PORTABLE KEYBOARD WITH MEMORY

PROJECT REPORT

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF ENGINEERING IN

INFORMATION TECHNOLOGY

OF THE BHARATHIAR UNIVERSITY, COIMBATORE

Submitted by

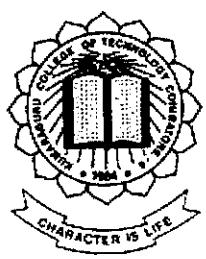
HINABEN.D.PATEL (0027S0077)

SUNAINA ABDUL SALAH (0027S0114)

Under the valuable Guidance of

Dr.S.Thangasamy Ph.D.,

Head of the Department, IT DEPARTMENT



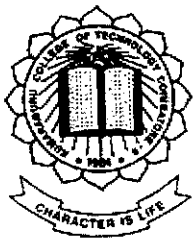
MARCH 2004

Department of Information Technology

Kumaraguru College of Technology

(Affiliated to Bharathiar University)

COIMBATORE – 641 006



KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)
COIMBATORE – 641 006, TAMIL NADU, INDIA
Approved by AICTE, New Delhi - Accredited by NBA

Department of Information Technology

CERTIFICATE

This is to certify that the project entitled

"PORTABLE KEYBOARD WITH MEMORY"

has been submitted by

Hinaben.D.Patel (0027S0077)

Sunaina Abdul Salah (0027S0114)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering Information Technology of the Bharathiar University, Coimbatore – 641 046 during the academic year 2003-2004

S. Thangasamy

Dr.S.Thangasamy Ph.D.,
(Head of the Department)

S. Thangasamy

Dr.S.ThangasamyPh.D.,
(Project Guide)

Submitted for the university examination held on 26/3/2004

S. Sreenath
26/3

Declaration

We,

HINABEN.D.PATEL 0027S0077

SUNAINA ABDUL SALAH 0027S0114

declare that the project entitled "PORTABLE KEYBOARD WITH MEMORY", done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor Of Information Technology of Bharathiar University.

Place: Coimbatore.

Date : 26/3/2004

HINABEN.D.PATEL

SUNAINA ABDUL SALAH

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr.K.K.Padmanabhan for ushering us in the path of triumph.

We are always thankful to our beloved Professor, the Head of the Department, and our project guide, Dr.S.Thangasamy whose consistent and timely support and enthusiastic involvement helped us a great deal.

We are greatly indebted project coordinator Mrs.Deviki, Senior Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we honor her for her assistance towards this cause.

We also thank our beloved class advisor Ms. P. Sudha B.E., for her invaluable assistance.

We also feel elated in expressing our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

SYNOPSIS

This project aims at developing a portable keyboard with a built in memory for users who would like to enter and store information but are not in the vicinity of a computer. It also caters to the needs of the majority who cannot afford a laptop or such expensive equipments and who only need to type in on the spot information. It proves to be more advantageous than the existing portable keyboard which uses infra red technology as even these wireless keyboards have to be within a certain range of the PC.

The keyboard is an assortment of keys consisting of alphanumeric, numeric and other essential keys such as enter, backspace, tab, space bar keys. Each key that is pressed is converted to its frequency value by the micro controller chip that is used and the data is written into the memory chip. At the same time, the user can view the information he is typing on the LCD screen where the values are mapped back to the corresponding character. To access the information on a PC, all the user has to do is take out the easily removable chip and connect it to the pc, where there is a Visual Basic application program that lets the user read the data and do as he wishes with the information.

This project uses a number of hardware components the main one being the atmel 89c51 microcontroller chip and a Visual Basic application on the server side.

CONTENTS

PAGE NO

1. INTRODUCTION

1.1 Existing System

Although the presence of notepads and laptops have increased the capacity for portable information, the system that exists now caters to larger needs as not only entering of information is done, but other complicated functions are also present. For users whose only need is to enter and store data, such devices turn out to be much more than they require and they have to pay the added costs for their minimum requirements. Thus the existing system turns out to be less cost effective for this section of users whose work would be much more efficient if they can just implement their needs. Also the other portable keyboards that are available such as the wireless keyboards use infra red technology and have to be within a certain range of the system attached.

1.2 Proposed System

To provide for the users who require only data entry and storage such that it can be accessed at a later stage in the PC, a new system was proposed in which the user can type in his information on to his custom made portable keyboard and this information is stored as it is on to an external memory device. When the user requires that he view his information in a PC at a different location, he can just take out the easily detachacble memory device and connect it to the server side which contains an application software that lets the user read the data.

The **advantages** of the system are that it specially caters to the needs of users who have to only type information and retrieve it at later stage or when the information has to be passed on, the chip can directly be passed on. The biggest advantage is the cost effectiveness, as users only have to pay a much lesser amount in the purchase of these keyboards compared to laptops, PDAs and notepads with their requirements being satisfied. Also the user can view whatever he is typing in the LCD display that is provided.

2. SYSTEM REQUIREMENTS

2. SYSTEM REQUIREMENTS AND ANALYSIS

2.1 PRODUCT DEFINITION

Our project develops a portable keyboard that is used for data entry and storage for users who want to save on the cost of buying expensive devices such as laptops and notepads as their requirements do not extend to the wide functionalities of these devices. Our user is able to simply type in his information using the assortment of keys provided so that it will be stored in an external memory. The keys provided consists of alphanumeric, numeric and other necessary keys such as spacebar, punctuation keys, back space, and enter key. The memory chip can later be easily removed out from the keyboard so that It can be connected to a PC in any location and the stored data can be easily obtained using a server side application software which is done in Visual Basic in this particular project.

2.2 PROJECT PLAN

Every implementation of a new project requires a well defined project plan, where the hardware and the software requirements and constraints are charted out. Also a detailed study of the various aspects of the project as well as the components that go about making it is necessary.

Our project makes the use of the micro controller atmel 89c51 of the 8051 micro controller family. This IC was chosen as it is easily available in the market compared to the rest in the 8051 family. Along with this IC we also use the 555 timer for timer overflow calculations in order to determine frequency. The

memory chip used is the 256 byte EEPROM 24c01 IC which has several features such as flash programming. At the server side, a 20 pin Microcontroller is used of the same corporation for data transfer. MAX 232 voltage converter is relied upon for voltage compatibility and transmission of data takes place serially through RS232 standards.

The modules that make up the project implementation are:

- ✓ Frequency calculation
- ✓ Frequency to ascii mapping
- ✓ Writing into memory chip
- ✓ LCD ascii to character mapping
- ✓ Serial transmission of data at server side.

A detailed study of all the main hardware and software components is made in the next session for the further understanding of the working of the project.

3. SOFTWARE REQUIREMENT SPECIFICATION

Software RequirementS Specification

1. Introduction:

The purpose of the project is to make a portable user specific keyboard that enables the user to enter information on the spot which is written into a memory chip so that it can be accessed later at the server side. The keyboard is specially designed for the customer with a minimum number of keys so as to increase typing speed and at the same time provide the facility to enter all kinds of data. The keyboard will consist of alphanumeric keys, numeric keys and others such as spacebar, enter, ounctuations etc. This project enables a user to carry around a handy keyboard and type in data according to his convenience and also store the data. It is available at a low cost and therefore is affordable by any kind of user such as students.

SRS 1:

Purpose of the document:

The purpose of this document is to provide the requirements for the development of the portable keyboard with built in memory. It is primarily intended for customers of the application, but will also be of interest for the hardware programmers.

SRS 2:

Scope:

This document in limited to the description of the portable keyboard that is being developed which enables a user to enter information without the aid of a PC and

to also access this information later. Our project scope can also be extended to wider applications apart from data entry such as calculations, appointment management and so on.

SRS 2.1

Product perspective:

This product was intended to fulfill the needs of users who travel around a lot, and cannot remain connected to the PC. Hence they need some cheap means of keying in the information they would need at a later stage. The design and documentation for this will make it convenient to expand and modify the keyboard purposes. It is interactive with users.

SRS 2.2

Product function:

The product is in the form of a keyboard that is presently an ordinary keyboard with the extra key functionalities taken out and only essential keys used. Later on custom made keyboards can be developed as well as foldable keyboards and such. The keyboard has the hardware fitted to it which is compact and an LCD is also provided which enables the user to view what he is typing.

SRS 2.3

User characteristics:

The user has very little knowledge about computers, hence the interface (LCD display and the VB application) is designed to be user friendly. All the user has to do is enter the data that he wants to. He does not know about the structuring of the keys or the resistance values of the characters. At the server side, he has to only fit the easily removable and placable chip on to the hardware provided and read the data from the VB application program.

The general category of users that can make use of this keyboard are students, journalists, secretaries and other people who don't need to spend the

extra cost to buy a notepad or other such devices and who need only to make a note of information.

SRS 2.4:

Constraints:

- There is a time constraint for pressing the keys and if it is not adhered to, the the wrong value could be stored in the chip

SRS 3:

Development and operating environment:

SRS 3.1

Hardware:

* Processor : Pentium III
* RAM : 128 MB
* RS232 Serial port connector
* 8 bit Microcontroller ATMEL 89c51
    - 8 bit CPU optimized for control applications
      Extensive Boolean processing (Single bit logic) capabilities
    - On- chip flash program memory
    - On- chip Data RAM
    - Bi-directional and individually addressable i/o lines
    - Multiple 16 bit timer counters
    - Full Duplex UART
    - Multiple Source/ Vector/ Priority Interrupt structure.
    - On- chip oscillator and clock circuitry.

- On- chip EEPROM
- Watch Dog Timer
- SPI Bus Interface.
- Operating Voltage Range : 2.0V to 5.5V
- Current : 25 mA

* MAX232 IC
* 555 Timer circuitry
* Oriole LCD
* LEDs
* Various capacitors and resistors

SRS 3.2

Software:

* Win 3.1 and higher versions
* Visual Basic 6.0
* Syntronix assembler software

SRS 4

Definitions and acronyms:

* RS232 → Serial port connector
* LCD → Liquid Crystal Display
* LED → Light Emitting Diode
* MAX232 → IC used to provide compatibility between serial port and the Micro controller
* EEPROM → Electrically Erasable Programmable Read Only Memory

4. DETAILED STUDY

4.1 HARDWARE REQUIREMENTS

The 8051 MicroController

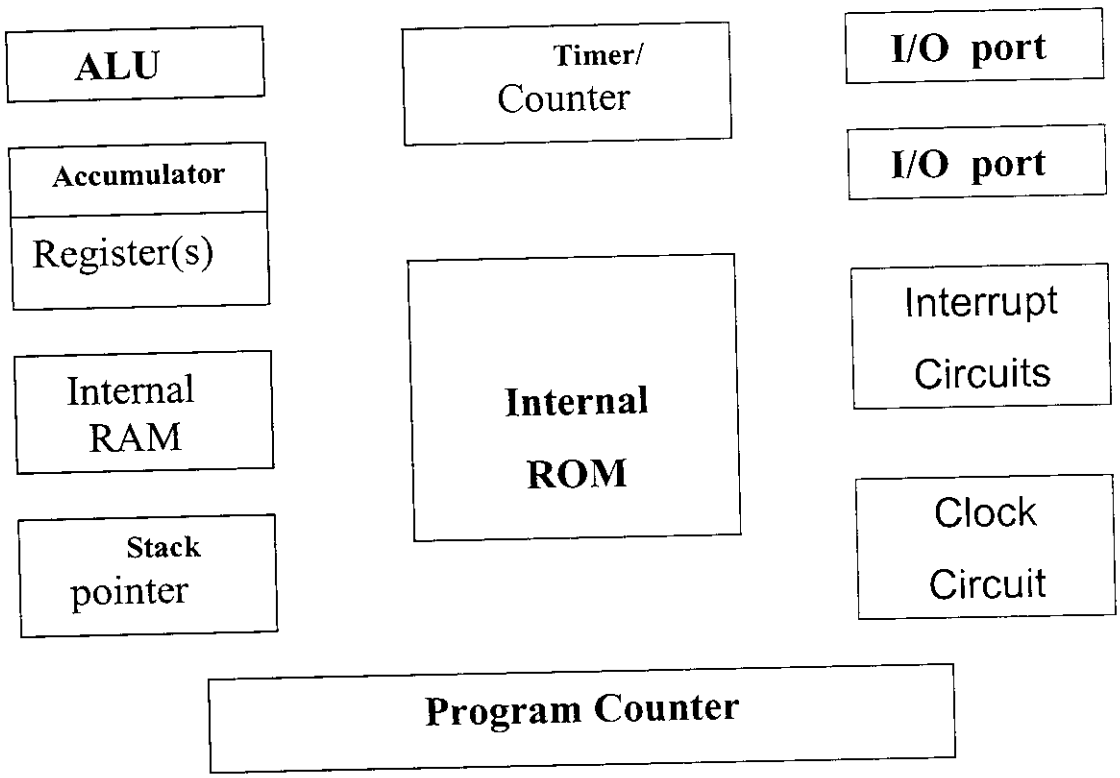


Figure shows the block diagram of a typical micro controller (MC), which is a true computer on a chip. The design incorporates all of the features found in a microprocessor (MP) CPU: ALU, PC, SP and registers. It also has added the other features needed to make a complete computer: ROM, RAM, parallel I/O, serial I/O, counters, and a clock circuit.

