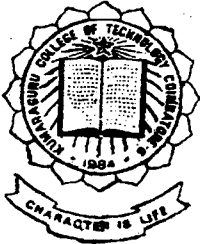


Automation Of Yarn Winding Machine

Project Report



P - 529

Submitted by

K. Karthik
S. Praveen Moses
K. Saravanan
K. Shanmuga Sundaram

Guided by

Mr. G. Sivasalopathy, B.E.,
Senior Lecturer
EEE Department.

Sponsored by

Miltronics, Coimbatore

In partial fulfilment of the requirements
for the award of the Degree of
BACHELOR OF ENGINEERING in
Electrical and Electronics Engineering
of the Bharathiar University.

2000-2001

Department of Electrical and Electronics Engineering

Kumaraguru College of Technology

Coimbatore – 641 006.

CERTIFICATE

**Department of Electrical and Electronics
Engineering
Kumaraguru College of Technology
Coimbatore – 641 006.**

CERTIFICATE


This is to certify that the Report entitled
AUTOMATION OF YARN WINDING MACHINE
Has been submitted by

**K. Karthik
S. Praveen Moses
K. Saravanan
K. Shanmuga Sundaram**

In partial fulfilment of the requirements for the award of
Bachelor of Engineering
in the Electrical and Electronics Engineering
Branch of the Bharathiar University, Coimbatore – 641 046
during the academic year 2000-2001.



.....
Guide



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

Certified that the candidate with University Register Number
was examined in project work viva-voce Examination held on

.....
Internal Examiner

.....
External Examiner



12.3.2001

TO WHOMSOEVER IT MAY CONCERN

This is to Certify that the Project titled AUTOMATION OF YARN WINDING MACHINE was sponsored by us & was developed by the following students of Final Year B.E(EEE) of Kumaraguru college of Technology,CBE.with the assistance of our research & development department completed by March 2001.

1. Mr.K. Karthik (97EEE22)
2. Mr.S.Praveen mooses (97EEE41)
3. Mr.K.Saravanan (97EEE48)
4. Mr.K.Shanmuga sundaram (97EEE66)

The above mentioned prototype was successfully completed with all the above students actively participating in the development.During the period of development, their attendance & Conduct was very good .

We wish them all success in future.

for MILTRONICS

Authorized Signatory

***Dedicated
to Our
Beloved Parents and
Friends***

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

With deep sense of gratitude we express our heartfelt thanks to our guide **Mr.G.Sivasalopathy**, BE., Senior Lecturer in EEE for his guidance, valuable suggestions and constant interest evinced by him throughout the course of the project work.

We are highly grateful to our beloved professor and Head of the Department **Dr.K.A.Palaniswamy**, M.Sc (Engg.), Ph.D., C.Eng.(I), M.I.S.T.E., F.I.E., for his remarkable support and encouragement.

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

We would like to thank **Mr. C.Jayaveeraiya**, Chief Executive of miltronics for providing us an opportunity to do this project work at miltronics, Coimbatore.

We have no words to express our profound gratitude to our esteemed guide **Ms.Shanthi**, B.E, for her valuable co-operation and suggestions without which our project would not have been feasible.

We are indebted to the support, encouragement and help rendered by all the faculty members and the non-teaching staff of the Electrical and electronics Engineering Department.

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

SYNOPSIS

SYNOPSIS

This project is based on interfacing of length measuring system used in winding section of textile industry to the personal computer. This system is accurate and versatile. Normally in textile industry, the total production is calculated from the length of yarn. Thus maintenance of yarn length to exact specified length is must for both in quality and quantity aspect.

The existing unit controls single machine and every time the unit is to be reset and the new preset value should be manually entered for all drums connected to it. In this project we connect many machines to PC.

A user-friendly frontend designed using VB displays the status of various units such as:

- ✓ Length of yarn measured/ drum.
- ✓ Error indication
- ✓ Preset length/ drum.

This project aims at connecting more number of machines to a single PC. Monitoring and controlling of all machines as well as drums are possible by single touch of a key.

CONTENTS

CONTENTS

Chapter	Page no.
CERTIFICATE	
ACKNOWLEDGEMENT	
SYNOPSIS	
1. INTRODUCTION	1
1.1. INTRODUCTION TO MICRO CONTROLLER	
1.2. 8031 MICROCONTROLLER	
1.3. FEATURES OF 8031	
2. HARDWARE DESCRIPTION	6
2.1. PORT STRUCTURE AND OPRETIIONS	
2.1.1 PORT 0	
2.1.2 PORT 1	
2.1.3 PORT 2	
2.1.4 PORT 3	
2.2 INTERRUPTS	
2.3 SERIAL INTERFACE	
2.4 TIMER/ COUNTERS	
3. PROJECT DESCRIPTION	17
3.1. MICROCONTROLLER UNIT	
3.2. INTERFACING UNIT	
3.2.1. TRANSCEIVER	
3.2.2. CONVERTER	
4. SOFTWARE DESCRIPTION	24
4.1.PROGRAM CODING	
5. INTRODUCTION TO VISUAL BASIC	45
5.1 DEVELOPING AN APPLICATION	

6.	FEATURES OF VISUAL BASIC –6	48
	6.1. DATA BASE CONCEPTS IN VISUAL BASIC	
	6.2. RECORD SETS	
	6.3. MS-COMM CONTROL	
7.	FRONTEND DESIGN USING VISUAL BASIC	52
8.	DESCRIPTION OF THE PROJECT	55
9.	SOFTWARE	58
	9.1. FORM WINDOWS	
	9.2. PROGRAM CODING	
10.	CONCLUSION	75
	REFERENCES	76
	APPENDIX	77

CHAPTER 1

INTRODUCTION

CHAPTER.1

Introduction

India is one of the fast developing countries in the world at present. The economy of India depends upon the various Industries. The textile industry plays an important role for the economy of our country. Textile goods manufactured in India is popular all over the world.

Nowadays automation is used almost in all the industries. Textile industry is also not lagging behind in the use of Electronics and computers, as it has to face the challenge of the emerging Globalization of the market. Thus various new Electronics technique should be employed to make the textile industry more efficient and profitable. So Electronics and computers provide a wide scope for development of textile industry.

The era of manual control is fast giving way to automatic control system. Computers play a major role in the field of automatic control and monitoring systems. The main advantage of using computers is that they are more accurate and compact. Moreover precise monitoring and are faster monitoring systems. With the advent of microprocessors and microcontroller the same chip is performing various complex functions. Thus the size of the control system is reduced greatly.

The main aim of this project is to monitor the cone winding process. The wastage of yarn is reduced considerably; the accurate length of Yarn is measured and the

production details are maintained as records. The advantage of this project is that it drastically reduces the number of laborers required.

This project is the interfacing of a personal computer with a microcontroller-based system.

1.1 INTRODUCTION TO MICRO CONTROLLERS:

A microcontroller is nothing but a true computer on a single chip. The design incorporates all of the features found in a microprocessor CPU : ALU, PC, SP and Registers. It also has added other features needed to make a complete computer: ROM, RAM, Parallel I/O, Serial I/O counters and a clock circuit.

Like the microprocessor a microcontroller is a general purpose device, but one that is mean to read data, perform limited calculations on that data, and control its environment based on those calculations. The prime use of a microcontroller is to control the operation of a machine using a fixed program that is stored in ROM and that doses not change over the life of a system.

The microcontroller design uses a much more limited set of single and double byte instructions that are used to move code and data from internal memory in to ALU. Many instructions are coupled with pins on the integrated circuit package, the pins are programmable

that is, capable of having several different functions depending on the wishes of the programmer.

The evaluation of the microcontroller based system follows these steps

- Define a specification.
- Design a microcontroller system to this specification.
- Write programs that will assist in checking the design.
- Write several common subroutines and test them.

1.2 8031 MICROCONTROLLER

The Intel MCS-51 family is a highly versatile general-purpose 8-bit system. Its enhanced architecture offers applications requiring a high degree of on-chip functionality. It is most suitable for control-oriented applications. The MCS-51 family includes 8031, 8051, and 8751.

The 8051 family is chosen for the following reasons:

- Low part cost.
- Multiple vendors.
- Available in NMOS and CMOS technologies.
- Software tools available and inexpensive.
- High-level language compilers available.

The 8031 has control-oriented RAM and I/O. It is actually a ROM-less version of the 8051 and it fetches all instructions from external memory. The CPU of 8031 is very fast with larger number of powerful single cycle opcodes. In the other general purpose microprocessor a lot of hardware is necessary, but in 8031 it needs very few simple interfaces to be associated with, due to its very scale integration on a single chip. It includes a clock, interrupts, timers,

UART, I/O ports, and RAM, etc. This chip is ideal in control, instrumentation, robotics, and data processing applications.

The 8031 operate on 5 volts supply with power down mode for saving internal RAM contents. It possesses 128 bytes of register memory on a chip. Four banks of 8 registers have special functions and 2 registers have special indexing capability in each bank. It is equipped with 12 MHz crystal having clock time 1 microsecond instruction cycle. It has 2-level priority, level or edge trigger able external interrupts, two timer interrupts and one UART interrupts.

The 8031 possess one bit addressable 8-bit I/O port and one UART port, which is high speed programmable. Two-timer inputs maybe gated by interrupt inputs. Two multi mode 16-bit timer or counters enhance the system performance.

The ALU has the capability of performing binary or decimal, arithmetic, Boolean bit processing for control logic programs, 8-bit multiplication or division in 4 microseconds, parity computations, and overflow detections, etc.

1.3 FEATURES OF 8031

The 8031 is a member of the MCS-51 family and it is an 8-bit microcontroller. The main features are as follows:

- High performance HMOS process.
- Two 16-bit Timers/Counters.
- Two levels interrupt priority structure.
- 32 bi-directional and individually addressable I/O lines.
- On chip oscillator and clock circuitry.
- 64k address space for external data memory.
- 64k address space for external program memory.
- 111 instructions.
- Boolean processor.
- Programmable Full Duplex serial channel.

CHAPTER 2

HARDWARE DESCRIPTION

CHAPTER. 2

HARDWARE DESCRIPTION

2.1 PORT STRUCTURES AND OPERATION

All four ports in the 8031 are bi-directional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of port 0 and 2, and the input buffers of port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the port 2 pins continue to emit the SFR content.

All the port 3 pins, and (in the 8031) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed,

Port Pin	Alternate Function
* P1.0	T2 (Timer/Counter 2 external input)
* P1.1	T2EX (Timer/Counter 2 Capture/ Reload trigger)
P3.0	RXD (special input port)
P3.1	TXD (Serial output port)
P3.2	<u> </u> INT0 (external interrupt)
P3.3	<u> </u> INT1 (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	<u> </u> WR (external data memory write strobe)
P3.7	<u> </u> RD (external data memory read strobe)

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

2.1.1. PORT 0

Port 0 pins may serve as inputs, outputs, or, when used together, as a bi-directional low-order address and data bus for external memory. For example, when a pin is to be used as an input, a 1 must be written to the corresponding port 0 latch by the program, thus turning both of the output transistors off, which in turn causes the pin to “float” in a high-impedance state, and the pin is essentially connected to the input buffer.

When used as an output, the pin latches that are programmed to a 0 will turn on the lower FET, grounding the pin. All the pins that are programmed to a 1 still float thus; external pull up resistors will be needed to supply logic high when using port 0 as an output.

When port 0 is used as an address bus to external memory, internal control signals switch the address lines to the gates of the Field effect transistor. Logic 1 on a address bit will turn the upper FET on and the lower FET to provide a logic high at the pin. When the address bit is zero, the lower Fet is on and the upper FET off to provide a logic low at the pin. After the address has been formed and latched into external circuits by the address latch enable pulse, the bus is turned around to become a data bus. Port 0 now reads the data from the external memory and must be configured as an input, so a logic 1 is automatically written by internal control logic to all port 0 latches.

2.1.2. Port 1

Port 1 one pins have no dual functions. Therefore, the output latch is connected directly to gate of the lower fit, which has s FET circuit labeled internal FET pull up as an active pull up load. Used as an input, a1 is returned to the latched, turning the lower fit off, the pin and the input to the pin buffer are pulled by the FET load. An external circuit can overcome the high impedance pull up and drive the pin low to input a 0 or leave the input high for 1.

If used as an output the latches containing can drive the input of an external circuit high through the pull up. If a 0 is written to the latch, the lower FET is on, the pull up is off and the pin can drive up the input of the external circuit low. To aid in speeding up switching times when the pin is used as an output, the internal FET pull up has another FET in parallel with it. The second FET is turned in two oscillators time periods during a low to high transition on the pin. This arrangement provides a low impedance path to the positive voltage supply to help reduce rise times in charging any parasitic capacitances in the external circuitry.

2.1.3 PORT 2

Port2 is used as an input or output port similar in operation to port1. The alternate use of port2 is to supply a high order address byte in conjunction with the port 0 low order byte to external memory.

Port 2 pins are momentarily changed by the address control signals when supplying the high byte of a 16-bit address. Port2 latches remain stable when external memory is addressed, as they do not have to be turned around for data input as in the case port 0.

2.1.4. PORT3

Port 3 is an input or output similar to port1. The input and output functions can be programmed under the control of P3 latches or under the control of various other special function registers. Unlike ports 0&2, and which can have external addressing functions and change all eight port bits when in all alternate use, each pin of port3 may be individually programmed to be used either as I/O or as one of the alternate functions.

2.2. INTERRUPTS:

The 8031 provide 5 interrupt sources where as 8052 provides 6 interrupts.

The external interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level activated or transition activated depending on bits IT0 and IT1 in register TCON. The flags that actually generate these interrupts are bits IE 0 and IE 1 in TCON. When an external is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition

activated. If the interrupt was level activated, then the external requesting source is what controls the request flag, rather than the on chip hardware.

The timer 0 and timer 1 interrupts are generated by TF0 and TF1, which are set by a roll over in their respective timer or counter register. When a timer interrupt is generated, the flag that generated it is cleared by the on chip hardware when the service routine is vectored to.

The serial port interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact the service routine will normally have to determine whether it was RI or TI that generated the interrupt and the bit will have to be cleared in software.

