
textbackground(black);
GetFileData;                                         {copy data from file to array}
```

```
  if exist and (filesize(fil) <> 0) then
  begin
    gotoXY(28,16);
    write('Press ESC to Exit to Menu');
    for I := 1 to 3 do
      begin
        gotoXY(15,9 + I);
        write(Print[I]);
      end;
      repeat
        Ch := readkey;
        Ch := upcase(Ch);
          case Ch of
            'A': begin                                  {if A then do monthly printing}
                   if PrintReady then Monthly else Message1;
                 end;
            'B': begin                                  {if B then do daly printing}
                   if PrintReady then Dayly else Message1;
                 end;
            'C': begin                                  {if C then do hourly printing}
                   if PrintReady then Hourly else Message1;
                 end;
          end;
      until Ch = #27;
  end;
(*  write(lst,#18);                                   {set printer for normal mode}*)
end;
(* ------------------------------------------------------------------------ *)
begin                                                 {main program}
  KillCursor;                                         {hide cursor}
  textcolor(white);
  textbackground(black);
  Choice := 1;
  repeat
    Screen;                                           {display main menu}
    WriteChoice;
    repeat
      Ch := readkey;
      case Ch of
        #72 : Up;
        #80 : Down;
      end;
    until Ch = #13;
    case Choice of
      1 : DownLoad;                                   {to down load data from card into a file}
      2 : Printing;                                   {to  print data}
      3 : Display;                                    {to display data on screen}
    end;
  until Choice = 4;
  clrscr;
  RestoreCursor;                                      {restore cursor}
end.
```