Like the microprocessor, a MC is a general-purpose device, but one that is meant to read data, perform limited calculations on that data, and control its environment based on that calculations. The prime use of a MC is to control the operation of a machine using a fixed program that is stored in ROM and that does not change during the lifetime of the system.

The design approach of the MC mirrors that of the MP (micro processor): make a single design that can be used in as many applications as possible in order to sell as many as possible. The MP design accomplishes this goal by having a very flexible and extensive repertoire of multi byte instructions. These instructions work in a hardware configuration that enables large amounts of memory and I/O to be connected to address and data bus pins on the integrated circuit package. Much of the activity in the MP has to do with moving code data to and from external memory to the CPU. The architecture features working registers that can be programmed to take part in the memory access process, and the instruction set aimed at expediting this activity in order to increase throughput. The pins that connect MP to external memory are unique, each having a single function. Data is handled in bytes, or larger, sizes.

The MC design uses a much more limited set of single and double byte instructions that are user to move code and data from internal memory to the 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 MC is concerned with getting data from and to its own pins; the architecture and instruction set are optimized to handle data in bit and byte size.

Micro Controller Hardware:

The 8051 MC generic part number actually includes a whole family of MCs that have numbers ranging from 8031 to 8751 and is available in N-channel Metal Oxide Silicon (NMOS) and Complementary Metal Oxide Silicon (CMOS) construction in a variety of package types. An enhanced version of the 8051, the 8052, also exists with its own family of variations and even includes one member that can be programmed in BASIC. Some unique features of a micro controller are,

- Internal ROM and RAM
- I/O ports with programmable pins
- Timers and Counters
- Serial data communication
- Program counter, ALU, working registers, and Clock circuits.

The 8051 architecture consists of these specific features:

- 8-bit CPU with registers A (the accumulator) and B
- 16-bit program counter and data pointer
- 8-bit program status word
- 8-bit stack pointer
- Internal ROM or EPROM of 0 to 4k

Internal RAM of 128 bytes:

- 4 register banks, each containing 8 registers
- 16 bytes, which may be addressed at the bit level

- 80 bytes of general purpose data memory

- 32 I/O pins arranged as 4 8-bit ports: p0-p3
- Two 16-bit timer/counters: t0 and t1
- Full duplex serial data receiver/transmitter: SBUF
- Control registers: TCON, TMOD, SCON, PCON, IP and IE
- Two external and three internal interrupt sources
- Oscillator and clock circuits

All the 8-bit and 16-bit registers and 8-bit memory locations can be made to operate using the software instructions that are incorporated as part of the design. The program instructions have to do with the control of registers and digital data paths that are physically contained inside the 8051, and the memory locations that are contained outside the 8051.

The number of special purpose registers complicates the model that must be present to make a microcomputer a MC. Most of the registers have a specific function; those do occupy the individual block have a symbolic name, such as A or TH0 or PC. Others, which are generally indistinguishable from each other, are grouped in a larger block, such as internal ROM or RAM memory.

Each register, with the exception of the program counter, has an internal 1-byte address assigned to it. Some registers are both byte and bit addressable. That is, the entire byte of data at such register addresses may be read or altered, or individual bits may be read or altered. Software instructions are generally able to specify a register by its address, its symbolic name, or both.

Input/Output Pins, Ports, and Circuits:

One major feature of a MC is the versatility built into the I/O circuits that connect 8051 to the outside world. The MP designs must add additional chips to interface with external circuitry; this ability is built into the MC.

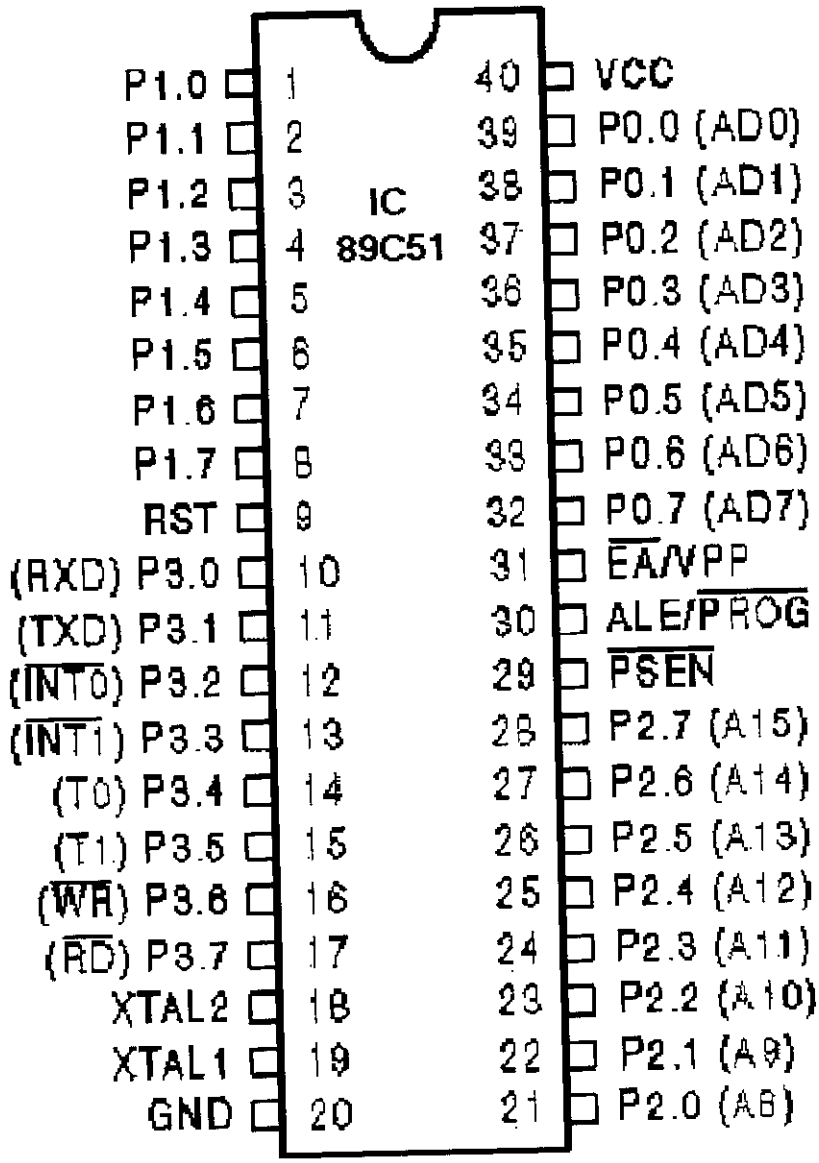
There are number of pins available in 8051 circuit design. The DIP has 40 pins, and the success of the design in the marketplace was determined by the flexibility built into the use of these pins.

For this reason, 24 of the pins may each be used for one of two entirely different functions, yielding a total pin configuration of 64. The function of a pin, first, depends on, what is physically connected to it and, on what software commands are used to "program" the pin.

The 8051 may be expanded to include additional memory, parallel ports, and serial data communication by using the alternate pin assignments. There are two data paths in the pin port circuitry that read the latch or pin data using two entirely separate buffers. The upper buffer is enabled when latch data is read, and the lower buffer, when the pin state is read. The status of each latch may be read from a latch buffer, while an input buffer is connected directly to each pin so that the pin status may be read independently of the latch state.

Programmable port pins have completely different alternate functions. The ports are not capable of driving loads that require currents in the tens of milliamperes.

PIN DESCIPTION OF ATMEL 89C51



The atmel 89c51 microcontroller chip belongs to the 8051 family and it is the IC that is used in this project. This particular microcontroller was chosen as it is easily available in the market and provides for all the features of the 8051 microcontroller and more.

89c51 have 40 pins that are dedicated for various functions. Some companies provide 20-pin version of the 89c51 with a reduced number of I/O ports for less demanding applications.

A total of 32 pins are set aside for the four ports P0, P1, P2, P3, where each ports take 8 pins. The rest of the pins are designated as Vcc, GND, XTAL1, XTAL2, RST, EA, PSEN.

Vcc

Pin 40 provides supply voltage to the chip. The voltage source is +5V.

GND

Pin 20 is ground.

XTAL1 and XTAL2

The 89c51 has an on-chip oscillator but requires an external clock to run it. A quartz crystal oscillator is connected to inputs XTAL1 and XTAL2. This oscillator needs two capacitors.

RST

Pin 9 is the RESET pin. It is an input and is active high. Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities. This is often referred to as a *power-on reset.*

$\overline{\text{EA}}$

This pin which stands for "external access", is pin number 31. it is an input pin and must be connected to either Vcc or GND.

ALE

It is an output pin and is active high. It is used for demultiplexing the address and data.

PSEN

This is an output pin. It stands for "program store enable".

I/O port pins and their functions

There are four ports P0, P1, P2, P3 each use 8 pins, making them 8 ports.

Port 0:

Port 0 occupies a total of 8 pins (pins 32 through 39). It can be used for input or output. To use the pins of port 0 as both input and output ports, each pin must be connected externally to a pull-up resistor. With these resistors connected upon reset, port 0 is configured as an output port.

Port 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It can be used as input or output. This port does not need any pull-up resistors since it already as pull-up resistors internally. Upon reset, port 1 is configured as an output port.

Port 2:

Port 2 occupies a total of 8 pins (pins 21 through 28). It can be used as input or output. This port does not need any pull-up resistors since it already as pull-up resistors internally. Upon reset, port 1 is configured as an output port.

Port 3:

Port 3 occupies a total of 8 pins (pins 10 through 17). It can be used as input or output. This port does not need any pull-up resistors since it already as pull-up resistors internally. Port 3 has the additional function of providing some extremely important signals such as interrupts.

Addressing modes:

There are five addressing modes in 8051.

- Immediate
- Register
- Direct
- Register indirect
- Indexed

Immediate addressing mode:

In this addressing mode, the source operand is a constant. When the instruction is assembled, the operand comes immediately after the opcode. Immediate data must be preceded by the pound sign, "#". They can be used to load information into any of the registers.

Register addressing mode:

This addressing mode involves the use of registers to hold the data to be manipulated. The source and destination registers must match in size. The operands are either inside one o the registers or tagged along with the instruction itself.

Direct addressing mode:

This type of addressing mode is used to access entire 128 bytes of RAM location. This is due to the fact that register bank locations are accessed by the register names of R0-R7 but there is no such name for RAM location.

Register indirect addressing mode:

In this type of addressing mode, a register is used as a pointer to the data. If the data is inside the CPU, only registers R0 and R1 are used for this purpose. When these registers are used as pointers, that is, when they hold the addresses of RAM locations, they must be preceded by the "@" sign.

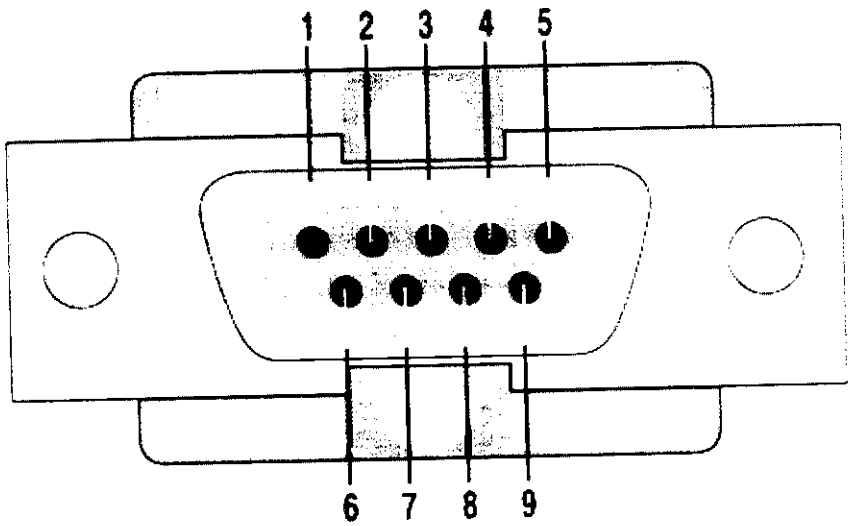
Indexed addressing mode:

This type of addressing mode is widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051.