2.3. SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still has not been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at special function register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port can operate in 4 modes:

MODE 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted / received: 8 data bits. The baud rate is fixed at 1/12 the oscillator frequency

MODE 1

10 bits are transmitted (through TXD) or received (through RXD): a start bit 0, 8 data bits, and a stop bit 1. On receive; the stop bit goes into RB8 in special function register SCON. The baud rate is variable.

MODE 2

11 bits are transmitted (through TXD) or received (through RXD): a start bit 0, 8 data bits, a programmable 9th data bit and a stop bit 1. On transmit; a 9th data bit can be assigned the value of 0 or 1. Or, for example the parity bit could be moved into TB8. On receive, the 9th data bit goes into RB8 in special function register SCON, while the stop bit is ignored. The baud rate is programmed to either 1/32 or 1/64 the oscillator frequency.

MODE 3

11 bits are transmitted (through TXD) or received (through RXD) : a start bit 0 , 8 data bits , a programmable 9th data bit and a stop bit 1. In fact, mode 3 is same as mode 2 in all respects except the baud rate. The baud rate in mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in mode 0 by the condition RI=0 and REN=1. Reception is initiated by in the other modes by the incoming start bit if REN=1.

2.4. TIMER/COUNTERS:

Many Microcontroller applications require the counting of external events, such as the frequency of a pulse train, or the generation of precise internal time delays between computer actions. Both of these tasks can be accomplished using software techniques, but software loops for counting or timing keep the processor occupied so that other, perhaps more important functions are not done. To relieve the processor of this burden two 16 bit up counters are provided for the general usage of the programmer. Each counter may be programme to count internal clock pulses acting as a timer ,or programmed to count external pulses as a counter.

The Counters are divided into 2 eight-bit registers called the timer low and high bytes. All counter action is controlled by bit

states in the timer mode control register, the timer/counter control register and certain program instructions.

The 8031 has two 16-bit timer/counter registers. Timer 0 and timer 1. All these can be configured to operate either as a timer or event counters. In the timer function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of oscillator frequency

In the counter function, the register is incremented in response to a 1-to-0 transition at its corresponding input pin, T0, T1, or T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was deducted. Since it takes two machine cycles (24 oscillators periods) to recognize a 1-to-0 transition, the maximum count rate is $1/12$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least for one full machine cycle.

In addition to timer or counter selection, timer 0 and timer 1 have four operating modes from which to select. These timer/counters are present in both 8031 and 8052. The timer or counter function is selected by control bits C/T in the special function

register TMOD. These 2 timer/counters have four operating modes, which are selected by bit pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both timer/counters. Mode 3 is different. The four operating modes are described in the following text.

MODE 0

Either timer in MODE 0 is a 8 bit counter with a divide by 32 prescaler. This 13 bit timer is MCS-48 compatible. In this mode, the Timer register is configured as 13-bit register. As the count rolls over from all 1s to all zero, it sets the timer interrupt flag TFI. The counted input is enabled to the Timer when TR=1 and either GATE=0 or INTI=1 (setting GATE=1 allows the TIMER to be controlled by external input INTI, to facilitate pulse width measurements). TR1 is a control bit in Special Function Register TCON.

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

Mode 0 operation is the same for timer 0 as for timer 1. Substitute TR0, TF0 and INT0 for the corresponding timer 1 signals. There are two different GATE bits, one for timer 1 (TMOD.7) and one for timer 0 (TMOD.3).

MODE 1

Mode 1 similar to mode 0 except TLX is configured as full 8 bit counter when the mode bits are set to 01b in TMOD. The timer flash would be set in 0.1311 secs using 8 mhz crystal.

MODE 2

Setting the mode bits to 10b in TMOD configures the timer to use only the TLX counter as an 8-bit counter. THX is used to hold a value that is loaded in to TLX, every time TLX overflows from FFh to 00h. The timer flag is set when TLX overflows. This mode exhibits an auto reload feature: TLX will count up from the number in THX, over flow, and be initialized again with the contents of THX. For example, placing 9Ch in THX will result in a delay of exactly in 0.0002 secs before the overflow flag is set if a 6 mhz crystal is used.

MODE 3

Timer 0 and 1 may be programmed to be in mode 0,1 or 2 independently of a similar mode for the other timer. This is not true for mode3, The timer do not operate independently if mode 3 is chosen for timer0. Placing timer 1 in mode3 causes it to stop counting, the control bit TR1 and the timer 1 flag TF1 are then used by timer 0.

CHAPTER 3

PROJECT DESCRIPTION

CHAPTER.3

PROJECT DESCRIPTION

The generalized block diagram in fig 3.1 shows the arrangement of microcontroller, interfacing unit and PC. The fig 3.2 shows the hardware arrangement of existing microcontroller unit and the location of 8031 microcontroller.

The speed of the drum is sensed by the sensors and is given to the microcontroller unit that is nothing but a counter, from the speed the microcontroller unit counts the length of the yarn in meters. This microcontroller unit requires manual settings for the operation. But it is Interfaced with the PC and made automatic by this project. This is possible by writing programs on both microcontroller side and PC side.

In the microcontroller side, we are programming the 8031 chips by keeping the existing hardware as it is. But before going into the programming side, the existing hardware is studied first to understand the different technical facts.

3.1 MICROCONTROLLER UNIT:

Description:

The counter receives pulses from the proximity sensor in the pin P1.6. Once the length count reaches the preset value, the preselection relay, RELAY 1 (pin P1.4) is activated which energizes the alarm signal. Similarly when final set value is reached the final set relay, RELAY 2 will be energized (pin P1.5) Pin 1.3 is connected to the security lock switch which is checked when the reset key is pressed. If the switch is closed, the system is reset else the key pressed is ignored.

The software is in the EPROM 2764 whose memory capacity is 2K. The pins (P1.0- P1.7 and P2.0 – P2.4) are used as address lines for 2764. Pins P2.5 – P2.7 is used to obtain the chip select lines for 8279.

Lines RL0 and RL1 of the 8279 is connected to the rows of the matrix keyboard and the output lines (A0 – A3 and B0 – B3) are connected to drive the LED segments through the transistors. The three scan lines are connected to the decoder to generate 8 decoded signals. In this circuit 6 output lines of the decoder are connected as digit drives, to turn on 6 seven segment LEDs. Two output lines are unused. In addition the data lines of 8279 are connected to the data bus of the 8031 and 2764. IRQ of 8279 is connected pin P3.2 of 8031.

Four signals RD, WR, CLK, RESET are connected directly from 8031. The system has 3.072 MHz clock when 8279 is reset, the clock prescaler is set to 31. This divides the clock frequency by 31 to provide the scan frequency of approximately 100 KHz. After the initialization of 8279, the respective codes are sent to the display RAM to display any character. The 8279 takes over the task of displaying the characters by outputting the codes and digit strobes. To read the keyboard, the 8279 scans the columns. If a key closure is detected, it debounces the key. If a key closure is valid it loads the key codes in the FIFO, and the IRQ line goes high to interrupt the system. Also the A1 flag is set, the IRQ line is cleared by the first data read operation and this enables further writing into RAM. For each IC, a tantalum capacitor of value 0.1mf is connected across each to suppress any unwanted noise entering the system.

3.2 INTERFACING UNIT:

The existing microcontroller unit is interfaced with the personal computer through an interfacing unit shown in figure 1. This consists of a transceiver (transmitter/receiver) and a converter. Here we have used RS 232 converter and RS 485 Transceiver. This interfacing unit is connected to the serial port and the personal computer.

3.2.1. TRANSCEIVER:

The RS 485 is a high-speed differential TRI-STATE Bus/Line transceiver with extended common mode range of +12v to -7v, for multipoint data transmission. In addition, it is compatible with RS - 422. The driver and receiver outputs feature TRI-STATE capability, for the driver outputs over the active common mode range of +12v to -7v. A thermal shutdown circuit, which forces the driver outputs into the high impedance state, handles bus contention or fault situations that cause excessive power dissipation within the device.

FEATURES OF RS-485 TRANSCEIVER:

The RS-485 Transceiver has the following features:

- Meets EIA standard RS 485 or multipoint bus transmission and is compatible with RS-422.
- Small outline package option available for minimum board space.
- 22 ns driver propagation delays.
- Single +5V supply.
- -7V to +12V Bus common mode range permits +/- 7V ground difference between devices on the bus.
- Thermal shutdown protection.
- High impedance to bus with driver in TRI-STATE or with power off, over the entire common mode range allows the usual devices on bus to be powered down.
- 70 mv typical receiver hysteresis.

3.2.2. CONVERTOR:

The converter is a multi-channel RS-232 Driver/Receiver and they are mostly used in communication interfaces and in particular for those applications where +/- 12V is not available. Also, this RS-232 is particularly useful in battery-powered systems. Since their low-power shutdown mode reduces power dissipation to less than 5 Micro Watt.

The RSS-232 converter has the following features they are as follows,

- Operate from single +5V power supply.
- Low power receive mode in shutdown.
- Meet all EIA/Ha –232E specifications.
- Multiple Drivers and receivers.
- 3- state Driver and Receiver outputs.
- Open-Line Detection.

Because of these above features, they are used in many applications such as,

- Portable computers.
- Low-power modems.
- Interface translation.
- Battery powered RS-232 systems.
- Multi-drop RS-232 networks.

This Rs-232 converter is connected to the serial port(D9 port) of the PC through which the communication takes place.

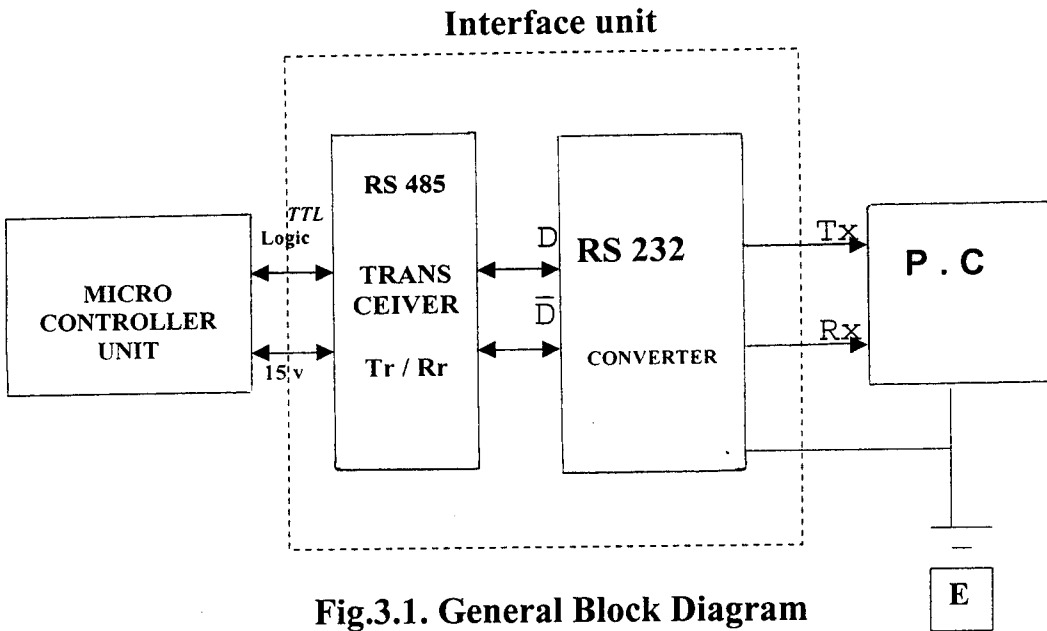


Fig.3.1. General Block Diagram

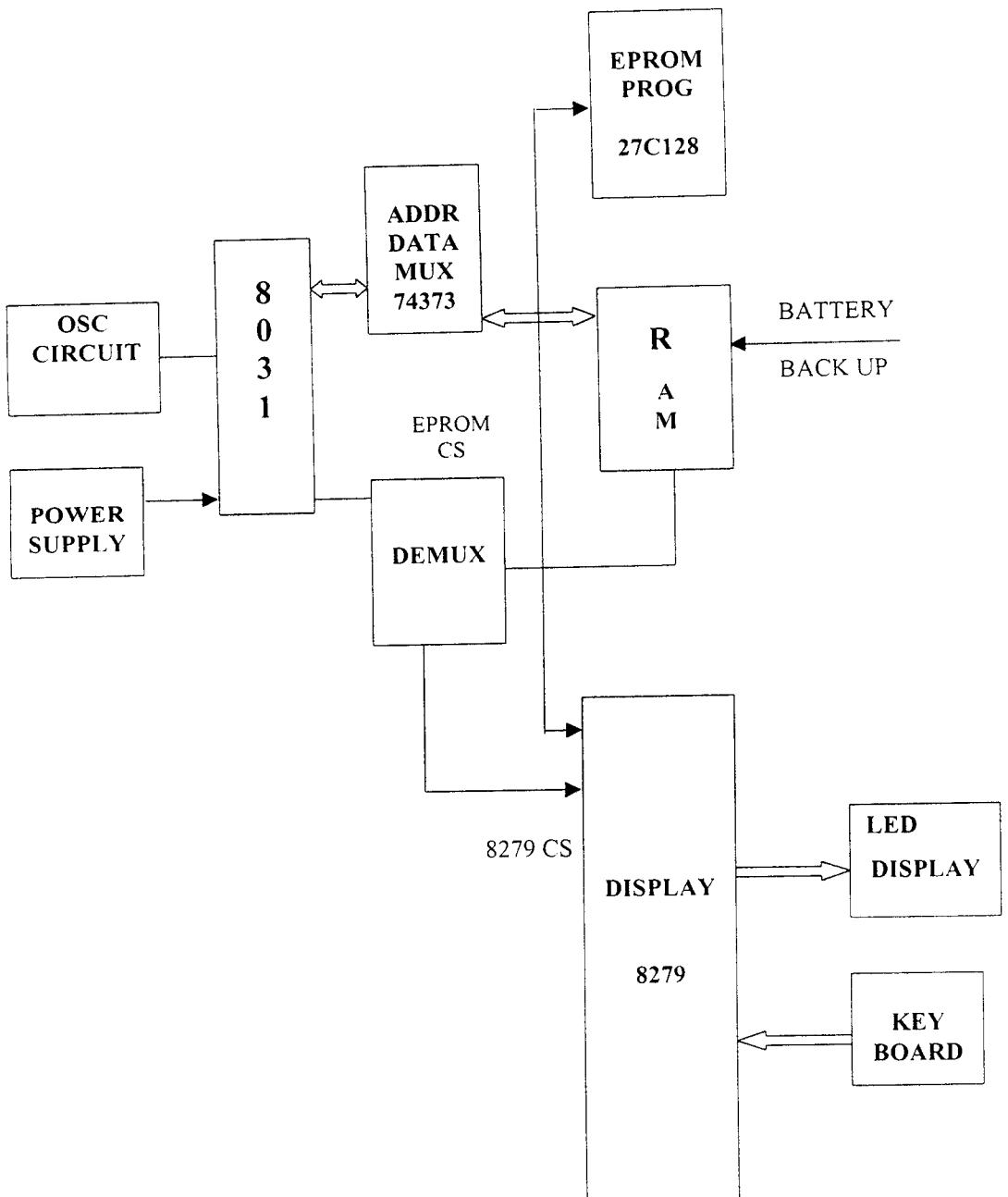


FIG.3.2 BLOCK DIAGRAM OF EXISTING MICROCONTROLLER SYSTEM

CHAPTER 4

SOFTWARE DESCRIPTION

CHAPTER.4

SOFTWARE DESCRIPTION:

4.1 Microcontroller coding:

```
CPU "8031.TBL"
HOF "INT8"
;-----
NOFCHAN:EQU    16D    ; Total number of channels used
TOTCHAN:EQU   NOFCHAN+2
BLCNT:EQU    250D    ; THE BLINKING RATE
;-----
;EQUATES FOR VARIABLES IN EXTERNAL DATA MEMORY
;-----
RAMAD:        EQU    2000H
RAMEND:       EQU    RAMAD+7FFH

RTNRIT:       EQU    RAMAD
RTNLFT:       EQU    RTNRIT+2
RRFTR: EQU    RTNLFT+2    ; To hold left side ratio
LRFTR: EQU    RRFTR+2    ; To hold right side ratio
TOTDRM: EQU    LRFTR+2
TOTDRML: EQU    TOTDRM+2H
SLIPADR: EQU    TOTDRML+1
SCNTADR: EQU    SLIPADR+TOTCHAN
RCNTADR: EQU    SCNTADR+{3*TOTCHAN}
PRAM: EQU    RCNTADR+{4*TOTCHAN}
CODEXT: EQU    PRAM+1    ; 12 bytes are required
BITRADR: EQU    CODEXT+15D
BITLADR: EQU    BITRADR+{{NOFCHAN+2}/2}
DFCNTR: EQU    BITLADR+{{NOFCHAN+2}/2}
CTCNTR: EQU    DFCNTR+2 ; 2BYTES
BYTLADR: EQU    CTCNTR+2
BYTRADR: EQU    BYTLADR+{{NOFCHAN+2}/2}
FDLADR: EQU    BYTRADR+{{NOFCHAN+2}/2}
FDRADR: EQU    FDLADR+{{NOFCHAN+2}/2}
INCLOCL: EQU    FDRADR+{{NOFCHAN+2}/2}
INCLOCR: EQU    INCLOCL+{{NOFCHAN+2}/2}
PASSWORD: EQU    INCLOCR+{{NOFCHAN+2}/2}
SIDE: EQU    PASSWORD+10H
SETPASS: EQU    SIDE+1
TSETPASS: EQU    SETPASS+10    ;4 bytes
;;;
```



```

ADRCM: EQU TSETPASS+10
STSRAM: EQU ADRCM+1
TMPCOM: EQU STSRAM+17

```

```

;-----
; DEFINE PORT ADDRESS FOR KB/D CONTROLLER
;-----

```

```

CMD79: EQU 4001H
DAT79: EQU 4000H

```

```

;-----
; EQUATES FOR BIT ADDRESSABLE VARIABLES
;-----

```

```

P10: EQU 90H
WSIG: EQU P10
LBYRIT: EQU WSIG+1

```

```

STSIN: EQU LBYRIT+1
SCLOK: EQU STSIN+1
PCLOK: EQU SCLOK+1

```

```

STROBE: EQU PCLOK+1
SHCLK: EQU STROBE+1
DOUT: EQU SHCLK+1

```

```

MKEY: EQU 0B5H
CNTL: EQU 0B4H

```

```

DPSIDE: EQU 07FH
RDRM: EQU 07EH
LDRM: EQU 07DH
EDSIDE: EQU 07BH
INVLDBIT: EQU 07AH
BRKBIT: EQU 079H
YESBIT: EQU 078H
SYESBIT: EQU 077H
NVLDBIT: EQU 076H
SBRKBIT: EQU 075H

```

```

PLFLG: EQU 074H ; INDICATION FOR POWER ENTRY FOR FDI

```

```

PRFLG: EQU 073H

```

```

DFLG: EQU 072H ; 00H= LEFT : 01H = RIGHT :FDI

```

```

ZFLG: EQU 071H ; 01 = RCNT IS NOT EQUAL TO 00000

```

```

PASSOK: EQU 06FH

```

```

NP: EQU 6EH ; Used to display digit value in direct drum no. press

```

```

;-----

```

```

GETFLG: EQU 6CH ; Set to put In recieve mode

```

MCNFLG:	EQU	6BH	; Set if recieved m/c address get matched.
TRNSFR:	EQU	6AH	; Set to put in transmit mode
MAST:	EQU	69H	
SCRT:	EQU	68H	
SPIN:	EQU	67H	
DELV:	EQU	66H	
HANK:	EQU	65H	
RRRCLS:	EQU	64H	
REESST:	EQU	63H	
VELOC:	EQU	62H	

ALLPR:	EQU	61H	
INDCAT0:	EQU	60H	
HOUR:	EQU	5FH	
PARAM1:	EQU	5EH	
SBIT:	EQU	5DH	
REVSGL:	EQU	5CH	
CHKBIT:	EQU	5BH	
RPMBIT:	EQU	5AH	

TBIT:	EQU	59H	
REVENBL:	EQU	58H	
REVBIT:	EQU	57H	
PARAM2:	EQU	56H	
PARAM3:	EQU	55H	
PARAM4:	EQU	54H	
PARAM5:	EQU	53H	
UPFLG:	EQU	51H	
DOWNFLG:	EQU	50H	
BUSY:	EQU	4FH	
SPDABRT:	EQU	4EH	
ALTVIEW:	EQU	4DH	
TEMP2:	EQU	4CH	
INVLDBYTE:	EQU	4BH	
PASS2OK:	EQU	4AH	
RESET:	EQU	49H	
RCLOSE:	EQU	48H	
STXFLG:	EQU	47H	

; CODES USED FOR SERIAL COMMUNICATION

EOT:	EQU	4AH	; End Of Transmission code
STATUS:	EQU	1AH	; System STATUS code
EOC:	EQU	7AH	; End of m/c communication
STX:	EQU	6AH	; Start of communication
SACK:	EQU	4FH	; Acknowledge with address byte

```

SSTAT: EQU 1AH ; Send system status
RT2PARA: EQU 5EH
RSLSPD: EQU 4DH
RDOFFP: EQU 4EH
BRKCNTC: EQU 26H
RXD: EQU 0B0H
TXD: EQU 0B1H
T2CON: EQU 0C8H
RCAP2H: EQU 0CBH
RCAP2L: EQU 0CAH
TR2: EQU 0CAH

```

```

;-----
;DEFINE ADDRESS FOR INTERNAL DATA MEMORY
;-----

```

```

OBUFF: EQU 30H ; Contain data that is displayed
DRAM: EQU OBUFF+8 ; Holds code for display
IBUFF: EQU DRAM+8 ; Holds key code pressed last
RPNTR: EQU IBUFF+1 ; Pointer for real time count
SPNTR: EQU RPNTR+2 ; Pointer for set count
TEMP: EQU SPNTR+2 ; Temporary location
TEMP1: EQU TEMP+1 ; Scratch pad in main rtn
FLAG: EQU TEMP1+1 ; Holds status of display
POINTER: EQU FLAG+1 ; Holds addr. of slip loc.
SCRATCH: EQU POINTER+2 ; Used in intr routines
SETVAL: EQU SCRATCH+1 ; A scratch for set count/slip
ASKPRAM: EQU SETVAL+3 ; To hold count for auto skip
BLRAM: EQU ASKPRAM+1 ; Blinking rate counter
DPCHNL: EQU BLRAM+1 ; Channel being displayed
SUBCHNL: EQU DPCHNL+1 ; Temp. channel storage
MSGF: EQU SUBCHNL+1 ; Holds status of msg display
SUBF: EQU MSGF+1 ; Sub flag used to run within selected menu
YNCNTR: EQU SUBF+1 ; To hold count for FULL-PACK
FLCNTR: EQU YNCNTR+2 ; To hold count for YARN-CUT
CNT: EQU FLCNTR+2 ; Temp. counter
DUMMY: EQU CNT+1 ; Temp. used during scanning
LCHNUM: EQU DUMMY+1 ; Right side total drum
LCHNUML: EQU LCHNUM+1 ; Left side total drum
LKY: EQU LCHNUML+1 ; Used in non. ejection routine for side
LKY1: EQU LKY+1 ; " " " for chnl. no.
NEPTR: EQU LKY1+1 ; Used for addr. storage
GST1: EQU NEPTR+2 ; group starting channel
GEN1: EQU GST1+1 ; group ending channel
CHKSUM: EQU GEN1+1
SUMBYT: EQU CHKSUM+1

```

```

;-----
;DEFINE CONSTANTS USED IN VARIOUS ROUTINE

```

```

;-----
SKIPKY:    EQU  0AH        ; Skip to previous stage
NXTKY:     EQU  0BH        ; Confirmation key
UNITKY:    EQU  0CH        ; To increment chnl number in units
LBYRKY:    EQU  0DH        ; Select the side of channel
INDVKY:    EQU  0EH        ; To select a function for particular chnl
TENKY:     EQU  0FH        ; To increment chnl number in tens
SELKY: EQU  10H            ; To select a function from main menu
UNIVKY:    EQU  11H        ; To select a function for all chnls
PSTN1: EQU  90H            ; Display start position
NULL: EQU  00H
EMPTY:    EQU  80H        ; Ibuff empty flag
PCODE: EQU  66H            ; Power failure code
DLPCNT:    EQU  05H        ; Parallel pulse width
BLICNT:    EQU  025D       ; Blink once in every .25s
DOT: EQU  08H

```

```

;-----
; IE0 = 0003H
; IE1 = 0013H
; TF0 = 000BH
; TF1 = 001BH
;-----

```

```

ORG 0H
SJMP WRMSTRT ; Poweron entry point

```

```

ORG 03H
LJMP RDRMISR ; Right side drum sense service routine

```

```

ORG 0BH
LJMP KBDISR ; Key board interrupt service routine

```

```

ORG 13H
LJMP LDRMISR ; Left side drum sense service routine

```

```

ORG 1BH
RETI

```

```

ORG 23H
LJMP SSISR

```

```

,*****
*****
,*****
*****
,*****
*****

```

PROGRAM STARTS HERE

```

ORG 30H

```

```

WRMSTRT:
MOV  P1,#0FFH      ; Define port1
MOV  SP,#07H       ; Initialise stack

MOV  R0,#20H
WLP2:
MOV  R1,#0FFH
WLP3:
DJNZ R1,WLP3
DJNZ R0,WLP2

MOV  DPTR,#CMD79   ; Define mode of opn for 8279
MOV  A,#00H        ; Encoded scan, 2key lockout
MOVX @DPTR,A       ; 8 digit display, left entry
MOV  A,#3FH
MOVX @DPTR,A       ; Program clock by xx
SETB ES
,*****
,*****
*****
,*****INITIALISATION          OF          USER          DEFINED
,PARAMETERS*****
,*****
*****
SETB PCLOK
SETB PCLOK

CLR  STROBE
CLR  STROBE
CLR  DOUT
CLR  DOUT

CLR  LDRM
CLR  RDRM

CLR  DPSIDE
MOV  DPTR,#SIDE
MOVX A,@DPTR
JZ   SIDE10
MOV  A,#0FFH
MOVX @DPTR,A
SETB DPSIDE          ;Initialisation of parameters
SIDE10: CLR NP
MOV  DPCHNL,#01H
MOV  SUBCHNL,#01H