RS-232

RS-232 stands for Recommended Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. It sends and receives data using twisted pair cable. The full RS-232C standard specifies a 25-pin 'D' connector of which 22 pins are used. Most of these pins are not needed for normal PC communications and indeed; most new PCs are equipped with male 'D' type connectors having only nine pins.

PIN SCHEMATIC



Pin Number	Direction of Signal
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request to Send (RTS) Outgoing flow control signal
8	Clear to Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

RS-232 uses an unbalanced signal communication method. That is, there is one signal wire for each circuit with a common return for all signals. This method is somewhat susceptible to electrical noise.

To communicate, the device sends a series of binary signals to the receiver. These binary pulses make up predefined words that indicate either commands or status conditions.

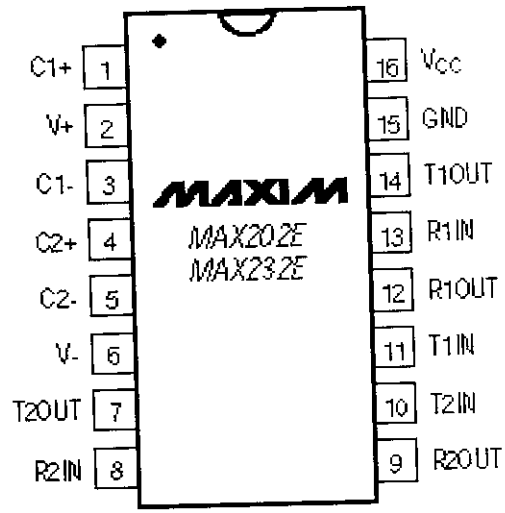
The RS-232 communication standard supports only one transmitter driver and a receiver. Distance is limited to 50 feet.

RS-232 MAX CONVERTER:

RS-232 was introduced in 1960, and is currently the most widely used communication protocol. It is simple, inexpensive to implement, and though relatively slow, it is more than adequate for simple serial communication devices.

MAX232 IC belongs to the MAX220 – MAX249 series, which contains four sections: dual charge–pump DC-DC voltage converters, RS-232 drivers, RS-232 receivers and receiver & transmitters enable control inputs. The MAX232 is a dual RS-232 driver/receiver pair that generates RS-232 voltage levels from a single +5V power supply. Additional +12V supplies are not needed since the MAX232 uses on-board charge pumps to convert the +5V supply to +10V. Some of its technical features are:

- Single ended data transmission system
- One driver per line and one receiver per line
- Maximum cable length is 50 feet
- Maximum data rate is 20 kbps
- Driver output maximum voltage is -25 to +25v
- Driver output signal level is -5 to +5v(loaded)
- Driver output signal level is -15 to +15v(unloaded)
- Driver load impedance is 3 KOhm to 7 KOhm
- Maximum Driver output current is Vmax/300 ohm(power off)
- Slew rate is 30 V/micro sec (max)
- Receiver input voltage range is -15v to +15v
- Receiver input sensitivity is -3 to +3v
- Receiver input resistance is 3 KOhm to 7 Kohm

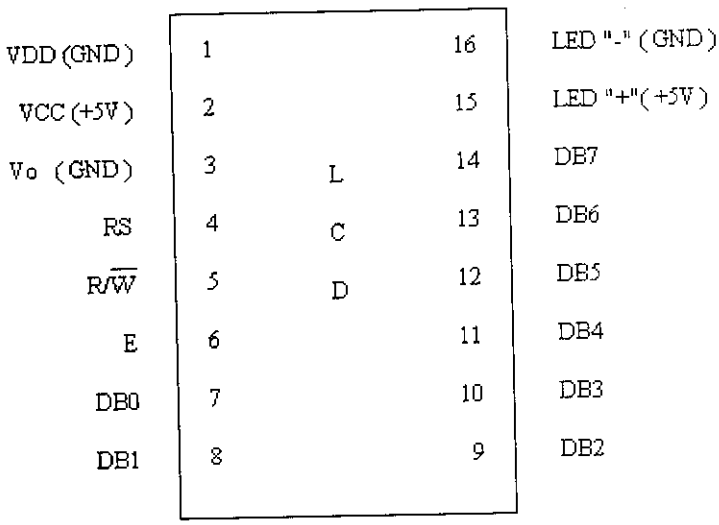


Pin description

VCC	-	+5 Volt Supply
GND	-	Ground
V+	-	Positive Supply Output
V-	-	Negative Supply Output
T1IN, T2IN	-	RS-232 Driver Inputs
T1OUT, T2OUT	-	RS-232 Driver Outputs
R1IN, R2IN	-	Receiver Inputs
R1OUT, R2OUT	-	Receiver Outputs
C1+, C1-	-	Capacitor 1 Connections
C2+, C2-	-	Capacitor 2 Connections

LCD DISPLAY

Pin configuration



The LCD display used is 16x1 Oriole LCD with its data lines connected to the microcontroller port 1 pins and the three control lines connected to port 3 pins.

Description Of Terminals

Pin No.	Pin Name	Input/ Output	External Connection	Function
1	VSS	—	Power Supply	VSS:GND
2	VDD	—		VDD: +5V
3	VO	—		V_{LCD} adjustment
4	RS	INPUT	MPU	Register select signal "0":Instruction register (when writing) Busy flag & address counter (When reading) "1":Data register (when writing & reading)
5	R/W	Input	MPU	Read/write select signal "0" for writing , "1" for reading
6	E	Input	MPU	Operation (data read/write) enable signal
7 / 10	DB0-DB3	Input	MPU	Low-order lines of data bus with 3-state, bi-directional function for use in data transaction with the MPU. These lines are not used when interfacing with a 4-bit microprocessor.
11 / 14	DB4-DB7	Input	MPU	High-order lines of data bus with 3-state, bi-directional function for use in data transactions with the MPU. DB7 may also be used to check the busy flag.
15 / 16	LED „+„ LED „-„	Input	LED BACKLIGHT POWER SUPPLY	LED„+„VOLTAGE TYPE:4.2V MAX : 4.5V LED„-„: GND

3.2 SOFTWARE REQUIREMENTS

PROGRAMMING THE 8051:

Lines of Code:

Assembly language programs are written as a sequence of text lines. A text line is nothing more than a line of all alphanumeric characters that are stored in a disk file. Each line ends with a carriage return character (or carriage return line feed) is stored by the text editor on the disk. A file is created using a word processor or text editor. The only condition placed on the file is that it must be stored on the disk in pure ASCII (American Standard Code for Information Interchange) format. Using ASCII format for the file ensures that unusual characters, used by most word processors for special functions (such as underlying or tabs), do not appear in the final file.

Each line of text is an instruction to the CPU, a directive to the assembler, or a combination of the two. Many names have been used for a single text line in a source file, such as a statement, or a line of code, or an instruction.

8051 Instruction Syntax:

Almost all computer instructions involve taking one piece of data from a location somewhere inside the computer and "operating" on that data together with another piece of data located somewhere else inside the computer. Data

originates at the source location and ends up at the destination location. Instructions commonly start with an action mnemonic that reminds us of what the instruction does. The instruction mnemonic 6 is the first word of the instruction and indicates in what manner, or how, the data are to be combined or otherwise manipulated by the CPU.

Each code line may also contain comments and a name for the instruction address in code memory, called the label of the code address. A label is a name, or symbol, given to the address number where the first byte of the label instruction is stored in code memory. Instructions are assembled into code memory with each byte of code located at a unique address number. Labels let us convert address numbers in code memory into names of our choosing and let the assembler worry about exactly which address number has each name. If the address number of a line of code should happen to change as a result of adding a new instruction, for instance the name remains the same, only the address number for that name will change. The assembler can keep track of address numbers for each name, and we can keep track of the names.

The overall syntax of a line of 8051 program code is shown :

An assembly language instruction line

```
Label:      instruction          ;comment(s)
```

Labels:

A label can be any combination of up to 8 letters (A-Z), numbers(0-9), and special characters like question mark(?), period(.), at(@), underline(_), and dollar sign($). The first character of label must be an alphabetic character. Examples of code address labels are the following:

square:

tabletwo:

curses:

setmode:

fred.2:

transmit:

receive:

ad1234h:

Try to keep labels name short and related to the program.

Labels should be placed in the first text column, on the far left of the source file page, so that we can see, quickly, that it is label and not an instruction. The assembler will associate the label with the first instruction line after the label. Labels always end with a colon(:) to indicate they are labels and not instructions. They cannot be the name of any register or instruction used by the 8051.

Instructions:

An 8051 instruction is any of the coded set that has been defined by the manufacturer of the 8051. Every instruction can be converted into a unique machine-language binary code that can be acted on by the 8051 internal circuitry.

Each instruction is made up to three distinct parts,

An 8051 instruction

Instruction Part	mov	destination,source

Part 1, (mov) it is the operation to be performed by the CPU. It will move data from one location (source) to another location (destination).

Part 2, (destination) is called an operand or data address, because it specifies the destination for the data that is being copied from the source.

Part 3, (source) is also an operand and contains the address of the source location in the computer that is providing data to be copied to the destination.

The names of the destination and source address are separated by a comma(,) so that the assembler will know when one operand ends and other begin.

Comments:

The programmer includes comments as simple, terse explanations of exactly what the instruction is doing to make the program function. Omitting comments leads to sloppy and erroneous programs. They begin with a semicolon(;) to indicate they are not one of the operands. Anything can be typed after the semicolon, even reserved words. Examples of comments include the following:

```
;this is a comment
;and so is this
;mov contents of b to a
;add value to ass
```

One rather handy way to use the semicolon when debugging a program is to place a semicolon in front of a line of a code that you may think needs to be deleted. Placing a semicolon in front of the line of suspect code will ensure it is not part of the assembled program.

ABOUT VISUAL BASIC:

Visual Basic is the newest addition to the family of Visual Basic products. It allows you to develop Windows application quickly. Visual Basic contains many integrated tools to make the application process simpler. This collection of tools makes up the *Integrated Development Environment* (IDE). It provides a graphical environment to design the forms and controls that becomes building blocks of applications. These include other tools like projects, class objects, templates, custom controls, add-ins, and database managers. Visual Basic continues to sport the Explorer-style development environment. Visual Basic allows you to work on multiple projects simultaneously.

When you start Visual Basic for the first time, the Project Wizard will open, and New Project dialog box is seen. From this window several types of projects can be selected that gives head start for developing projects.

The Visual Basic IDE is made up of number of components:

- Menu bar
- Toolbar
- Project Explorer
- Properties window
- Form Layout window
- Toolbox
- Form Designer
- Object Browser

The menu bar is the line of text that lies across the top of the Visual Basic window. Immediately below the menu bar is the Visual Basic toolbar. The Project Explorer allows you to expand and collapse the subfolders. The Properties window exposes the various characteristics of selected objects. Every form in an application is an *object.* All the characteristics of an object are called its *properties.* The Form Layout window purpose is to give a thumbnail view of the current form. The Toolbox contains tools or objects that are referred to as *controls* to build application interface. Most of them are *built-in* or *standard* controls. The Form Designer is the workspace where actual design of the visual layout of the form and the controls lies. The Object Browser allows you to browse through the various properties, events, and methods that are made available or exposed.

Form is a window that you can add different elements to in order to create a complete application. It contains many elements like Control menu, Caption, Titlebar, Minimize button, Maximize/Restore button, Close button and Border. A Form's *container* is also known as multiple document interface (MDI) form. It allows users to open more than one file at a time. Properties can be used to manipulate the identity of an object, its appearance or even its behavior. To open the code, double click the command for the command button's Click event.

Windows is an *event-driven* operating system. Events are triggered by messages. Whenever a button is clicked, Windows will generate a message.

This message then gets sent to the message queue. The message from there is send to the appropriate control, a form. When the control receives this message, it then generates an appropriate event. A form is actually activated after it is initialized. The form then receives the focus after it has been activated. The Deactivate event occurs when the form ceases to be active. The Error event is triggered every time an error occurs. The error may occur due to calling a wrong arguments or attempting to set a nonexistence property.

Controls add functionality to programs. A label control displays read only text. A collection is a simple structure that works like an array storing related items. Collection lets you access its items via a key. Item is a method of the collection that returns a collection item based its key or index. It provides three methods and one property. Add, item, remove methods and count property. The easiest way to debug a Visual Basic program is by placing statements in your program to display variable information and to interrupt execution. The MsgBox statement has this ability.

Three types of errors are encountered while developing an application.

- Compile-time errors
- Run-time errors
- Logic errors

Compile-time errors result from syntax errors or an invalid use of a function or property. *Run-time error* results when a syntactically correct line of code can't execute because of the current environmental circumstances. *Logic*

errors result from a mistake or oversight on the programmer's part and can be very difficult to track down. Error trap is set for this behavior using the On Error statement. This statement has three forms:

- On Error Go To Label
- On Error Resume Next
- On Error Go To 0

Using Err, Visual Basic provides information to find in which error has occurred. The VBA debugging environment has several tools to aid you in hunting down bugs and logic errors. Breakpoints set locations in code at which VBA will temporarily halt its execution of the code.

There are many string manipulation and variable functions for type conversions. Few are

- VarType(variable)
- Asc(string)
- StrComp(string1,string2[,start][,compare])
- Str(number)

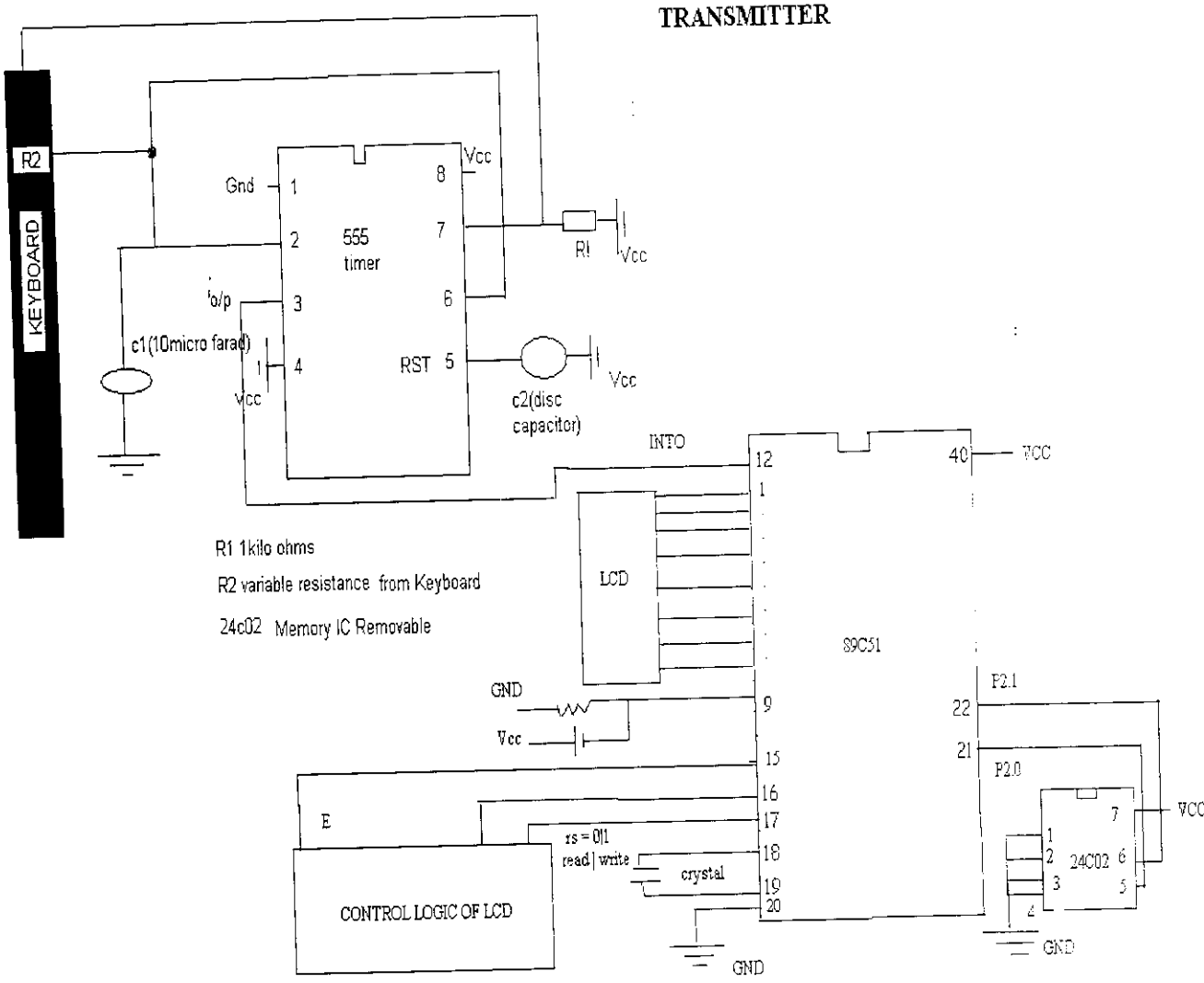
The CreateObject() function starts an OLE compliant application and returns a reference to the application or one of the objects it exposes. It starts a new instance of the specified application, even if one is already running. The GetObject() function opens an existing document with its application.

5. DESIGN DOCUMENT

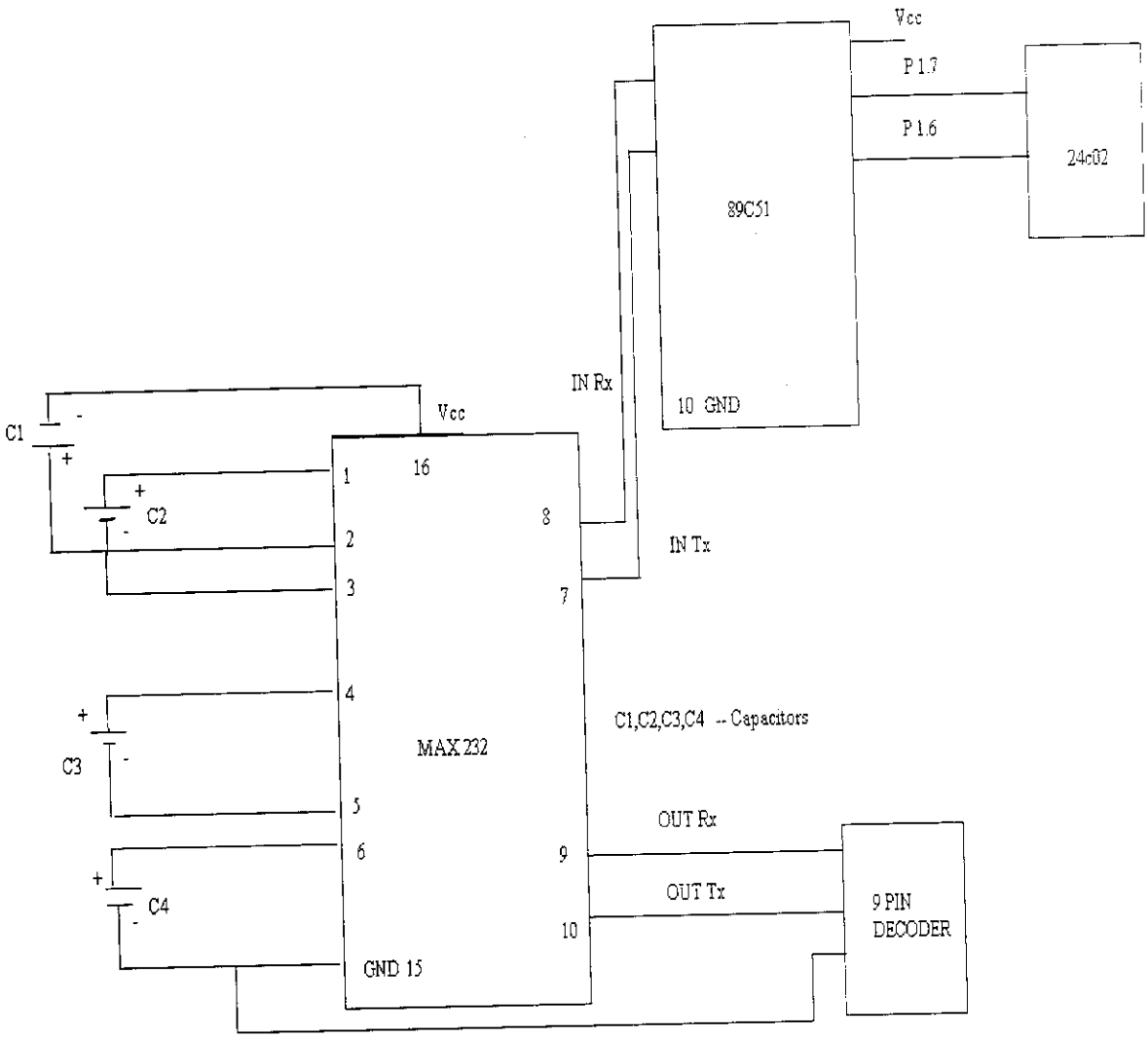
5.1 CIRCUIT DIAGRAMS

TRANSMITTER SIDE

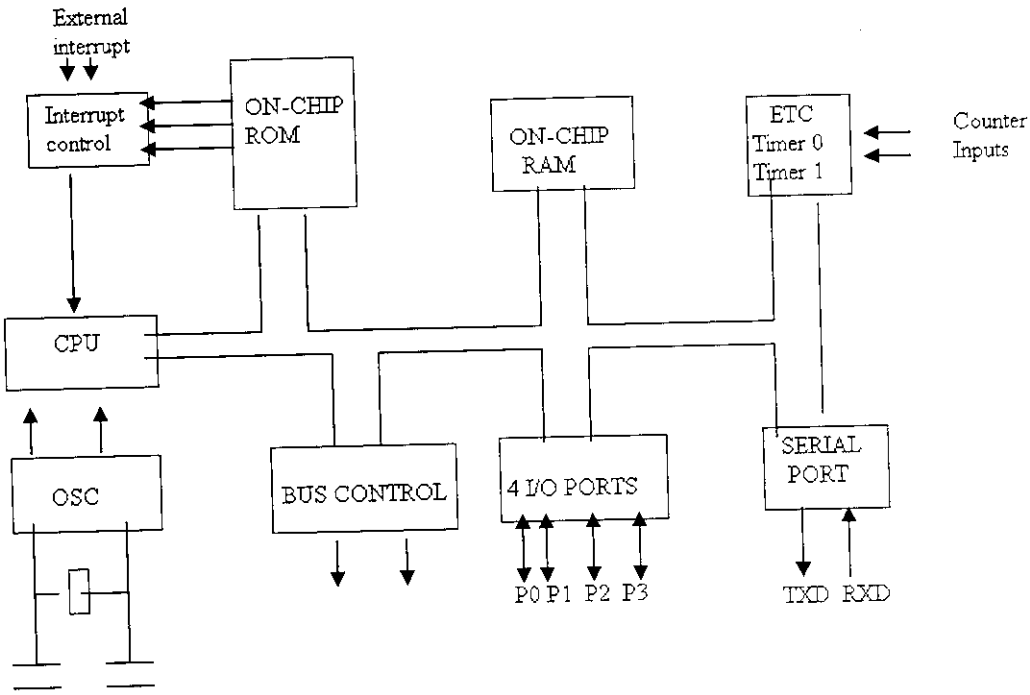
TRANSMITTER



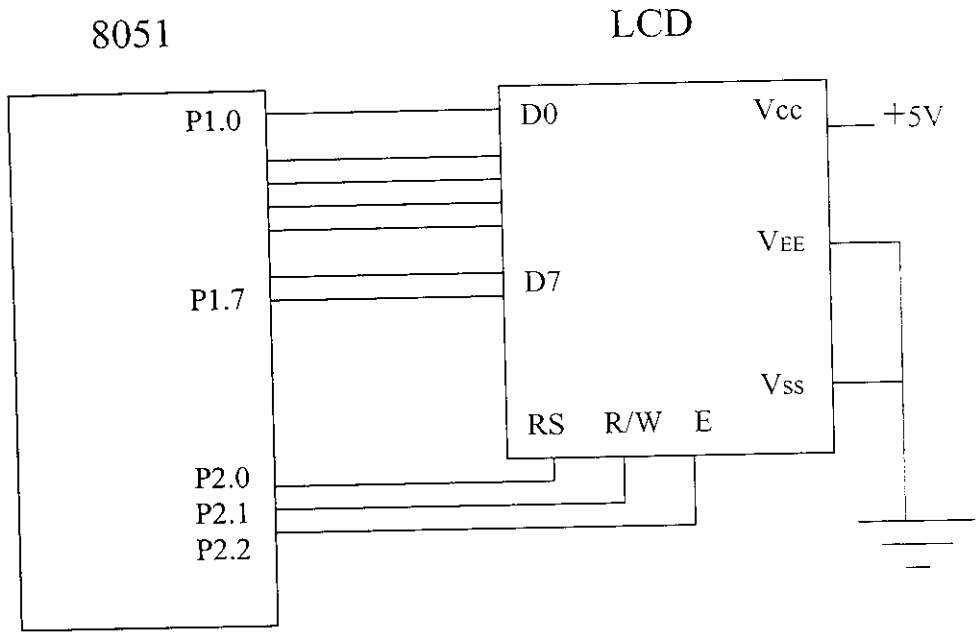
RECEIVER


Vcc
P1.7
P1.6
24c02
89C51
10 GND
IN Rx
IN Tx
C1
C2
C3
C4
Vcc
1
16
2
8
3
7
4
MAX 232
5
6
9
10
GND 15
C1,C2,C3,C4 -- Capacitors
OUT Rx
OUT Tx
9 PIN
DECODER