```

```
MOV FLAG,#NULL
MOV MSGF,#NULL
MOV SUBF,#NULL
```

```
MOV TEMP1,#01H
MOV BLRAM,#BLCNT
MOV DPTR,#DFCNTR
MOVX A,@DPTR
MOV FLCNTR,A
INC DPTR
MOVX A,@DPTR
MOV FLCNTR+1,A
```

```
MOV DPTR,#CTCNTR
MOVX A,@DPTR
MOV YNCNTR,A
INC DPTR
MOVX A,@DPTR
MOV YNCNTR+1,A
```

```
CLR INVLDBIT
CLR BRKBIT
CLR SBRKBIT
CLR YESBIT
CLR SYESBIT
CLR NVLDBIT
```

```
LCALL INITDRM
LCALL INITDRML
MOV DPTR,#TOTDRM
MOVX A,@DPTR
MOV LCHNUM,A
MOV DPTR,#TOTDRML
MOVX A,@DPTR
MOV LCHNUML,A
```

```
SETB PLFLG
SETB PRFLG
```

```
MOV LKY,#00H
MOV TEMP1,#01H
MOV LKY1,#00H
MOV GST1,#00
MOV GEN1,#00
MOV DPTR,#PASSWORD
MOV A,#0
```

```
MOVX @DPTR,A
CLR PASSOK
```

```
LCALL LTRCHK
LCALL RTRCHK
```

```
CLR STROBE
CLR STROBE
```

```
LCALL UPDTDPL ; Display run message
LCALL ENBLINT
```

```
.*****
,
*****
; SCANNING OF DRUMS STARTS HERE
.*****
,
*****
```

```
MERGE1:
```

```
CLR STROBE
LCALL UPDTDPL
JB LDRM,SCANL1 ; Check for completion of left ratio
JB RDRM,SCANR1 ; Check for completion of right ratio
SJMP MERGE1
```

```
MERGE2:
```

```
CLR STROBE
LCALL UPDTDPL
JB RDRM,SCANR1
JB LDRM,SCANL1
SJMP MERGE2
```

```
SCANR1:
```

```
LJMP SCANR
```

```
SCANL1:
```

```
LJMP SCANL
```

```
;-----
```

```
SSISR: PUSH DPH
PUSH DPL
PUSH A
PUSH PSW
PUSH B
PUSH 5
PUSH 0
PUSH 7
PUSH 1
```

```
RECIV: JNB RI,RECIV ; Wait till sbuff is filled
```

```

MOV  A,SBUF          ; save in B register
CLR  RI
MOV  B,A

JB   STXFLG,SSISR10 ; STX already recieved
CJNE A,#STX,SQUIT   ; Set STXFLG if STX recieved
SETB STXFLG
SJMP SQUIT

SSISR10:
JB   MCNFLG,SELMCN ; See if m/c is already selected
MOV  R0,A           ; adr. recieved
MOV  DPTR,#ADPMC
MOVX A,@DPTR
CJNE A,00,SQUITA ; compare with the m/c addr. sent from
SETB MCNFLG       ; pc and set m/c select flag
CLR  GETFLG
CLR  TRNSFR
LJMP SRLQIT
SQUITA: CLR STXFLG
SQUIT: LJMP SQUIT1

SELMCN: CJNE A,#7AH,SEL1; END(7A) of this machine communi
CLR  STXFLG
LJMP SQUIT1          ; clear control flags and go out

SEL1: CJNE A,#3AH,SEL2; Code (3A) to reset the machine
SETB GETFLG
CLR  TRNSFR
SETB RESET
LJMP NXTQIT
SEL2: CJNE A,#3BH,SEL3; Code (3B) to rclose emergncy
SETB GETFLG
CLR  TRNSFR
SETB RCLOSE
LJMP NXTQIT
SEL3: CJNE A,#3EH,PROCEED
CLR  PASSOK
CLR  PASS2OK
CLR  STXFLG
LJMP SQUIT1

SELNOT: LJMP RES1      ; refuse since SECURITY is ON

PROCEED: JB GETFLG,RCVPT ; if not see if reception or
JB TRNSFR,TRAES ; transmission is enabled

```



```
MOV   A,B           ; Code (5A) to check for trns or recp
CJNE  A,#5AH,RDYTRS ; not equal go to transmit
LJMP  RESPT         ; Equal go to receive
```

```
RDYTRS:   CLR  GETFLG
SETB  TRNSFR
CLR    CNTL
LJMP  SRLQIT
RCVPT:
```

```
JB    MAST,SOMST
JB    VELOC,SOVEL
JB    ALLPR,SOALL
JB    SCRT,SOSCR
JB    SPIN,SOSPN
JB    PARAM1,SOPARA1
JB    PARAM2,SOPARA2
JB    PARAM3,SOPARA3
JB    PARAM4,SOPARA4
JB    PARAM5,SOPARA5
JB    RESET,SORESET
JB    RCLOSE,SORCLOSE
LJMP  RECIPT
```

```
SOMST:    LJMP GETMST
SOVEL:    LJMP GETVEL
SOALL:    LJMP GETALL
SOSCR:    LJMP GETSCR
SOSPN:    LJMP GETSPN
SOPARA1:  LJMP  GETPAR1
SOPARA2:  LJMP  GETPAR2
SOPARA3:  LJMP  GETPAR3
SOPARA4:  LJMP  GETPAR4
SOPARA5:  LJMP  GETPAR5
SORESET:  LJMP  GETRESET
SORCLOSE: LJMP  GETRCLOSE
```

```
TRAES: MOV  A,B
CJNE  A,#SSTAT,SRLPRG ; Code to send system status
MOV   A,#67H
CLR   C
ORL   C,PASSOK
ORL   C,PASS2OK
JNC   TM10
MOV   A,#68H
TM10: MOV  DPTR,#STSRAM+12
MOVX  @DPTR,A
```

```

MOV    A,OBUFF+2
CJNE  A,#LLETR,TM20
MOV    DPTR,#STSRAM
MOV    A,#0CH
MOVX   @DPTR,A
TM20: JNB  BUSY,TM30
SETB  SPDABRT
SJMP  TM40
TM30:
MOV    DPTR,#STSRAM+4
MOVX   @DPTR,A
MOV    DPTR,#STSRAM+5
MOVX   @DPTR,A
TM40: MOV  DPTR,#STSRAM+16
MOV    A,#4AH
MOVX   @DPTR,A
MOV    CHKSUM,#00
MOV    DPTR,#STSRAM
MOVX   @DPTR,A
LCALL  PAUSE      ;***
LCALL  PAUSE
LJMP   SRLSTAT

```

```

SRLPRG:  MOV  A,B
CJNE  A,#SPGM1,SL2PRG
; MOV  DPTR,#TOTBYT
MOVX  A,@DPTR
LCALL  CHK_VALPGM_BYTES
JNB   INVLDBYTE,TM45
MOV   A,#75
TM45: MOV  SUMBYT,A
SETB  CNTL
MOV   CHKSUM,#00
; MOV  DPTR,#SRLONE
REPET: LCALL  PAUSE      ;***
LCALL  PAUSE      ;***
LCALL  PAUSE
LCALL  PAUSE
LCALL  TRPRG
; LCALL  PAUSE
LCALL  PAUSE
CLR   CNTL
CLR   STXFLG
LJMP  SRLQIT

```

```

SL2PRG: MOV  A,B

```

```
JNB    INVLDBYTE, TM50
MOV    A, #75D
TM50:
MOV    SUMBYT, A
SETB   CNTL
; MOV  DPTR, #SRLTWO
MOV    CHKSUM, #00
LJMP   REPET
```

```
SL1RT1:    MOV  A, B
LJMP   REPET
```

```
SL2RT2:    MOV  A, B
SETB   CNTL
; MOV  DPTR, #DLVRTIO
MOV    CHKSUM, #00
LJMP   REPET
```

```
SRLTM1:
MOV    A, B
; MOV  DPTR, #T1RTIO
MOV    CHKSUM, #00
LJMP   REPET
```

```
SRLTM2:
MOV    A, B
SETB   CNTL
; MOV  DPTR, #T2RTIO
MOV    CHKSUM, #00
LJMP   REPET
```

```
SRLDF:
MOV    A, B
SETB   CNTL
; MOV  DPTR, #D1RTIO
MOV    CHKSUM, #00
LJMP   REPET
```

```
SL3RT3:    MOV  A, B
SETB   CNTL
; MOV  DPTR, #HNKRTIO
MOV    CHKSUM, #00
LJMP   REPET
```

```
SL4RT4:
MOV    A, B
MOVX   A, @DPTR
; MOV  DPTR, #SLTRAM
MOVX   @DPTR, A
```

```

MOV  CHKSUM,#00
LJMP REPET
SUIT1A:    CLR  STXFLG
SUIT1:    LJMP SQUIT1
SL5RT5:    MOV  A,B
CJNE  A,#SACK,SUIT1A    ; Code to send acknowledge
MOV  SUMBYT,#01H
SETB  CNTL
MOV  DPTR,#ADPMC
MOVX  A,@DPTR
; MOV  DPTR,#ACK
; MOVX  @DPTR,A
MOV  CHKSUM,#00
LJMP  REPET
SRLSTAT:
MOV  DPTR,#STSRAM
LCALL      PAUSE
LCALL PAUSE
SETB  CNTL
LCALL PAUSE
LCALL PAUSE
LCALL      TRANSD
LCALL      PAUSE
CLR  CNTL
CLR  STXFLG
LJMP  SRLQIT
GETSCR60A:LJMP  GETSCR60
GETSCR:
LCALL QUEUPA
MOV  A,DUMMY
JNZ  GETSCR60A
DEC  DPL
MOVX  A,@DPTR
XRL  CHKSUM,A
CJNE  A,#0A4H,GETSCR5
MOV  A,#4AH
GETSCR5:CJNE  A,CHKSUM,GETSCR70
MOV  DPTR,#TMPCOM+1
MOVX  A,@DPTR
MOV  R7,A
ADD  A,A
ADD  A,R7
INC  A
MOV  R7,A    ; No of bytes to be copied

; MOV  DPTR,#CODE

```

```

MOVX A,@DPTR
CJNE A,#01H,GETSCR30
MOVX @DPTR,A
INC DPTR
MOV R0,DPH
MOV R5,DPL ; destination area

```

```

GETSCR10:MOV DPTR,#TMPCOM+1
MOV TEMP1,DPH
MOV TEMP1+1,DPL ; source area

```

```

GETSCR20:MOV DPH,TEMP1
MOV DPL,TEMP1+1
MOVX A,@DPTR
INC DPTR
MOV TEMP1,DPH
MOV TEMP1+1,DPL
MOV DPH,R0
MOV DPL,R5
MOVX @DPTR,A
INC DPTR
MOV R0,DPH
MOV R5,DPL
DJNZ R7,GETSCR20
CLR SCRT
CLR STXFLG
LJMP SQUIT1

```

```

GETSCR30:
CJNE A,#02H,GETSCR40
MOVX @DPTR,A
INC DPTR
MOV R0,DPH
MOV R5,DPL ; destination area
SJMP GETSCR10

```

```

GETSCR40:
CJNE A,#03H,GETSCR50
MOVX @DPTR,A
INC DPTR
MOV R0,DPH
MOV R5,DPL ; destination area
SJMP GETSCR10

```

```

GETSCR50:
CJNE A,#04H,GETSCR70
MOVX @DPTR,A
INC DPTR
MOV R0,DPH

```

```

MOV R5,DPL ; destination area
SJMP GETSCR10
GETSCR60:LJMP SRLQIT
GETSCR70:
CLR SCRT
MOV CHKSUM,#00
CLR STXFLG
LJMP SQUIT1
GETSPN: LCALL QUEUPA
MOV A,DUMMY
JNZ GETSPN10
DEC DPL
MOVX A,@DPTR
XRL CHKSUM,A
CJNE A,#0A4H,GETSPN5
MOV A,#4AH
GETSPN5:
CJNE A,CHKSUM,GETSPN20
MOV DPTR,#TMPCOM
MOVX A,@DPTR
MOV R0,A ; MSB
INC DPTR
MOVX A,@DPTR ; LSB
; MOV DPTR,#RTIOLOC+1
MOVX @DPTR,A
DEC DPL
XCH A,R0 ; R0 - LSB , A - MSB
MOVX @DPTR,A
MOV A,R0
INC DPTR
MOVX @DPTR,A
CLR SPIN
CLR STXFLG
LJMP SQUIT1
GETSPN10:
LJMP SRLQIT
GETSPN20:
MOV CHKSUM,#00
CLR SPIN
CLR STXFLG
LJMP SQUIT1

GETPAR1:LCALL QUEUPA
MOV A,DUMMY
JNZ GETPAR110
DEC DPL

```

```

MOVX A,@DPTR
XRL  CHKSUM,A
CJNE A,#0A4H,GETPAR15
MOV  A,#4AH
GETPAR15:
CJNE A,CHKSUM,GETPAR120
MOV  DPTR,#TMPCOM
MOVX A,@DPTR
MOV  R0,A
INC  DPTR
MOVX A,@DPTR
; MOV DPTR,#T1RTIO+1
MOVX @DPTR,A
DEC  DPL
XCH  A,R0
MOVX @DPTR,A
; MOV DPTR,#T1RBAK
MOVX @DPTR,A
INC  DPTR
MOV  A,R0
MOVX @DPTR,A
CLR  PARAM1
CLR  STXFLG
LJMP SQUIT1
GETPAR110:
LJMP SRLQIT
GETPAR120:
MOV  CHKSUM,#00
CLR  PARAM1
CLR  STXFLG
SJMP SQUIT1

```

```

SQUIT1:      CLR  GETFLG
CLR  MCNFLG
CLR  TRNSFR

```

```

SRLQIT:      POP  1
POP  7
POP  0
POP  5
POP  B
POP  PSW
POP  A
POP  DPL
POP  DPH
RETI

```

```

GETPAR2:LCALL  QUEUPA
MOV  A,DUMMY
JNZ  GETPAR210
DEC  DPL
MOVX A,@DPTR
XRL  CHKSUM,A
CJNE A,#0A4H,GETPAR25
MOV  A,#4AH
GETPAR25:
CJNE A,CHKSUM,GETPAR220
MOV  DPTR,#TMPCOM
MOVX A,@DPTR
MOV  R0,A
INC  DPTR
MOVX A,@DPTR
; MOV DPTR,#T2RTIO+1
MOVX @DPTR,A
DEC  DPL
XCH  A,R0
MOVX @DPTR,A

; MOV DPTR,#T2RBAK
MOVX @DPTR,A
INC  DPTR
MOV  A,R0
MOVX @DPTR,A

CLR  PARAM2
CLR  STXFLG
LJMP SQUIT1
GETPAR210:
LJMP SRLQIT
GETPAR220:
MOV  CHKSUM,#00
CLR  PARAM2
CLR  STXFLG
LJMP SQUIT1

GETPAR3:LCALL  QUEUPA
MOV  A,DUMMY
JNZ  GETPAR310
DEC  DPL
MOVX A,@DPTR
XRL  CHKSUM,A
CJNE A,#0A4H,GETPAR35

```



```

MOV  A,#4AH
GETPAR35:
CJNE A,CHKSUM,GETPAR320
MOV  DPTR,#TMPCOM
MOVX A,@DPTR
MOV  B,A
INC  DPTR
MOVX A,@DPTR
; MOV DPTR,#D1RTIO+1
MOVX @DPTR,A
DEC  DPL
MOV  A,B
MOVX @DPTR,A
CLR  PARAM3
CLR  STXFLG
LJMP SQUIT1
GETPAR310:
LJMP SRLQIT
GETPAR320:
MOV  CHKSUM,#00
CLR  PARAM3
CLR  STXFLG
LJMP SQUIT1

GETPAR4:LCALL  QUEUPA
MOV  A,DUMMY
JNZ  GETPAR410
DEC  DPL
MOVX A,@DPTR
XRL  CHKSUM,A
CJNE A,#0A4H,GETPAR45
MOV  A,#4AH
GETPAR45:
CJNE A,CHKSUM,GETPAR420
MOV  DPTR,#TMPCOM
MOVX A,@DPTR
MOV  B,A
INC  DPTR
MOVX A,@DPTR
; MOV DPTR,#SLRAM
MOVX @DPTR,A
INC  DPTR
MOV  A,B
MOVX @DPTR,A
CLR  PARAM4
CLR  STXFLG