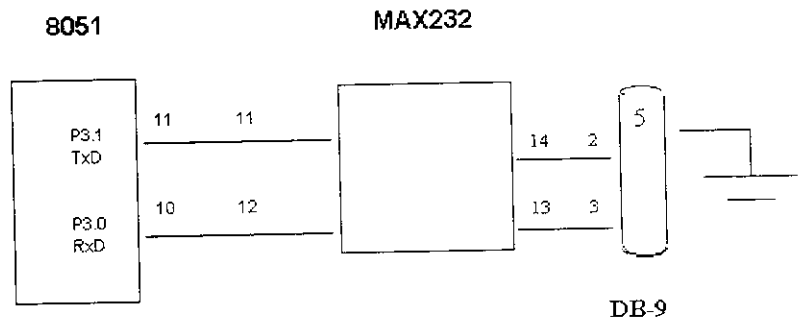
INSIDE THE 8051 MICROCONTROLLER



LCD CONNECTION



MAX232 Connection to 8051



6. IMPLEMENTATION

6. IMPLEMENTATION

The Implementation part of a project comes after proper planning and designing. This part of the document describes the modules that have been executed in the project,

MODULE DESCRIPTION:

Frequency calculation

* The very first module deals with the calculation of the frequency corresponding to each key that is typed. Every key in the key board adds up to a particular resistance value that is given as input to the timer. The timer generates an interrupt signal in the microcontroller which calls the interrupt service routine. Here for the number of timer overflows that is set, the frequency counter is incremented for each transition of the pulse from high to low. Thus each character corresponds to a particular counter value which is the frequency of the character.

Frequency to character mapping

* The frequency that is calculated is mapped to the ascii value of the character for reading purposes.
* Each character corresponds to a range of frequencies. Therefore if this range occurs, this module maps it to the particular characters ascii value. For ex, if key a is pressed, and the frequency is say 35, in this module it is mapped to ascii code 97.

Writing into the EEPROM

* As soon as the frequency is calculated and mapped to its ascii value, it is written into the EEPROM 24c01 chip.

LCD display

* To enable the user to view what he is typing the keyboard has been interfaced with an LCD. This module sends the ascii data directly to the data pins of the LCD so that automatic ascii to character conversion is done and the data is displayed.

Serial transmission of data

* At the server side, the chip is fitted into a setup with a 20 pin 8051 microcontroller and the data is transmitted serially, bit by bit using RS232 data transfer. Using a VB application, the data in the chip is read after ascii to character conversion is done.

7. TESTING

6. SYSTEM TESTING

Testing is an activity to verify that a correct system is being built and is performed with the intent of finding faults in the system. Testing is an activity, however not restricted to being performed after the development phase is complete. But this is to be carried out in parallel with all stages of system development, starting with requirement specification. Testing results once gathered and evaluated, provide a qualitative indication of software quality and reliability and serve as a basis for design modification if required.

Some of the hardware problems encountered during the project and its corresponding corrections were:

- For connecting LCD there are 8 data pins associated with it, so while during interfacing the 89c51 with LCD one of the data pins was interconnected. These types of errors are difficult to find since some ascii characters were displayed correctly while some others were displayed incorrectly.

- While connecting 24c02 memory chip problems were encountered such as whether we read or write output was not being produced. This particular problem was because of the three address pins which have to be grounded to work properly, otherwise the proper address has to be given through the program.

- The power to the keyboard has to be constant, since initially we used an adapter to map all frequencies to ascii values. The mapping turned out to be inconsistent since the adapter delivered different voltages at different times(good power ratings at night and poor rating at day time).

- Problems were encountered due to misconnections too as a result of improper soldering etc.

Some of the software problems encountered during the project were:

- In the VB application, first the serial connection is checked as to whether it is successful or not. Once the connection is established it cannot be given again in order to avoid data discrepancies.

- While integrating the frequency detection module with memory module, the variable r0 was shared between two modules that was not noticed initially which resulted in the malfunction of the memory module.

- Mapping the correct values from frequency to ascii proved to be tedious as a slight change in the frequency could cause error in the ascii value. Also one cannot find the frequency by adding extra circuitry since that affects the power which invariably affects the frequencies again.

8. FUTURE ENHANCEMENTS

FUTURE ENHANCEMENTS

- ✓ Inclusion of a calculator functionality where all operations can be done and displayed in the LCD

- ✓ A provision for keeping and managing appointment dates can be included with the complete availability of reminders.

- ✓ The keyboard can be modifies to be used by steganographers, with shorthand keys implemented.

- ✓ The keyboard can be custom made according to the requirement of the user and this technology can be used in all kinds of keyboards such as foldable keyboards and ones of various sizes.

9. CONCLUSION

CONCLUSION

The project to develop a portable remote keyboard has been successfully implemented. The keyboard prototype implements all the essential keys for typing any kind of text information. Successive tests and corrections have eliminated the junk value discrepancies that occur in the LCD part as well as the overlapping of resistance combinations. Using this technology, users can now record information as if he was working on his PC with the advantage that he does not have to be connected and can be at any location. He can not only do this but also access this information on a PC so that further modification can be done to it such as editing the data, transferring it and incorporating the data into other systems or files. A wide range of applications can be foreseen with this keyboard prototype. Custom made keyboards that can be easily affordable will go a long way in serving basic user needs.

10. BIBLIOGRAFIA

BIBLIOGRAPHY

 Douglas V Hall, "microprocessors and Interfacing, programming and hardware"

 Mike Predko, "Programming and customizing the 8051 microcontroller", Tata Mcgraw edition.

 Muhammed Ali Mazidi & Janice Gillispie, "The 8051 microcontroller and embedded systems."

 Wayne Freeze, "Expert guide to Visual Basic 6"

 www.google.com

 www.atmel.com

 www.8051.com

11. APPENDIX

11.1 SAMPLE CODE

ASSEMBLY CODE

READ & TRANSMIT

```
        org 0000h
        ljmp main

                org 001bh
          reti

                org 0100h
main:

inverse_add equ 40h
inc_add     equ 41h
temp        equ 42h
temp_bit    equ 20h
            mov inc_add,#0
            lcall init

            ;mov r7,#0
loop:

            ;mov p0,#9
            cpl     p3.5
            mov     p0,inc_add
            ;lcall delay
            ;lcall eepromwrite
            ;lcall delay
            lcall eepromwrite1  ;without dummy write cycle there was an error
            lcall delay
            lcall eepromread


        ;lcall chumma

                ;lcall inverseAddress
        ;mov p2,inverse_add ; address in the eeprom (reverse order i.e 00000001b)
        ;mov p2,a



            mov     p2,a
            mov temp,a
            lcall delay

            lcall transmit
            lcall delay

            ;lcall delay1
```

```
                inc inc_add
                ;inc r7


;               lcall inverseAddress
;               mov     p2,inverse_add

;               lcall delay1
;               mov p2,#9
;               lcall delay

;               inc inc_add
;               ljmp loop


                ;mov r6,#7
                ;lcall eepromwrite

                ;lcall delay

                ;mov p2,#0
                ;mov a,#0

                ;lcall eepromread
                ;lcall delay
                ;mov p2,A

                ;lcall delay1
                ljmp loop


                org 0200h
chumma:

                lcall inverseAddress
    mov p2,inverse_add ; address in the eeprom (reverse order i.e 00000001b)
                ret

                org 0300h
inverseAddress:

                ;logic is to swap 0th and 7th bit,1 and 6th bit,and so on

                clr a
                clr C

                mov a,inc_add

                mov C,acc.0
                jb acc.7,set1
                clr acc.0
                sjmp cont1

 set1:
                 setb acc.0
 cont1:
```

```
                mov acc.7,C

                ;mov p2,#0
                ;lcall delay
                ;mov p2,a
                ;lcall delay1


    clr C
                mov C,acc.1
                jb acc.6,set2
                clr acc.1
                sjmp cont2

set2:
                setb acc.1
cont2:
                mov acc.6,C

                ;mov p2,#0
                ;lcall delay
                ;mov p2,a
                ;lcall delay1

                clr C
                mov C,acc.2
                jb acc.5,set3
                clr acc.2
                sjmp cont3

 set3:
                setb acc.2
 cont3:
                mov acc.5,C

                ;mov p2,#0
                ;lcall delay
                ;mov p2,a
                ;lcall delay1

                clr C
                mov C,acc.3
                jb acc.4,set4
                clr acc.3
                sjmp cont4

  set4:
                setb acc.3
  cont4:
                mov acc.4,C

                ;mov p2,#0
                ;lcall delay

                ;mov p2,a
                ;lcall delay1
```

```
                mov inverse_add,a

mov inverse_add,a ;now moving the reversed content to variable
                ret


                org 0400h

delay:
                mov r2,#0ffh
loop2:  mov r3,#0ffh
loop1:  djnz r3,loop1
                djnz r2,loop2
                ret

delay1:
                mov r1,#15
loop6:  mov r2,#0ffh
loop7:  mov r3,#0ffh
loop8:  djnz r3,loop8
                djnz r2,loop7
                djnz r1,loop6
                ret




 eepromwrite:

                LCALL I2CStart
        mov A,#00000101B ; control address(constant)
        LCALL I2CSend
           lcall inverseAddress
        mov A,inverse_add ; address in the eeprom (reverse order i.e 00000001b)
        ;mov p2,inverse_add
        LCALL I2CSend

        lcall delay
        mov a,inc_add
        ;mov p0,inc_add
        ;mov A,#10000011B ; data to be written (reverse order)
        LCALL I2CSend
        lcall I2CStop

                lcall delay

                ret

  eepromwrite1:

                LCALL I2CStart
        mov A,#00000101B ; control address(constant)
        LCALL I2CSend
           lcall inverseAddress
```

```
      mov A,inverse_add ; address in the eeprom (reverse order i.e 00000001b)

      ;mov A,#10000000B ; address in the eeprom (reverse order i.e 00000001b)
      ;LCALL I2CSend

      ;mov a,#1

      ;mov A,#10000011B ; data to be written (reverse order)
      LCALL I2CSend
      lcall I2CStop

                  lcall delay

                  ret


eepromread:

      ;for reading the value in the eeprom
       LCALL I2CStart
       mov A,#00000101B ; control address(constant)
       LCALL I2CSend
       ;lcall delay
          lcall inverseAddress
          ;mov a,#01000000b

       mov A,inverse_add ; address in the eeprom (reverse order)
       ;mov p2,inverse_add
       ;lcall delay
       LCALL I2CSend

       lcall I2CStop
        ;lcall
;noack:
 ;       jb p1.3,noack  ;if SDA line is low indicaing acknowledgement



       LCALL I2CStart
       ;LCALL I2CStart
       mov A,#10000101B ; control address
       LCALL I2CSend

 ;noack1:
 ;       jb p1.3,noack1  ;if SDA line is low indicaing acknowledgement

       ;mov A,#10000000B ; address in the eeprom (reverse order i.e 00000001b)
       ;LCALL I2CSend
       ;lcall delay
       ;clr a
       LCALL I2CGet    ; the value in th address will be retrived in accumulator (reverse order)
       lcall I2CStop

                  ret
```