```

```

LJMP  SQUIT1
GETPAR410:
LJMP  SRLQIT
GETPAR420:
MOV   CHKSUM,#00
CLR   PARAM4
CLR   STXFLG
LJMP  SQUIT1
;,,,,,,,,;TO BE CHNAGED FOR CHECKSUM
GETPAR5:RET

```

```

TRNS1:
LCALL PAUSE
LCALL TRNOUT
INC   DPTR
TRANSD:
MOVX  A,@DPTR
XRL   CHKSUM,A
CJNE  A,#EOT,TRNS1
MOV   A,CHKSUM
XRL   A,#4AH
CJNE  A,#4AH,TRANSD10
MOV   A,#0A4H
TRANSD10:
LCALL PAUSE
LCALL TRNOUT
LCALL PAUSE
MOV   A,#4AH
LCALL PAUSE
LCALL TRNOUT
RET

```

```

TRNOUT:    MOV  SBUF,A
TROUT:    JNB  TI,TROUT
CLR  TI
RET

```

```

PAUSE:MOV  R3,#07H
PAUS:  MOV  R5,#0EFH
PAUS1:DJNZ R5,PAUS1
DJNZ  R3,PAUS
RET

```

```

TPROG:
LCALL PAUSE
LCALL TRNOUT

```

```

INC DPTR
TRPRG:
MOVX A,@DPTR
XRL CHKSUM,A
DJNZ SUMBYT,TPROG
LCALL PAUSE
LCALL TRNOUT
LCALL PAUSE
MOV A,CHKSUM
CJNE A,#4AH,TRPRG10
MOV A,#0A4H
TRPRG10:LCALL PAUSE
LCALL TRNOUT
LCALL PAUSE
MOV A,#4AH
LCALL PAUSE
LCALL TRNOUT
RET

```

```

;*****
;*****

```

```

INITIAL:MOV PCON,#80H
MOV SCON,#50H
MOV TMOD,#21H
MOV TH1,#0F3H
SETB TR1
RET

```

```

SENDIN:
LCALL PAUSE
LCALL TRNOUT
INC DPTR

```

```

SEND: CLR A
MOVC A,@A+DPTR
CJNE A,#EOT,SENDIN
MOV A,#4AH
LCALL PAUSE
LCALL TRNOUT
RET

```

```

GRPMSG:
BLANK,BLANK,BLANK,GLETR,RSLETR,OLETR,ULETR,PLETR
GRP1MSG: DFB BLANK,GLETR,BLANK,0,0,DASH,0,0

```

DFB

;G 00-

00

GPRCNTMSG:	DFB
BLANK,BLANK,GLETR,PLETR,RSLETR,CLETR,NSLETR,TLETR	
DRMLMSG:	DFB
DASH,DLETR,RSLETR,DASH,LLETR,DASH,BLANK,BLANK	
DRMMSG:	DFB
DASH,DLETR,RSLETR,DASH,RSLETR,DASH,BLANK,BLANK	
PASSMSG:	DFB
PLETR,DLETR,BLANK,BLANK,BLANK,BLANK,BLANK,BLANK	
; PD msg	
GRSTMSG:	DFB
BLANK,BLANK,GLETR,RSLETR,ESLETR,SLETR,ESLETR,TLETR	;GReSet
msg	

```

DISYES:
MOV  DPTR,#YESMSG
LCALL LOAD
LCALL ENCODE
LCALL DISPLAY
RET
;-----
END

```

CHAPTER 5

**INTRODUCTION TO VISUAL
BASIC**

CHAPTER. 5

Introduction to visual basic:

Visual basic is a powerful programming system for developing sophisticated, graphical applications for Microsoft windows environment. Its productivity has been enhanced by addition of a complete set of tools to simplify rapid application development.

“Visual” refers to the method used to create the graphical user interface (GUI), that uses illustrations, rather than writing numerous lines of code to describe the appearance, function and location of interface elements. “Basic” refers to the BASIC programming language, a widely preferred language by many programmers for its simplicity. Visual basic has evolved from the original BASIC language and now contains several hundred statements, functions and keywords, many of which relate directly to the windows GUI.

Visual Basic offers many salient features to aid in the development of full-featured applications including:

- Data access functionality allows creation of front-end applications that can work on most of the popular database systems.
- ActiveX technology allows usage of the functionalities provided by other applications, such as Microsoft Word, Microsoft Excel, and other windows applications and their possible deployment on the web.
- Access to documents and applications across the Internet from within your application is made easier through Internet capabilities.

- Applications developed using visual basic provide a true .EXE file that uses a run time dynamic link library (DLL), which can be freely distributed.
- Calling powerful API functions available in Visual Basic optimizes application performances.

Visual basic is an ideal programming language for developing sophisticated professional applications for Microsoft Windows. It makes use of Graphical user interface for creating robust and powerful applications. The graphical user interface as the name suggests illustrates to text, which enable users to interact with an application. This feature makes it easier to comprehend things in quicker and an easier way.

Coding in a GUI environment is quite transition to traditional, linear programming methods where the user is guided through a linear path of execution and is limited to a set of operations. In a GUI environment, the number of options open to the user is much greater, allowing more freedom to the user and the developer. Features such as easier comprehension, user friendliness, faster application development and many other aspects such as introduction to ActiveX technology and Internet features make Visual Basic an interesting work tool to work with.

Visual Basic was developed from the basic programming language. In 1970's Microsoft got it start by developing ROM based interpreted BASIC for the early microprocessor based computers. In 1982 Microsoft quick Basic revolutionized basic and legitimized as a series development language

for MS-DOS environment. Later on, Microsoft corporation created the enhanced version of BASIC called Visual Basic for Windows.

5.1. Developing an application:

To create an application with Visual Basic, we work with project. A project is a collection of files that are used to build an application. Writing an Visual Basic program involves two steps

- Visual programming step
- Code programming step

Visual programming step involves designing an application With various tools that come along with visual Basic package. The code-programming step involves writing programs using a text editor.

Visual basic uses building blocks such as variables, data types, procedures, functions and control structures in its programming environment. Code in Visual Basic is stored in the form of modules. The three kinds of modules are form modules, standard modules and class modules. In this project we have used standard modules.

CHAPTER 6

FEATURES OF VISUAL BASIC-6

CHAPTER. 6

FEATURES OF VISUAL BASIC-6.

6.1 Data base concepts in visual basic:

Nearly all business applications need to store large volumes of data, organized in a format that simplifies retrieval. This is accomplished with a database management system (DBMS), a mechanism for manipulating tabular data with high level commands. The database management system hides low-level details such as how data are stored in a database, and frees the programmer to concentrate on managing information, rather than on the specifics of manipulating files or maintaining links among them.

Visual Basic provides a wealth of tools for creating and accessing databases on both individual machines and networks. The two major tools are

- The data control.
- The data access object.

The data control gives access to databases without any programming. Few properties of the control can be set and regular controls such as textboxes can be used to display the values of the fields in the database. This is the no code approach to database programming, which is implemented quite nicely in Visual Basic.

The data access object is a structure of objects for accessing databases through the code. All the functionality of the data control is also available in the code, through the data access object (DAO). A database is simply a grouping of related

information organized for easy processing. The actual data in a database is stored in tables, which are similar to random access files. Data in a table is made up of columns and rows. The rows contained identically structured pieces of information, which are equivalent to the records of random access files. A record is a collection of values (called Fields).

6.2. Record sets:

Record sets are objects that represent collections of records from one or more tables. Record sets are equivalent of variables in regular programming. The tables of a database cannot be accessed directly. The only way to view or manipulate records is via Record Set objects. A Record Set is constructed of columns and rows and is similar to a table, but it can contain data from multiple tables.

A Record Set, therefore, is a view of some of the data in the database, selected from the database according to user-specified criteria. The three types of Record Sets are:

- Dynasets, which are updatable views of data.
- Snapshots, which are static (read-only) views of data.
- Tables, which are direct views of tables.

Dynasets are updated every time user changes the database, and changes they make to the corresponding Record Sets are reflected in the underlying tables. Snapshots are static views of the same data. A Snapshot contains the records requested the moment the Snapshot was generated and Snapshots cannot be updated.

The Dynaset is the most flexible and powerful type of Recordset, although a few operations may be faster with the table RecordSets

6.3 MSComm Control :

The MSComm control provides serial communications for your application by allowing the transmission and reception of data through a serial port.

The MSComm control provides the following two ways for handling communications:

- Event-driven communications is a very powerful method for handling serial port interactions. In many situations where it is to be notified the moment an event takes place, such as when a character arrives or a change occurs in the Carrier Detect (CD) or Request To Send (RTS) lines. In such cases, the MSComm control's OnComm event is used to trap and handle these communications events. The OnComm event also detects and handles communications errors.
- Events and errors can be polled by checking the value of the CommEvent property after each critical function of your program. This may be preferable if your application is small and self-contained.

Each MSComm control corresponds to one serial port. If we need to access more than one serial port in our application, we must use more than one MSComm control. The port address and interrupt address can be changed from the Windows Control Panel.

Although the MSComm control has many important properties, there are a few that you should be familiar with first. The CommEvent property contains the numeric code of the actual error or event that generated the OnComm event. Note that setting the Rthreshold or Sthreshold properties to 0 disables trapping for the comEvReceive and comEvSend events, respectively. The OnComm event is generated whenever the value of the CommEvent property changes, indicating that either a communication event or an error occurred.

CHAPTER 7

**FRONTEND DESIGN USING
VISUAL BASIC**

CHAPTER. 7

Frontend design using VB:

The data accessing through PC is facilitated through the frontend tool visual Basic. Visual Basic is just not a language. It is an integrated development environment in which you can develop, run, test and debug your applications. Data project is a feature of the enterprise edition, and it doesn't corresponds to a new project type. It is identical to standard EXE project type, but it automatically adds the controls that are used in accessing Databases to the toolbox. The toolbox contains the icons of the controls, which you can place on a Form to create an applications user, interface. Here in this project we make use of user-defined controls to browse through the databases.

The Recordset placeholder is an object variable that represents an updatable Recordset object to which you want to add a new record. We use the Add New method to create and add a new record in the Recordset object named by *record set*. This method sets the fields to default values, and if no default values are specified, it sets the fields to null (the default values specified for a table-type Recordset).

After you modify the new record, use the Update method to save the changes and add the record to the Recordset. No changes occur in the database until you use the Update method.

If we issue an Add New and then perform any operation that moves to another record, but without using Update, the changes are lost without warning. In addition, if we close the Recordset or end the procedure that declares the Recordset or its Database object, the new record is discarded without warning.

When we use Add New in a Microsoft Jet workspace and the database engine has to create a new page to hold the current record, page locking is pessimistic. If the new record fits in an existing page, page locking is optimistic.

If we haven't moved to the last record of your Recordset, records added to base tables by other processes may be included if they are positioned beyond the current record. If we add a record to our own Recordset, however, the record is visible in the Recordset and included in the underlying table where it becomes visible to any new Recordset objects.

The position of the new record depends on the type of Recordset:

- In a dynaset-type Recordset object, records are inserted at the end of the Recordset, regardless of any sorting or ordering rules that were in effect when the Recordset was opened.
- In a table-type Recordset object whose Index property has been set, records are returned in their proper place in the sort order. If you haven't set the Index property, new records are returned at the end of the Recordset.

The record that was current before you used AddNew remains current. If you want to make the new record current, you can set the Bookmark property to the bookmark identified by the LastModified property setting.

Note To add, edit, or delete a record, there must be a unique index on the record in the underlying data source. If not, a "Permission denied" error will occur on the AddNew, Delete, or Edit method call in a Microsoft Jet workspace, or an "Invalid argument" error will occur on the Update call in an ODBCDirect workspace.

CHAPTER 8

DESCRIPTION OF THE PROJECT

CHAPTER. 8

Description of the project:

This automation project is aimed at serving to the fast needs of the textile industry. This interfacing system is designed with VB as frontend and microprocessor as the interfacing unit. The data storage is facilitated by the usage of MS-access as the backend unit. While working with data in a Microsoft Access database, first a connection to a database file is created. The easiest way to create a connection to a Microsoft Access file is to create a data environment using the Data Environment designer. A Data Environment designer provides an easy way to create connections to many types of databases.

The databases are accessed through DAO environment facilitated by the use of codings for the data retrieval window. The recordset type of data is maintained with the MS-access. The interrupt from the microcontroller is verified through comm control in VB and the data interrupt is feedback to the microcontroller for further processing. The sensor in the machine collects the data from the machine and interrupts to the microcontroller, which is interfaced to the computer by the interfacing unit.

The measurement of the length of the yarn is measured by the counting the rotations of the main drum thereby, the yarn intake is calculated. The yarn take-up of each drum

is updated depending on the yarn running signal available from the sensor of the particular drum. This counting continues till the package reaches the preset length. If there is no yarn in the drum, then the counting process is halted and it resumes only when the yarn is being wound.

When the preset length is reached on the particular drum, the centralized counter actuates the lifting mechanism or sends cutting signal to stop the further take-up of yarn, along with full doff indication. The main advantage of this automation project is that it indicates all the production details. The drums are installed with the sensors for measuring the yarn intake. The microcontroller based preset length measurement is already in existence. This microcontroller-based system has to be initialized manually for each and every drum. By this automation project these parameters can be automatically set from the PC alone and individual setting for the machines can be eliminated.

The automation process is maintained in two different levels through two individual forms, namely:

Form1: This form asks the user to specify the machine number to which the parameters are interrupted. It sends an interrupt signal to the microcontroller sensors to check whether the machine is ready or not. When an OK signal is received from the machine the microcontroller is instructed to activate the sensors for receiving data from the machine.

Form2: When the machine is ready for interruption, the parameters are set for each and every drum and the data signals are send back to the microcontroller for updating the preset parameters required for interrupting the yarn-winding machine. Thus the parameters for the drums are updated even at runtime and manipulated. These parameters are stored in the form of recordsets for production details scheduling.