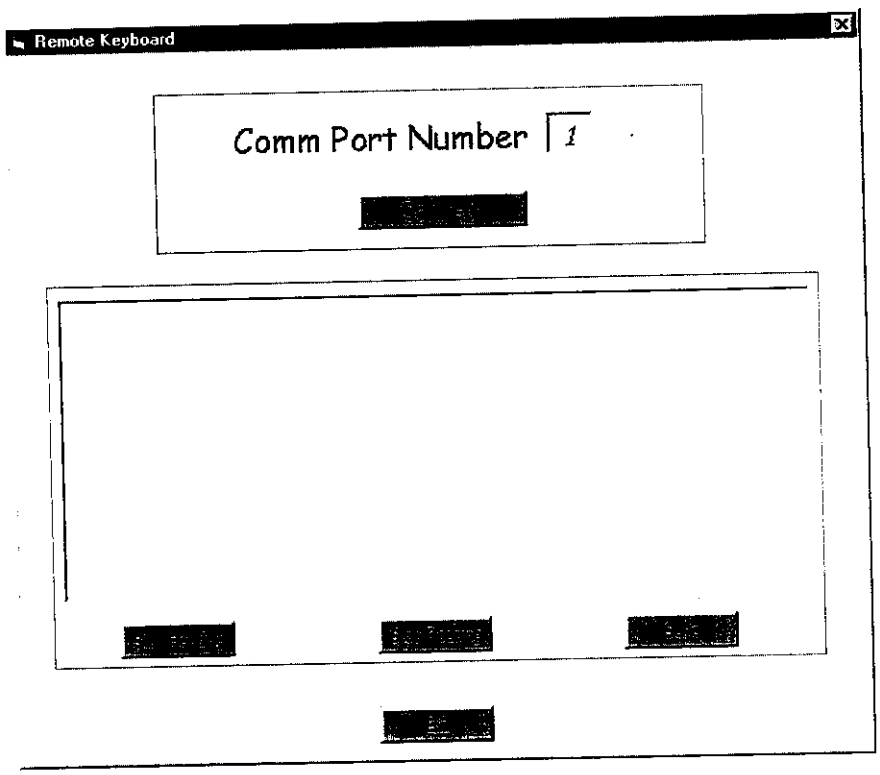
```
eepromread1:

      ;for reading the value in the eeprom
       LCALL I2CStart
       mov A,#00000101B ; control address(constant)
       LCALL I2CSend
       mov A,#10000000B ; address in the eeprom (reverse order)
       LCALL I2CSend
       ;lcall I2CStop

        nop
        nop
        nop
        nop



        ;LCALL I2CStart
        ;LCALL I2CStart
        mov A,#10000101B ; control address
        LCALL I2CSend

        ;mov A,#10000000B ; address in the eeprom (reverse order i.e 00000001b)
        ;LCALL I2CSend

        clr a
        LCALL I2CGet    ; the value in th address will be retrived in accumulator (reverse order)
        lcall I2CStop

                ret



  I2CStart:
        setb p1.7 .; p1.7 - SDA
        clr p1.6  ;p1.6 - SCL
        setb p1.6

        clr p1.7

        clr p1.6

        ret

  I2CStop:
        setb p1.6
        setb p1.7

        ret

  I2CSend:
        mov R0,#8
  I2CS_Loop:
        rrc A
        mov p1.7,C
```

```
    setb p1.6
    clr p1.6

     djnz R0,I2CS_Loop


     setb p1.7
     setb p1.6
     mov C,p1.7
     clr p1.6
     clr p1.7

     ret

I2CGet:
    rrc A
    mov R0,#8
    setb p1.7
    ;clr p1.1
I2CG_Loop:

      setb p1.6
      mov C,p1.7
      rrc A
      clr p1.6
      djnz R0,I2CG_Loop

      mov p1.7,C
      setb p1.6
      clr p1.6
      clr p1.7
      ;mov R1,A
      ;mov a,#5

      ret


init: mov  tmod,#20h
     mov  scon,#58h
     mov  th1,#0e6h
     setb ea
     setb et1
     setb es
     setb tr1
     ret

transmit:
            ;mov   a,r5

     mov   sbuf,temp
here  jnb ti,here
     clr   ti
     ;inc r5
          ret
```

VISUAL BASIC

```
Dim intPortID As Integer ' Ex. 1, 2, 3, 4 for COM1 - COM4
Dim lngStatus As Long
Dim strError  As String
Dim strData   As String


' Initialize Communications
lngStatus = CommOpen(intPortID, "COM" & CStr(intPortID), _
   "baud=9600 parity=N data=8 stop=1")

If lngStatus <> 0 Then
      ' Handle error.
   lngStatus = CommGetError(strError)
      MsgBox "COM Error: " & strError
End If


' Set modem control lines.
lngStatus = CommSetLine(intPortID, LINE_RTS, True)
lngStatus = CommSetLine(intPortID, LINE_DTR, True)

' Write data to serial port.
lngSize = Len(strData)
lngStatus = CommWrite(intPortID, strData)
If lngStatus <> lngSize Then
      ' Handle error.
End If



' Read maximum of 64 bytes from serial port.
lngStatus = CommRead(intPortID, strData, 64)
If lngStatus > 0 Then
   ' Process data.
ElseIf lngStatus < 0 Then
   ' Handle error.
End If

' Reset modem control lines.
lngStatus = CommSetLine(intPortID, LINE_RTS, False)
lngStatus = CommSetLine(intPortID, LINE_DTR, False)



' Close communications.
Call CommClose(intPortID)
```

11.2 SAMPLE OUTPUT SCREEN



S24C02-3

2,048-BIT SERIAL ELECTRICALLY ERASABLE PROM

PRELIMINARY
JULY 1997

FEATURES

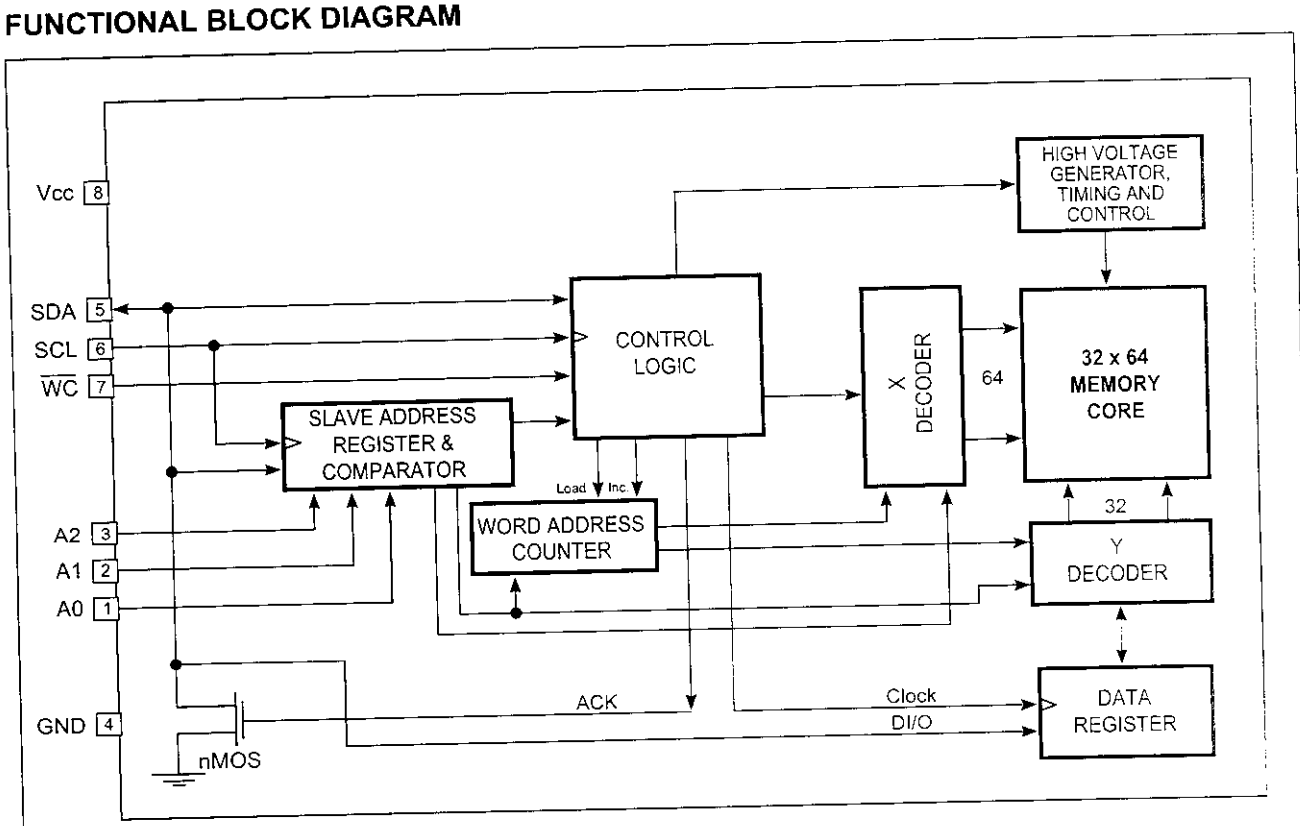
- Low power CMOS
  - — Active current less than 2 mA
  - — Standby current less than 8 µA
- Low voltage operation
  - — 3.0V (Vcc = 2.7V to 5.5V)
- Hardware write protection
  - — Write control pin
- Internally organized as 256 x 8
- Two-wire serial interface
  - — Bidirectional data transfer protocol
- 8-Byte page-write mode
  - — Minimized total write time per byte
- 100 KHz clock speed compatible
- Automatic word address incrementing
  - — Sequential register read
- Self-timed write cycle
  - — Maximum write cycle time of 10 ms
- Endurance: 100,000 cycles per byte
- 8-pin PDIP or SOIC packages
- Filtered inputs for noise suppression

OVERVIEW

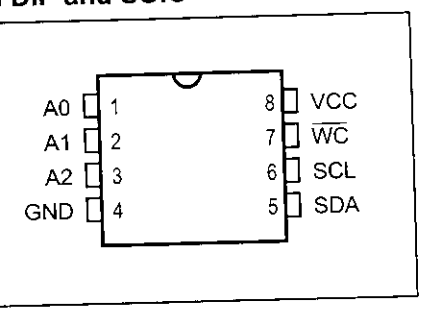
The IS24C02-3 is a low cost 2,048-bit serial EEPROM. It is fabricated using *ISSI*'s advanced CMOS EEPROM technology and operates from a single supply.

The IS24C02-3 is internally organized as a 256 x 8 memory bank. The IS24C02-3 features a serial interface and software protocol allowing operation on a simple 2-wire bus. Up to eight IS24C02-3s may be connected to the 2-wire bus by programming the A0, A1, and A2 inputs.

FUNCTIONAL BLOCK DIAGRAM



CONFIGURATION
DIP and SOIC



PIN DESCRIPTIONS

A0-A2	Address Inputs
SDA	Serial Data I/O
SCL	Serial Clock Input
$\overline{WC}$	Write Control Input
Vcc	Power
GND	Ground

DESCRIPTIONS

al Clock (SCL) - The SCL input is used to clock all data and out of the device. In the WRITE mode, data must ain stable when SCL is HIGH. In the READ mode, data ocked out on the falling edge of SCL.

ial Data (SDA) - The SDA pin is a bidirectional pin used ransfer data into and out of the device. Data may nge only when SCL is LOW. It is an open-drain output, may be wire-ORed with any number of open-drain or n-collector outputs.

A1, and A2 - The address inputs are used to set the st significant three bits of the slave address. These uts may be tied HIGH or LOW, or they may be actively ren. These inputs allow up to eight IS24C02-3 devices e connected together on the bus.

ite Control ($\overline{WC}$) - The Write Control input is used to able any attempt to write to the memory. When HIGH, memory is protected; when LOW, the write function is rmal. The part can be read independent of the state of C pin. When not connected this pin will be pulled LOW.