Each and every recordsets can be browsed through the browser controls set with individual seek commands. These recordsets are linked with the date control by which daily schedules for the yarn lengths are taken into production details. The interfacing unit just interrupts with the computer for retrieving information from the machine. The slip factors for drums are programmed with the microcontroller. This automation device monitors up to 200 drums. And it can handle two different speeds.

Individual drum settings are the main feature of this automation project. The production details of all the machines are maintained within the database, which is maintained with MS-access.

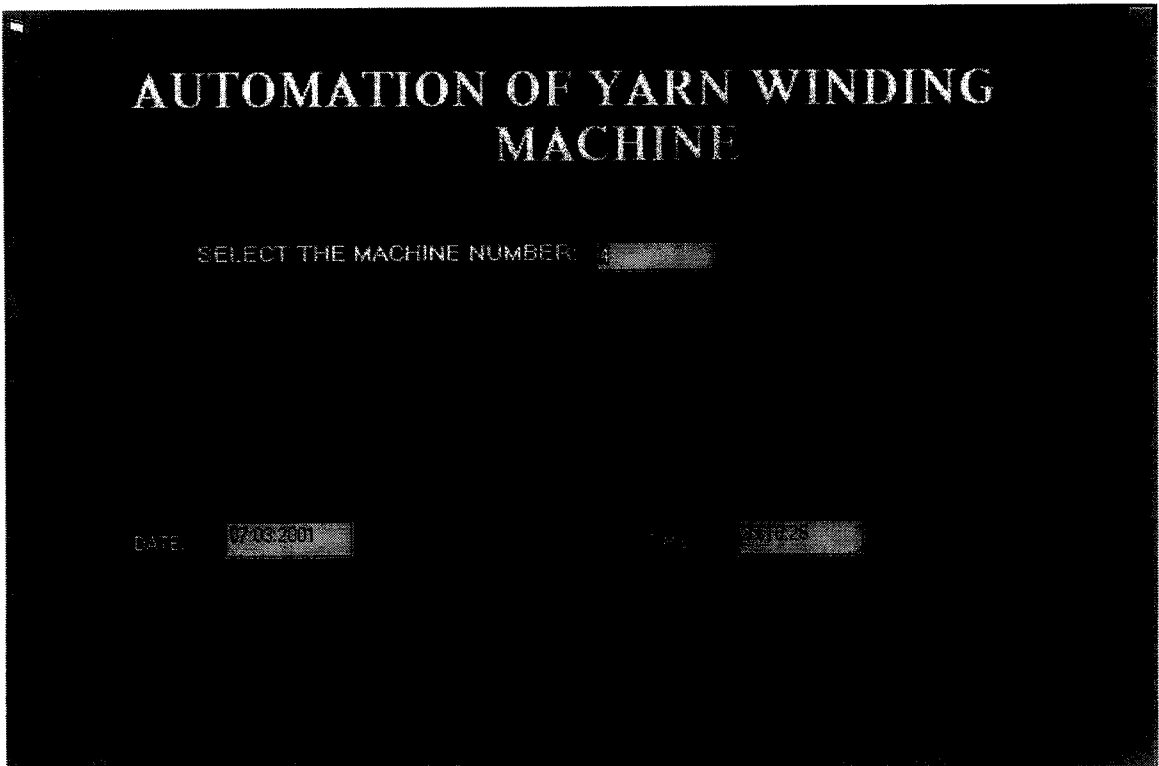
CHAPTER 9

SOFTWARE

CHAPTER. 9

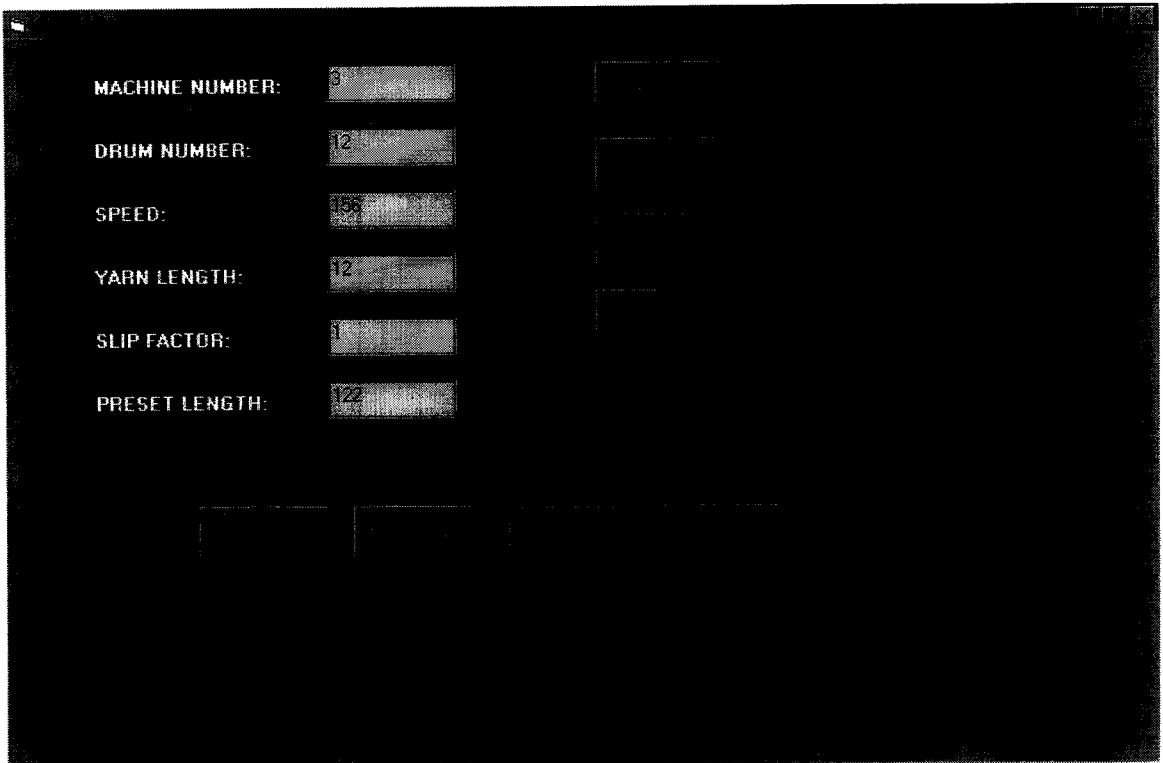
9.1. Form window:

Entering window:



The image shows a screenshot of a software window with a black background and white text. The title of the window is "AUTOMATION OF YARN WINDING MACHINE" in a large, bold, serif font. Below the title, there is a prompt "SELECT THE MACHINE NUMBER:" followed by a small rectangular input field. At the bottom of the window, there are two labels: "DATE:" and "TIME:". The "DATE:" label is followed by a rectangular input field containing the text "17-03-2001". The "TIME:" label is followed by a rectangular input field containing the text "05:10:28".

Data retrieval window:

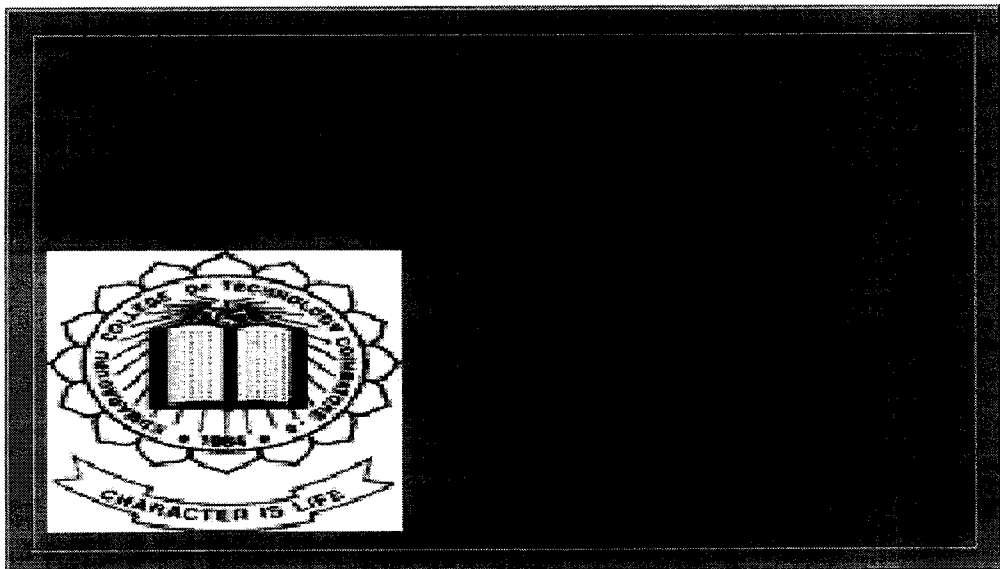


A screenshot of a software window titled "Data retrieval window". The window has a dark background and contains several input fields on the left side, each with a label and a corresponding text box. The labels and values are as follows:

Label	Value
MACHINE NUMBER:	3
DRUM NUMBER:	12
SPEED:	158
YARN LENGTH:	12
SLIP FACTOR:	1
PRESET LENGTH:	122

Below these input fields, there are several buttons and a large empty rectangular area, likely for displaying data or performing operations.

Splash screen:



9.2. Project coding:

The coding for the data transmission window is as follows:

Form Module coding:

```
Public mcno As Integer
Public s As String
Public a As Variant
Public rs As Recordset
Public db As Database
Public MSComm1 As MSComm

' ***** COMMAND WORDS *****
Public Const SEND_STATUS = &H1A
Public Const SEND_PROGRAM1 = &H1B
Public Const SEND_PROGRAM2 = &H1C
Public Const RX_COMMANDS = &H5A
Public Const RX_RESET = &H3A
Public Const CONFIRM_RESET = &HA3

Public Const RX_CLOSE = &H3B
Public Const CONFIRM_CLOSE = &HB3

Public Const RX_CUR_TIME = &H2A
Public Const RX_CUR_STEP = &H3C

Public Const RX_PROGRAM1 = &H1B
Public Const RX_PROGRAM2 = &H2B

Public Const CODE0 = &H0
Public Const CODE1 = &H1
Public Const PRESENT_OR_NOT = &H4F

Public Const START_COMMUNICATION = &H6A
Public Const END_SEQUENCE = &H4A ' Marks end of command sequence*/
Public Const END_TX_FROM_PC = &H7A 'Marks end of transmission from PC*/
' ***** END OF COMMAND WORDS *****
Const RX_PROGRAM = &H2B
Const STX = &H6A
Const MCNADDR = &H1
Const TRANSMIT = &H5F
Const RECIEVE = &H5A
Const ACK = &H4F
Option Explicit
Public data As Integer
```

```

Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Const SEND_PGM = &H1B
Const RCV_PGM = &H2B
Const STATUS = &H1A
Const ETX = &H4A 'Marks end of command sequence*/
Const EOT = &H7A 'Marks end of transmission from PC*/
Public Function send_a_char(data)
'Form2.MSComm1.InBufferSize = 20
' 1024 default
If (Form2.MSComm1.PortOpen) Then
    Form2.MSComm1.PortOpen = False
End If
Form2.MSComm1.PortOpen = True
Form2.MSComm1.RTSEnable = True

Dim i As Integer
Form2.MSComm1.RTSEnable = False
Form2.MSComm1.Output = Chr(data)
For i = 0 To Form2.Text1.Count - 1

'Form2.Text1(i).Text = " " + Hex(data)
Next i
DoEvents
Sleep (5)
Form2.MSComm1.RTSEnable = True
DoEvents
End Function
Public Function SendProgram(recno As Integer) As Integer

    Form2.MSComm1.InBufferSize = 20 ' 1024 default
If (Form2.MSComm1.PortOpen) Then
    Form2.MSComm1.PortOpen = False
End If
Form2.MSComm1.PortOpen = True
Form2.MSComm1.RTSEnable = True
    Dim NoSteps As Integer, i As Integer, dummy As Integer
    Dim byte1 As Integer, byte2 As Integer, byte3 As Integer
    Dim CheckSum As Byte, Reply As Variant
    NoSteps = i
    send_a_char (Chr(EOT))
    CheckSum = 0
    CheckSum = NoSteps Xor 0
    send_a_char (Chr(STX))
    send_a_char (Chr(MCNADDR)) 'M/c address
    send_a_char (Chr(RECIEVE))
    send_a_char (Chr(RX_PROGRAM))

```

```

send_a_char (Chr(0))
send_a_char (Chr(NoSteps)) 'Send No. of bytes as it is DO NOT CONVERT to
BCD
For i = 1 To NoSteps
    dummy = dec2bcd(TempProg2(i).step, , , byte1)
    dummy = dec2bcd(TempProg2(i).time, , byte2, byte3)
    send_a_char (Chr(byte1)) 'Send STEP
    GoToSleep (2)
    send_a_char (Chr(byte2)) 'Send TIME - 1st byte
    send_a_char (Chr(byte3)) 'Send TIME - 2nd byte
    CheckSum = CheckSum Xor byte1 Xor byte2 Xor byte3
Next i
send_a_char (Chr(CheckSum))
send_a_char (Chr(ETX)) ' Marks end of command sequence*/
send_a_char (Chr(EOT)) 'End Sequence
send_a_char (Chr(EOT)) 'End Sequence
GoToSleep (5)
Form2.MSComm1.InputLen = 0
Reply = Form2.MSComm1.Input
GoToSleep (5)
End Function
Public Function dec2bcd(inbyte As Integer, Optional ByRef byte1 As Integer, Optional
ByRef byte2 As Integer, Optional ByRef byte3 As Integer)
Dim item%
    Dim mystr$, a1$, a2$, a3$
    mystr = Format(inbyte, "000000")
    a1$ = Mid$(mystr, 1, 2)
    a2$ = Mid$(mystr, 3, 2)
    a3$ = Mid$(mystr, 5, 2)
    item = Val(a1$)
    byte1 = (item \ 10) * 16 + item Mod 10
    item = Val(a2$)
    byte2 = (item \ 10) * 16 + item Mod 10
    item = Val(a3$)
    byte3 = (item \ 10) * 16 + item Mod 10
End Function
Public Function Reset()
    Form2.MSComm1.InBufferSize = 20 ' 1024 default
If (Form2.MSComm1.PortOpen) Then
    Form2.MSComm1.PortOpen = False
End If
Form2.MSComm1.PortOpen = True
Form2.MSComm1.RTSEnable = True
    Dim CheckSum As Byte
    CheckSum = RX_RESET Xor CONFIRM_RESET
    Form2.MSComm1.InputLen = 0