NDURANCE AND DATA RETENTION

e IS24C02-3 is designed for applications requiring high- durance write cycles and unlimited read cycles. It pro- les 10 years of secure data retention, with or without wer applied, after the execution of 100,000 write cycles.

PPLICATIONS

e IS24C02-3 is ideal for high volume applications requir- g low power and low density storage. This device uses low-cost, space-saving 8-pin plastic package. Candi- te applications include robotics, alarm devices, elec- nic locks, meters and instrumentation.

GENERAL DESCRIPTION

The IS24C02-3 features a SERIAL communication, and supports bidirectional data transmission protocol allowing operation on a simple two-wire bus between the different devices connected somewhere on the system bus. The two-wire bus was defined as a serial data line (SDA), and a serial clock line (SCL). (Refer to Figure 1. Typical System Bus Configuration.)

The protocol defines any device that sends data onto the SDA bus as a transmitter, and the receiving device as a receiver. The device controlling the data transmission is named MASTER device, and the controlled device is named SLAVE device. In all cases, the IS24C02-3 will be a slave device, since it never initiates any data transfers. Up to eight IS24C02-3 can be connected to the bus. Device's physical address inputs A0-A2 must be connected to either Vcc or GND.

Following a START condition, the MASTER (transmitter) device must initiate the "Device Addressing Byte" including device type identifier, device address, and a read or write operation to select a slave device (receiver) connected to the system bus. The receiver will then respond with an ACKnowledge by pulled the SDA line LOW.

The ACKnowledge is used to indicate successful data transfers. The transmitting device will release the data bus (SDA goes HIGH) after transmitting eight bits (one data bit is transfered at the falling edge of each clock cycle). During the ninth clock cycle, the receiver will pull the SDA line LOW to ACKnowledge the transmitter that it received the eight bits of data. (Refer to Figure 2. ACKnowledge Response from Receiver Diagram.)

EVICE OPERATION

TART and STOP Conditions

oth SDA and SCL lines remain HIGH when the SDA bus not busy. A HIGH-to-LOW transition of SDA line, while CL is HIGH, is defined as the START condition. A LOW-)-High transition of SDA line, while SCL is HIGH, is efined as STOP condition. (Refer to Figure 3. Start and top Conditions.)

)ata Validity Protocol

)ne data bit is transferred during each clock cycle. The lata on the SDA line must remain stable during the HIGH)eriod of the clock cycle, because changes on SDA line luring the SCL HIGH period will be interpreted as START)r STOP control signals. (Refer to Figure 4. Data Validity Protocol.)

)evice Addressing Byte Definitions

The most significant four bits of Device Addressing Byte (Bit 7 to Bit 4) are defined as the device type identifier. For IS24C02-3, this is fixed as 1010. The next three significant address bits (Bit 3 to Bit 1) address a particular device. Up to eight IS24C02-3 devices can be connected on the bus. These eight addresses are defined by the state of the A0, A1, and A2 inputs. The last bit Bit 0 defines the write or read operation to be performed. When set to "1", a READ operation is selected; when set to "0" a WRITE operation is selected. (Refer to Figure 5. Device Addressing Byte Definitions.)

WRITE OPERATION

Byte Write

For a WRITE operation, the IS24C02-3 requires another 8-bit data word address following the Device Addressing Byte and ACKnowledgement. This data word address provides access to any one of the 256 data words of device's memory array.

Upon receipt of the data word address, the IS24C02-3 responds with an ACKnowledge on SDA, and waits for the next 8-bit data word, then again responding with an ACKnowledge. The master device terminates the Byte Write Operation by generating a STOP condition, afterward the IS24C02-3 begins the internal WRITE cycle to the nonvolatile memory array. Refer to Write Cycle Timing. All inputs are disabled during this write cycle and the device will not response to any requests from the master. (Refer to Figure 6. Write Operation for the Address, ACKnowledge, and Data Transfer Sequence.)

Page Write

The IS24C02-3 is capable of 8-byte page- WRITE operation. A page-WRITE is initiated in the same manner as a byte write, but instead of terminating the internal write cycle after the first data word is transfered, the master device can transmit up to 15 more words. After the receipt of each data word, the IS24C02-3 responds immediately with an ACKnowledge on SDA line, and the four lower order data word address bits are internally incremented by one while the four higher order bits of the data word address remain constant. If the master device should transmit more than 8 words, prior to issuing the STOP condition, the address counter will "roll over," and the previously written data will be overwritten. All inputs are disabled until completion of the internal WRITE cycle. (Refer to Figure 6. Write Operation for the Address, ACKnowledge, and Data Transfer Sequence.)

Acknowledge Polling

Once the internal write cycle has started and the IS24C02-3 inputs are disabled, acknowledge poiling can be initiated. This involves sending a start condition followed by the Device Addressing Byte. The read/write bit is representive of the operation desired. Only if the internal write cycle has been completed will the IS24C02-3 respond with an acknowledge on the SDA bus allowing the read or write sequence to continue.

READ OPERATION

READ operations are initiated in the same manner as WRITE operations, except that the read/write bit of the device addressing byte is set to "1". There are three READ operation options: current address read, random address read and sequential read.

Current Address Read

The IS24C02-3 contains an internal address counter which maintains the address of the last data word accessed, incremented by one. For example, if the previous operation either a read or write operation addressed to the address location n, the internal address counter would address to address location n+1. When the IS24C02-3 receives the Device Addressing Byte with a READ operation (read/write bit set to "1"), it will respond an ACKnowledge and transmit the 8-bit data word stored at address location n+1. If the Current Address READ operation only accesses a single byte of data, the master device terminates the Current Address READ operation by pulling ACKnowledge HIGH (lack of ACKnowledge) indicating the last data word to be read, followed by a STOP condition. (Refer to Figure 7. Current Address Read Diagram.)

om Access Read

om Address READ operation allows the master e to access any memory location in a random fash- his operation involves a two-step process. First, the er device generates a START condition and initiates e Addressing Byte with a WRITE operation (read/ bit sets to "0"), followed by the address of the data the master device is to READ. This procedure stores esired address of data word to the internal address ter of the IS24C02-3.

the data word address ACKnowledge is received by master device, the master device now initiates a *RENT ADDRESS READ* by sending Device Ad- sing Byte with a READ operation (read/write bit sets to The IS24C02-3 responds with an ACKnowledge and smits the eight data bits stored at the address location re the master device is to READ. At this point, the ter device terminates the operation by pulling Knowledge HIGH (lack of ACKnowledge) indicating the data word to be read, followed by a STOP condition. fer to Figure 8. Random Address Read Diagram.)

Sequential Read

Sequential Reads can be initiated as either a Current Address Read or Random Address Read. The first data word is transmitted as with the other byte read modes, the master device now responds with an ACKnowledge indicating that it requires additional data from the IS24C02-3. The IS24C02-3 continues to output data for each ACKnowledge received. the master device terminates the sequential READ operation by pulling ACKnowledge HIGH (lack of ACKnowledge) indicating the last data word to be read, followed by a STOP condition.

The data output is sequential, with the data from address n followed by the date from address n+1, ... etc. The address counter increments by one automatically, allowing the entire memory contents to be serially read during sequential read operation. When the memory address boundry (address 255) is reached, the address counter "rolls over" to address 0, and the IS24C02-3 continues to output data for each ACKnowledge received. (Refer to Figure 9. Sequential Read Operation Starting with a Random Address READ Diagram.)

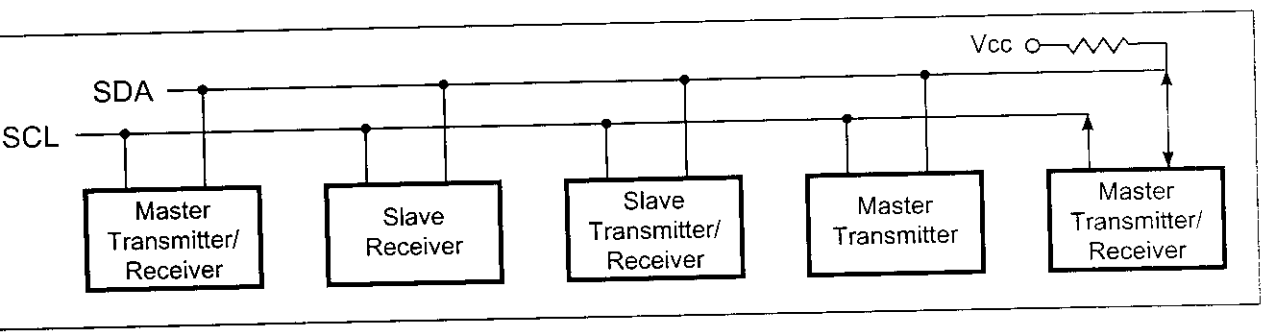


Figure 1. Typical System Bus Configuration

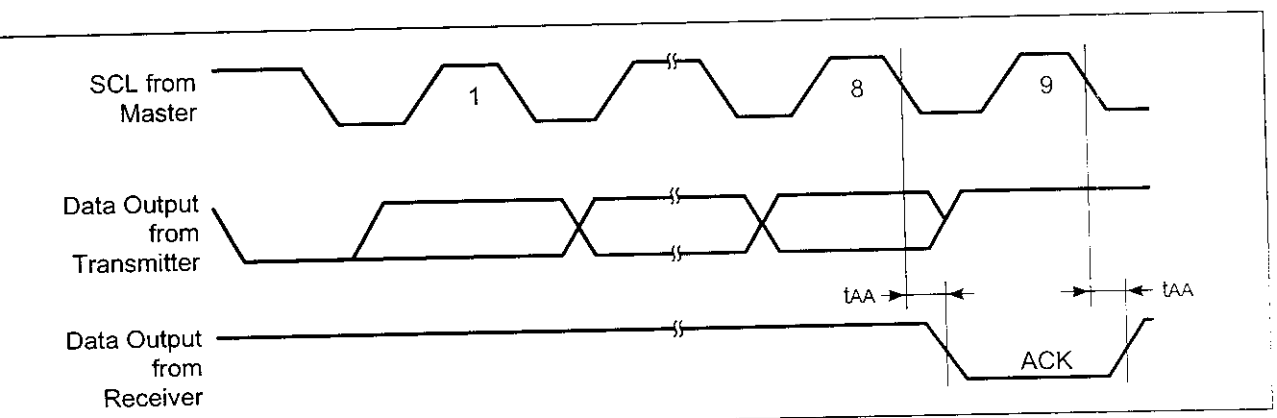


Figure 2. ACKnowledge Response from Receiver

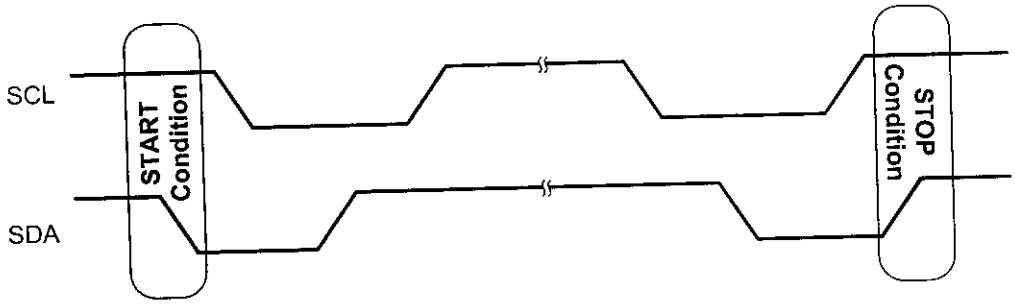


Figure 3. START and STOP Conditions

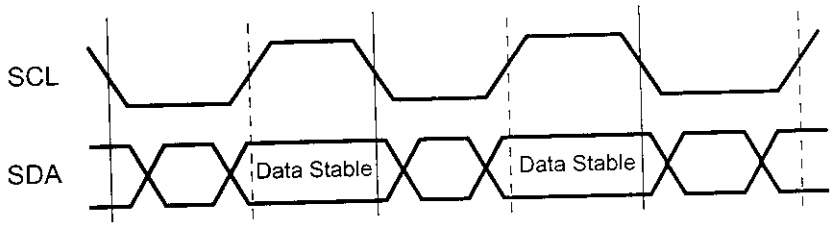


Figure 4. Data Validity Protocol

BIT 7			BIT 4	BIT 3		BIT 1	BIT 0
1	0	1	0	A 2	A 1	A 0	R W

BIT 7 - BIT 4: Are defined as the Device Type Identifier.
For IS24C02, this is fixed as 1010.