```

```

send_a_char (STX)
send_a_char (1) 'M/c address
send_a_char (&H5F) 'RX_RESET_EMERGENCY) 'Rx_Reset_Emergency
send_a_char (RX_RESET) ' Rx_Reset
send_a_char (CONFIRM_RESET) ' Rx_Reset
send_a_char (Checksum)
Sleep (5)
send_a_char (ETX) ' Marks end of command sequence*/
send_a_char (EOT) 'End Sequence
End Function

```

Data Transmission window coding:

```

Const STX = &H6A
Const MCNADDR = &H1
Const TRANSMIT = &H5F
Const RECIEVE = &H5A
Const ACK = &H4F
Const SEND_PGM = &H1B
Const RCV_PGM = &H2B
Const STATUS = &H1A
Const ETX = &H4A
Const EOT = &H7A
'-----
' Definition of Global Constants
Const MAX_STEPS = 25 ' for program
Const MAX_FRAMES = 100 ' for frames - not significant
Dim Current(MAX_FRAMES, MAX_STEPS) As Program, TempProg1(MAX_STEPS)
As Program, TempProg2(MAX_STEPS) As Program
' Definition of different structures
Private Type Program
    step As Integer
    time As Integer
End Type
Dim mcaddr%

Public db As dao.Database
Public rs As dao.Recordset
Public rs1 As dao.Recordset
Public flag As Boolean

Private Sub Combo1_Click()

'Text2(0).Text = Combo1.Text

End Sub

```

```

Private Sub Combo1_KeyPress(KeyAscii As Integer)
Combo1.Text = ""
Select Case KeyAscii
Case 65 To 90
MsgBox ("invalid entry")
Combo1.Text = ""
Form1.Show
Combo1.Text = ""
Case 97 To 122
Combo1.Text = ""
MsgBox ("invalid entry")
Form1.Show
End Select
End Sub

```

```

Private Sub Command1_Click()

```

```

mcno = Combo1.Text
Text2(0).Text = mcno

```

```

Dim flag As Boolean
'MSComm1.Output = Text2(0).Text
Form2.Show
'End If

```

```

End Sub

```

```

Private Sub Form_Load()
MSComm1.InBufferSize = 20 ' 1024 default
If (MSComm1.PortOpen) Then
MSComm1.PortOpen = False
End If
'MSComm1.PortOpen = True
Form1.MSComm1.RTSEnable = True

```

```

Dim db As dao.Database
Dim rs As dao.Recordset
Combo1.AddItem "1", 0
Combo1.AddItem "2", 1
Combo1.AddItem "3", 2
Combo1.AddItem "4", 3
Combo1.AddItem "5", 4

```

```

Combo1.AddItem "6", 5
Combo1.AddItem "7", 6
Combo1.AddItem "8", 7
Combo1.AddItem "9", 8
Combo1.AddItem "10", 9

```

```

Set db = dao.Workspaces(0).OpenDatabase("c:\kar.mdb ")
Set rs = db.OpenRecordset("drumno")
'App.Path
If rs.EditMode = dbEditAdd Then
rs.Update
End If

```

```

MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"
MSComm1.InputLen = 0
'If MSComm1.PortOpen = True Then
'MsgBox (" port is open")
'Else:
'MSComm1.PortOpen = True

```

```

'End If
End Sub

```

```

Private Sub Timer1_Timer()
Text1(0) = Format(Date, "dd:mm:yyyy")
Text1(1) = Format(time, "hh:mm:ss")
End Sub

```

Data retrieval window coding:

```

Const STX = &H6A
Const MCNADDR = &H1
Const TRANSMIT = &H5F
Const RECIEVE = &H5A
Const ACK = &H4F
Const SEND_PGM = &H1B
Const RCV_PGM = &H2B
Const STATUS = &H1A
Const ETX = &H4A
Const EOT = &H7A
'-----
' Definition of Global Constants
Const MAX_STEPS = 25 ' for program
Const MAX_FRAMES = 100 ' for frames - not significant

```

```
Dim Current(MAX_FRAMES, MAX_STEPS) As Program, TempProg1(MAX_STEPS)
As Program, TempProg2(MAX_STEPS) As Program
' Definition of different structures
```

```
Private Type Program
    step As Integer
    time As Integer
End Type
```

```
Dim mcaddr%
Dim db As dao.Database
Dim rs As dao.Recordset
Dim flag As Boolean
Dim i As Integer
Dim a As Variant
Private Sub Command1_Click()
'With rs
'Set db = dao.Workspaces(0).OpenDatabase("c:\kar.mdb")
'Set rs = db.OpenRecordset("drumno")
rs.MoveLast
rs.AddNew
For i = 1 To Text1.Count - 1
If Val(Text1(1).Text) >= 128 Then
MsgBox ("invalid entry")
Text1(1).Text = ""
Text1(1).SetFocus
Else: Text1(2).SetFocus
End If
If Val(Text1(2).Text) >= 1000 Then
MsgBox ("invalid entry")
Text1(2).Text = ""
Text1(2).SetFocus
Else: Text1(3).SetFocus
End If
If Val(Text1(3).Text) >= 99999 Then
MsgBox ("invalid entry")
Text1(3).Text = ""
Text1(3).SetFocus
Else: Text1(4).SetFocus
End If
If Val(Text1(4).Text) >= 100 Then
MsgBox ("invalid entry")
Text1(4).Text = ""
Text1(4).SetFocus
Else: Text1(5).SetFocus
```

```

End If
If Val(Text1(5).Text) >= 99999 Then
MsgBox ("invalid entry")
Text1(5).Text = ""
Text1(5).SetFocus
Else: Command1.SetFocus
End If
Next i
For i = 0 To Text1.Count - 1
rs(i) = Val(Text1(i).Text)
'Text1(i).Text = ""
Next i
'rs.Edit
If rs.EditMode = dbEditAdd Then
rs.Update
'End With
End If
End Sub
Private Sub Command10_Click()

```

```

Dim a As Variant, b As Integer
MSComm1.InputLen = 0
chk_ack
End Sub

```

```

Private Sub Command2_Click()
rs.Edit
For i = 1 To Text1.Count - 1
rs(i) = Val(Text1(i).Text)
Next i
rs.Update
End Sub

```

```

Private Sub Command3_Click()
MSComm1.InputLen = 0
lbl2 = ""
rcv_pgm1

```

```

'For i = 1 To Text1.Count - 1
'Text1(i) = ""
'Next i
'Dim m As Integer
'Dim a As Integer

```

```

MSComm1.InputLen = 0
'a = MSComm1.Input
'm = LenB(a)

```



```
'For i = 0 To m - 1
'Text1(i).Text = Text1(i) + Hex(a(i))
'Next i
End Sub
```

```
Private Sub Form2_Load()
Text1(0).Text = mcno
End sub
```

```
Private Sub Command4_Click(Index As Integer)
Unload Me
End Sub
```

```
Private Sub Command5_Click()
rs.MoveFirst
rs.Edit
For i = 1 To Text1.Count - 1
Text1(i).Text = rs(i).value
Next i
End Sub
```

```
Private Sub Command6_Click()
rs.MovePrevious
If rs.BOF Then
MsgBox "First Record"
Else
For i = 1 To Text1.Count - 1
Text1(i).Text = rs(i).value
Next i
End If
End Sub
```

```
Private Sub Command7_Click()
rs.MoveNext
If rs.EOF Then
MsgBox "End of the Record"
Else
'rs.Edit
For i = 1 To Text1.Count - 1
Text1(i).Text = rs(i).value
Next
End If
End Sub
```

```
Private Sub Command8_Click()
rs.MoveLast
```

```

For i = 1 To Text1.Count - 1
Text1(i).Text = rs(i).value
Next i
End Sub

```

```

Private Sub Form_Load()
Dim find As Integer
MSComm1.CommPort = 2
  MSComm1.Settings = "9600,N,8,1"
  MSComm1.InputLen = 0
Text1(0).Text = mcno
Text1(0).Enabled = False
Set db = dao.Workspaces(0).OpenDatabase("c:\kar.mdb")
Set rs = db.OpenRecordset("drumno")
find = Text1(0).Text
If Text1(0).Text = "mcno='text1(o).text'" & find & "" Then
'rs.Edit
rs.FindFirst find
MSComm1.InBufferSize = 20 ' 1024 default
If (MSComm1.PortOpen) Then
  MSComm1.PortOpen = False
End If
MSComm1.PortOpen = True
MSComm1.RTSEnable = True
  For i = 0 To Text1.Count - 1
Text1(i).Text = rs(i)
Next i
  With rs
Set db = dao.Workspaces(0).OpenDatabase("c:\kar.mdb")
Set rs = db.OpenRecordset("drumno")
  .MoveLast
  .FindFirst find
  If .NoMatch Then
    MsgBox "No records found with "

  Else
Do
  DoEvents
Loop Until MSComm1.InBufferCount >= 2
' Read the "OK" response data in the serial port.
For i = 0 To Text1.Count - 1
MSComm1.Output = Text1(i).Text
Next i
a(99) = MSComm1.Input
' Close the serial port.
MSComm1.PortOpen = False

```

```

'Dim k As Integer
'Dim t As Integer
'Dim c As Integer
'With rs
'c = Val(Text1(1).Text)
'For k = 0 To c - 1
'For t = 1 To 5
'.MoveLast
'.AddNew
'For i = 0 To MSComm1.InputLen - 1
'Text1(i).Text = a(i)
'rs(i) = Text1(i).Text
'Next i
.Update
If .EditMode = dbEditAdd Then
.Update
End If
'rs.Update
'Next t
'Next k
End If
End With
End If
End Sub

```

```

Private Sub Text1_Change(Index As Integer)
If Val(Text1(3).Text) >= Val(Text1(5).Text) Then
Text1(3).ForeColor = &HFF&
Else
Text1(3).ForeColor = &H0&
End If
End Sub

```

```

Private Function chk_ack() As Integer
'MSComm1.InBufferSize = 20 ' 1024 default
If (MSComm1.PortOpen) Then
MSComm1.PortOpen = False
End If
MSComm1.PortOpen = True
MSComm1.RTSEnable = True

```

```

Dim a As Variant
send_a_char (STX)
send_a_char (MCNADDR)
send_a_char (TRANSMIT)
send_a_char (ACK)

```

```

Sleep (50)
send_a_char (ETX)
send_a_char (EOT)
MSComm1.InputLen = 0
a = MSComm1.Input
i = LenB(a)
Dim m As Integer
For i = 0 To i - 1
For m = 3 To Text1.Count - 1
    Text1(3).Text = " " + Hex(a(i))
Next m
Next i
End Function

```

```

Private Function rcv_status() As Integer
MSComm1.InBufferSize = 20 ' 1024 default
If (MSComm1.PortOpen) Then
    MSComm1.PortOpen = False
End If
MSComm1.PortOpen = True
MSComm1.RTSEnable = True

```

```

Dim a As Variant, Speed As Single
send_a_char (STX)
send_a_char (MCNADDR)
send_a_char (TRANSMIT)
send_a_char (STATUS)
Sleep (100)
send_a_char (ETX)
send_a_char (EOT)
MSComm1.InputLen = 0
MSComm1.InputLen = 0
a = MSComm1.Input
i = LenB(a)
Dim m As Integer
For i = 0 To i - 1
For m = 3 To Text1.Count - 1
    Text1(m).Text = " " + Hex(a(i))
Next i
Next m

```

```

"
'item = Val(a(13)) 'Current step delay in mins.
'item1 = Val(a(14))
'value = (item \ 16) * 10 + (item Mod 16)
'value1 = (item1 \ 16) * 1000 + (item1 Mod 16) * 100

```

```

'TIMELEFT = value1 + value
'Text2 = TIMELEFT
'''
'CURSTEPNO = Val(a(15))
'Text4 = CURSTEPNO
,
'''
" Current SET SPEED is evaluated here
'item1 = Val(a(11))
,
'setSpeed = ((item1 \ 16) * 10 + (item1 Mod 16)) * 100 + 8000
'Text3.Text = setSpeed
,
" MEASURED SPEED
'Speed = 100 + CInt((a(4)) * 100 + CInt(a(5)) * 10)
'Text1.Text = Speed
End Function
Private Function rcv_pgm1() As Integer
'MSComm1.InBufferSize = 20 ' 1024 default
If (MSComm1.PortOpen) Then
    MSComm1.PortOpen = False
End If

MSComm1.PortOpen = True
MSComm1.RTSEnable = True

Dim l%, i%, item%, item1%, value%, value1%, tmp$
Dim inp, Strng As Variant, tmpstr As Variant, tmpchr As Integer
Dim CheckSum As Integer, RxdCheckSum As Integer
Dim j As Integer, ENDTRANS As Integer
    Strng = Array()

MSComm1.InputLen = 0
tmpstr = MSComm1.Input
send_a_char (STX)
send_a_char (1) 'M/c address
send_a_char (TRANSMIT) ' Rx_Reset
send_a_char (SEND_PGM)
Sleep (100)
MSComm1.InputLen = 0
MSComm1.InputLen = 0
a = MSComm1.Input

i = LenB(a)
Dim m As Integer

```

For m = 0 To i - 3

```
Text1(3).Text = " " + Hex(a(m))  
Text1(4).Text = "" + Hex(a(m + 1))  
Text1(5).Text = "" + Hex(a(m + 2))
```

Next m

```
send_a_char (ETX) ' Marks end of command sequence*/  
send_a_char (EOT) 'End Sequence  
End Function
```

Public Function SendProgram() As Integer

```
Dim NoSteps As Integer, i As Integer, dummy As Integer  
Dim byte1 As Integer, byte2 As Integer, byte3 As Integer  
Dim CheckSum As Byte, Reply As Variant
```

```
TempProg2(0).step = 10
```

```
TempProg2(0).time = 3
```

```
TempProg2(1).step = 2
```

```
TempProg2(1).time = 1
```

```
TempProg2(2).step = 3
```

```
TempProg2(2).time = 14
```

```
TempProg2(3).step = 4
```

```
TempProg2(3).time = 18
```

```
TempProg2(4).step = 5
```

```
TempProg2(4).time = 100
```

```
TempProg2(5).step = 0
```

```
TempProg2(5).time = 100
```

```
For i = 0 To 25
```

```
    If (TempProg2(i).step <= 0) Then Exit For
```

```
Next i
```

```
NoSteps = i
```

```
send_a_char (ETX)
```

```
CheckSum = 0
```

```
CheckSum = NoSteps Xor CODE0
```

```
send_a_char (STX)
```

```
send_a_char (1) 'M/c address
```

```
send_a_char (RECIEVE) 'Rx_Reset_Emergency
```

```
send_a_char (RCV_PGM)
```

```
send_a_char (2)
```

```
send_a_char (NoSteps) 'Send No. of bytes as it is DO NOT CONVERT to BCD
```

```
For i = 0 To NoSteps
```

```
    dummy = dec2bcd(TempProg2(i).step, , , byte1)
```

```
    dummy = dec2bcd(TempProg2(i).time, , byte2, byte3)
```

```
    send_a_char (byte1) 'Send STEP
```

```
    Sleep (2)
```

```

    send_a_char (byte2) 'Send TIME - 1st byte
    send_a_char (byte3) 'Send TIME - 2nd byte*
    CheckSum = CheckSum Xor byte1 Xor byte2 Xor byte3
Next i
send_a_char (CheckSum)
send_a_char (ETX) ' Marks end of command sequence*/
send_a_char (ETX) 'End Sequence
send_a_char (EOT) 'EndSequence
Sleep (5)
MSComm1.InputLen = 0
Reply = MSComm1.Input
Sleep (5)
' RESET MACHINE
Text1(i).Text = "      "
Reset
Reset
    Reset
' Module1.SendProgram (recno)
End Function

```

CHAPTER 10

CONCLUSION

CHAPTER. 10

CONCLUSION

The project has been successfully completed and tested for any errors. The system was found to give the best results. The expected accuracy of the project is of the order of 1% error under normal conditions. Thus the wastage of yarn can be reduced drastically. Skilled labor is not required since the process is made automatic.

This automation of yarn winding machine includes the following advantages.

- ✓ Multiplexed control.
- ✓ Individual drum settings.
- ✓ Indicates all production details.
- ✓ Reduced number of operating personals.
- ✓ Remote control of the entire machine.

FUTURE PROSPECTS:

In this project we have done interfacing with fixed number of drums. But this can be developed further such that more number of drums can be controlled.

REFERENCES

REFERENCES

1. Handbook of 16-bit Microcontroller.
Intel corporation ltd, 1989 USA,.
2. Interface Data book.
National semiconductor.
1990 edition.
3. Maxim Data book, volume-4.
New releases- 1995
4. Mastering Visual basic 6.
Evangelos Petroustos
BPB publications.
5. www.microcontrollers.com
6. www.vbonline.com

APPENDIX

APPENDIX

ARCHITECTURE OF 8031

The fig.1 shows the internal architecture of 8031 microcontroller. The standard functions, which make up a microprocessor, are in the center of the diagram. These include the ALU, accumulator, stack pointer, a block of registers and general purpose registers-the B register. All of these devices are connected to the 8031 internal 8-bit Data bus.

The functions of the Special function registers are outlined below:

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

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

Program status word: In order that the flags may be conveniently addressed, they are grouped inside the program status word and the power control registers. The PSW contains the math flags, user program flags F0, and the register select bits that identify which of the four general purpose register bank is currently in use by the program.

Stack pointer: The stack pointer register is 8-bits wide. It is incremented before data is stored during PUSH and CALL instructions. While the stack may reside anywhere in on chip RAM, the stack pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

Data pointer: The data pointer consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-Bit register or as a two independent 8-Bit register.

Serial data buffer: The serial data buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. When data is moved from SBUF, it comes from the receive buffer.

Timer registers: Register pairs (TH0, TLO), (TH1, TL1) and (TH2, TL2) are the 16-bit counting registers for timer/counters 0,1 and 2 respectively.

Capture registers: The register pairs (RCAP2H, RCAP2L) are capturing registers for the timer2 capture mode.

Control registers: Special function registers IP, IE , TMOD, TCON, T2CON, SCON, PCON contain controlled status bits for the interrupt system, the timer/counters, and the serial port.

Pin Description of 8031:

The pin configuration is shown in fig 2.

Vcc - Supply voltage.

Vss - Circuit ground potential.

Port 0 - It is an 8-Bit Bi-Directional I/O port. It serves as the multiplexed low order Address and Data bus during access to external memory.

Port 1 - It is an 8-Bit Bi-Directional I/O port. They are used as follows:

P1.0 - Reset of 8279.

P1.1 - Rotary switches.

P1.2 - Rotary switches.

P1.3 - Security lock.

P1.4 - Preset relay.

P1.5 - Finalset relay.

P1.6 -Sensing pulse.

P1.7 - Not used.

Port 2 - It is an 8-Bit Bi-Directional I/O port. It emits the high-order address byte during access to external memory that uses 16-Bit address

Port 3 - It is an 8-bit Bi-Directional I/O port. It serves the functions of various special features of the MCS-51 family as given below.

P3.0 – R X D (Serial input port)

P3.1 – T X D (Serial output port)

P3.2 – INT0 (External interrupt zero)

P3.3 – INT1 (External interrupt one)

P3.4 – T0 (Timer Zero external input)

P3.5 – T1 (Timer one external input)

P3.6 – WR (external data memory write strobe)

P3.7 – RD (external data memory read strobe)

RST - Reset input :

A reset is accomplished by holding the reset pin high for two machine cycles while the oscillator is running.

ALE/PROG – Address latch enable:

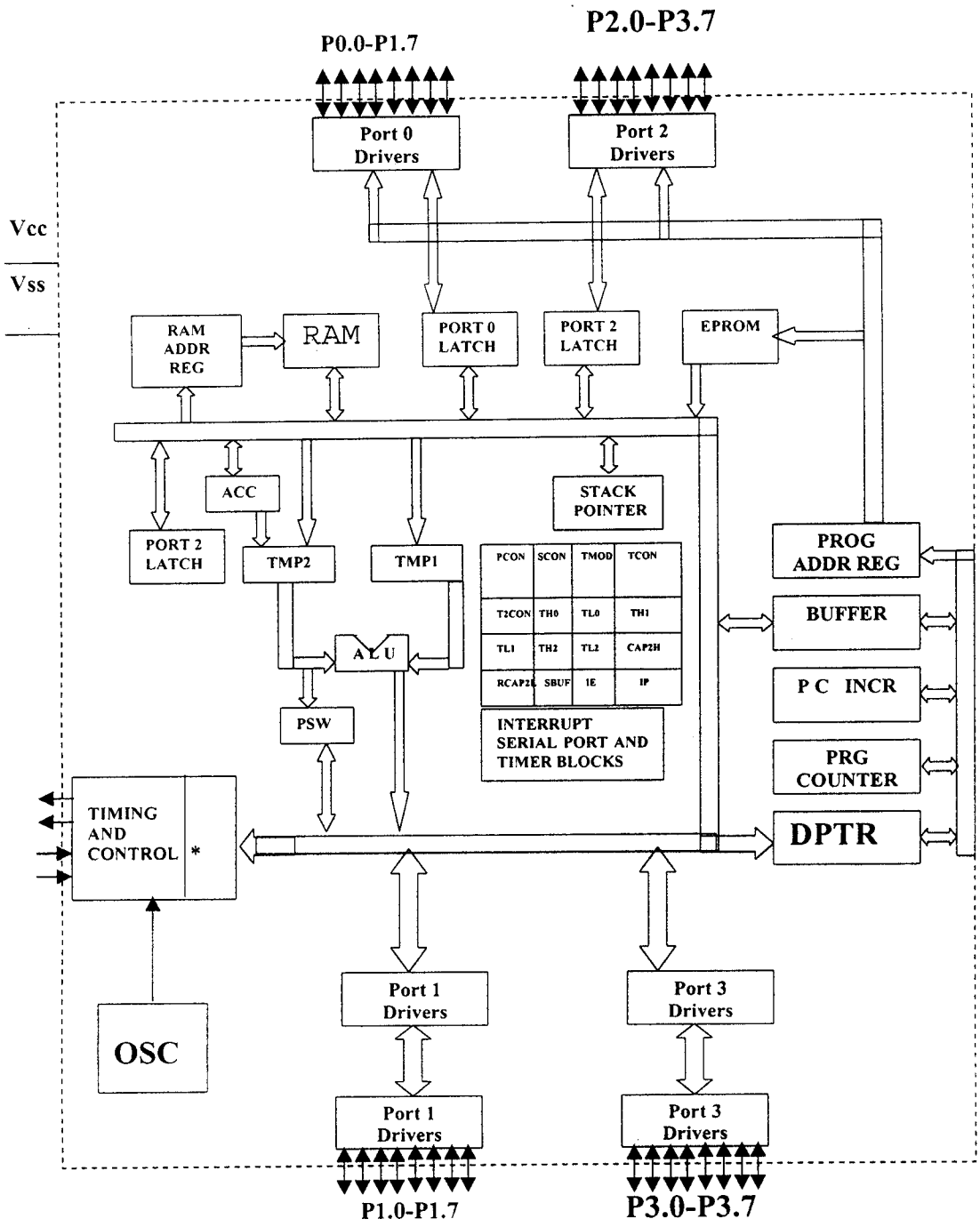
Output pulse for latching the low byte of the address during access to external memory. This pin is also the program pulse input during EPROM programming.

PSEN – program store enable is the read strobe to the external program memory. When the device is executing out of external program memory PSEN is activated twice each machine cycle.

EA/VPP – In 8031 EA should be externally wired low as the central processing unit executes out of external program memory.

XTAL1 – It is the input to the oscillators high gain amplifier, which is intended, for use as a crystal or external source can be used.

XTAL2 – It is the output from the oscillators amplifier which is required when a crystal is used.



- 1 - $\overline{\text{PSEN}}$
- 2 - ALE
- 3 - $\overline{\text{EA}}$
- 4 - $\overline{\text{RST}}$

* - Instruction Register

Fig 1

PIN CONFIGURATION:

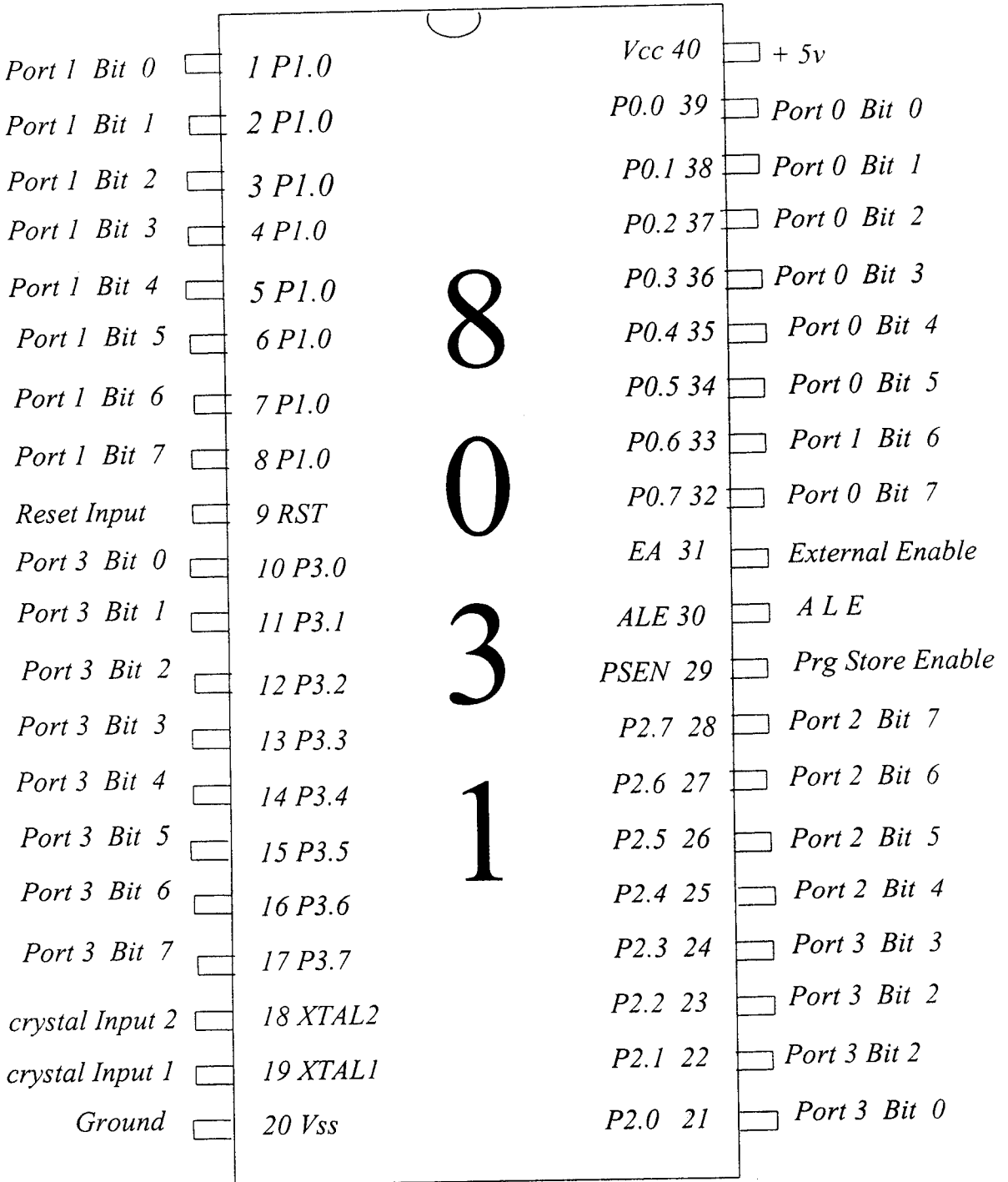


FIG.2.