BIT 3 - BIT 1: Address a particular device.

BIT 0: Defines the Write or Read Operation to be performec.
1: Read Operation
0: Write Operation

Figure 5. Device Addressing Byte Definitions

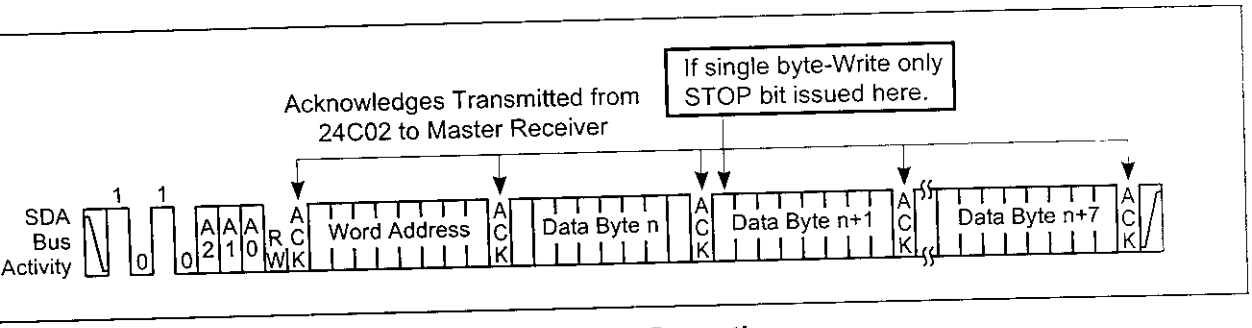


Figure 6. Write Operation

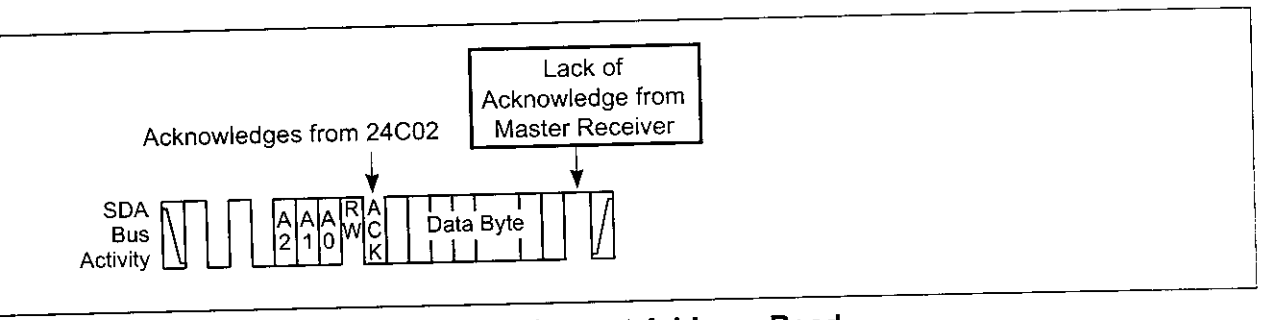


Figure 7. Current Address Read

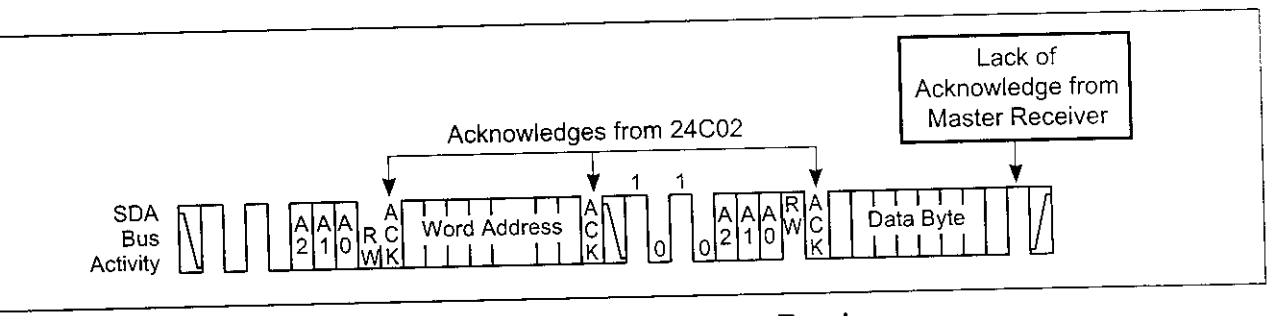


Figure 8. Random Access Read

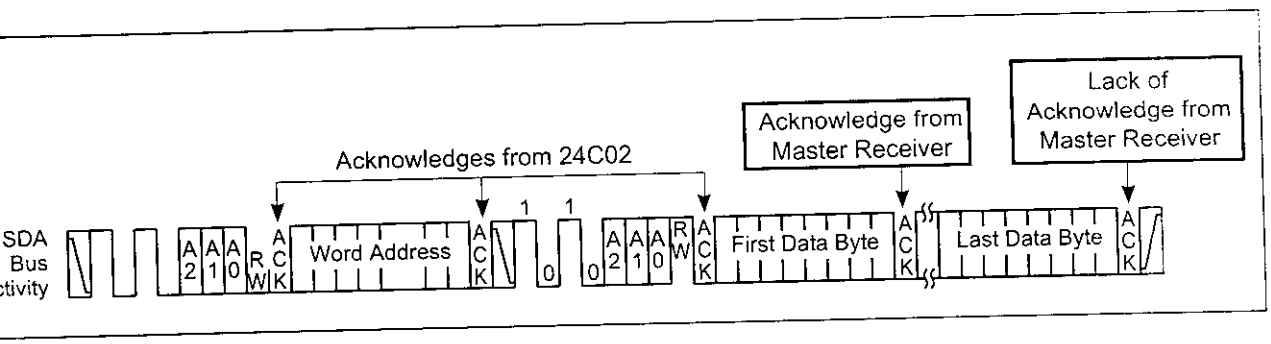


Figure 9. Sequential Read

BSOLUTE MAXIMUM RATINGS(1)

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	0 to +7.0	V
V_P	Voltage on Any Pin	–0.5 to Vcc + 0.5	V
T_{BIAS}	Temperature Under Bias	–40 to +85	°C
T_{STG}	Storage Temperature	–65 to +150	°C
T_{SOL}	Soldering Temperature (less than 10 sec)	300	°C
I_{OUT}	Output Current	5	mA

lotes:
. Stress greater than those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

OPERATING RANGE

Range	Ambient Temperature	Vcc
Commercial	0°C to +70°C	2.7V to 5.5V
Industrial	–40°C to +85°C	2.7V to 5.5V

CAPACITANCE(1,2)

Symbol	Parameter	Conditions	Max.	Unit
C_{IN}	Input Capacitance	V_{IN} = 0V	5	pF
C_{OUT}	Output Capacitance	V_{OUT} = 0V	8	pF

Notes:
1. Tested initially and after any design or process changes that may affect these parameters.
2. Test conditions: T_A = 25°C, f = 1 MHz, Vcc = 5.0V.

LECTRICAL CHARACTERISTICS

°C to +70°C for IS24C02-3 and –40°C to +85°C for IS24C02-3I, Vcc = 2.7V to 5.5V.

mbol	Parameter	Test Conditions	Min.	Max.	Unit
	Output LOW Voltage	Vcc = Min., IOL = 3.0 mA	—	0.4	V
	Input HIGH Voltage	A1-A2, SCL, SDA	0.7 x Vcc	—	V
	Input LOW Voltage(1)	A1-A2, SCL, SDA	—	0.3 x Vcc	V
	Input Leakage	VIN = 0V to Vcc	—	10	µA
	Output Leakage	VOUT = 0V to Vcc	—	10	µA
YS	Input Hysteresis	SCL, SDA	—	0.5 x Vcc	V

WER SUPPLY CHARACTERISTICS

0°C to +70°C for IS24C02-3 and –40°C to +85°C for IS24C02-3I, Vcc = 2.7V to 5.5V.

			IS24C02-3		IS24C02-3I		
bol	Parameter	Test Conditions	Min.	Max.	Min.	Max.	Unit
	Vcc Operating Supply Current	SCL = CMOS Levels at 100 KHz SDA = Open All Other Inputs = GND or Vcc	—	2	—	2	mA
	Standby Current CMOS	SCL = SDA = Vcc = 2.7V to 5.5V All Other Inputs = GND or Vcc	—	8	—	8	µA

ELECTRICAL CHARACTERISTICS

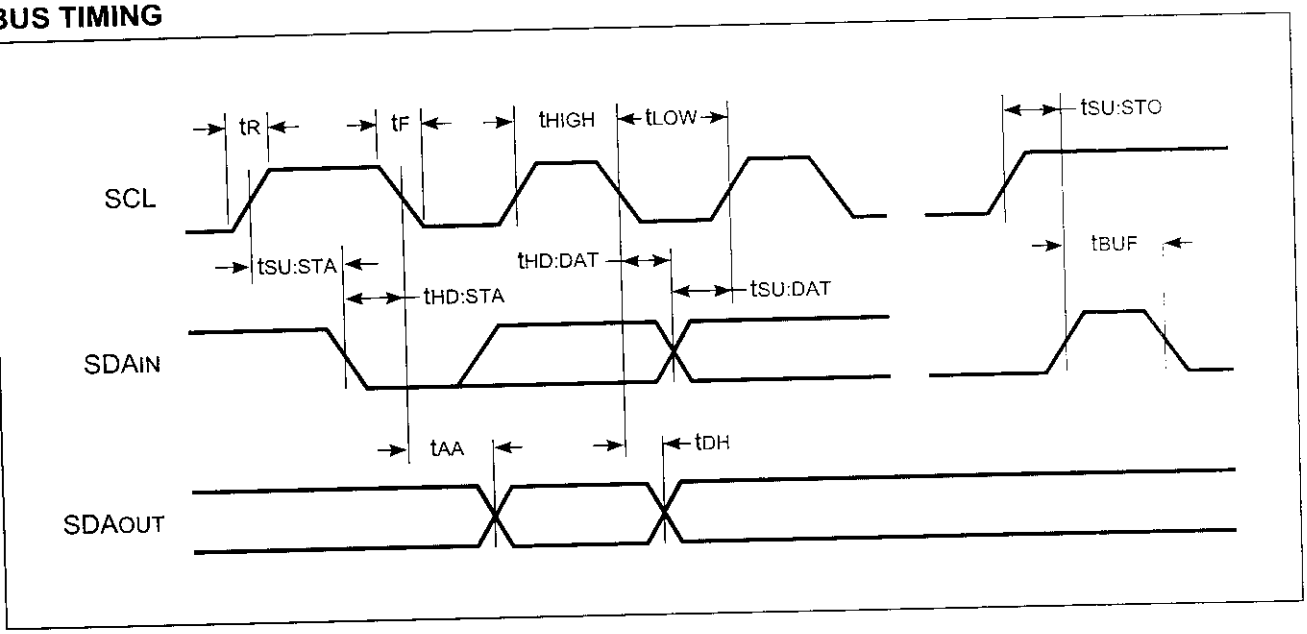
= 0°C to +70°C for IS24C02-3 and –40°C to +85°C for IS24C02-3I, Vcc = 2.7V to 5.5V.

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
SCL	SCL Clock Frequency		0	100	KHz
LOW	Clock LOW Period		4.7	—	µs
HIGH	Clock HIGH Period		4	—	µs
BUF	Bus Free Time	Before New Transmission	4.7	—	µs
SU:STA	Start Condition Setup Time		4.7	—	µs
SU:STO	Stop Condition Setup Time		4.7	—	µs
HD:STA	Start Condition Hold Time		4	—	µs
HD:STO	Stop Condition Hold Time		4	—	µs
SU:DAT	Data In Setup Time		250	—	ns
HD:DAT	Data In Hold Time		0	—	ns
HD	Data Out Hold Time	SCL LOW to SDA Data Out Change	0	—	ns
AA	Clock to Output	SCL LOW to SDA Data Out Valid	0.3	3.5	µs
R	SCL and SDA Rise Time		—	1	µs
F	SCL and SDA Fall Time		—	300	ns
WR	Write Cycle Time		—	10	ms
I	Noise Spike Width(1)	Time Constant at SCL, SDA Inputs	—	100	ns

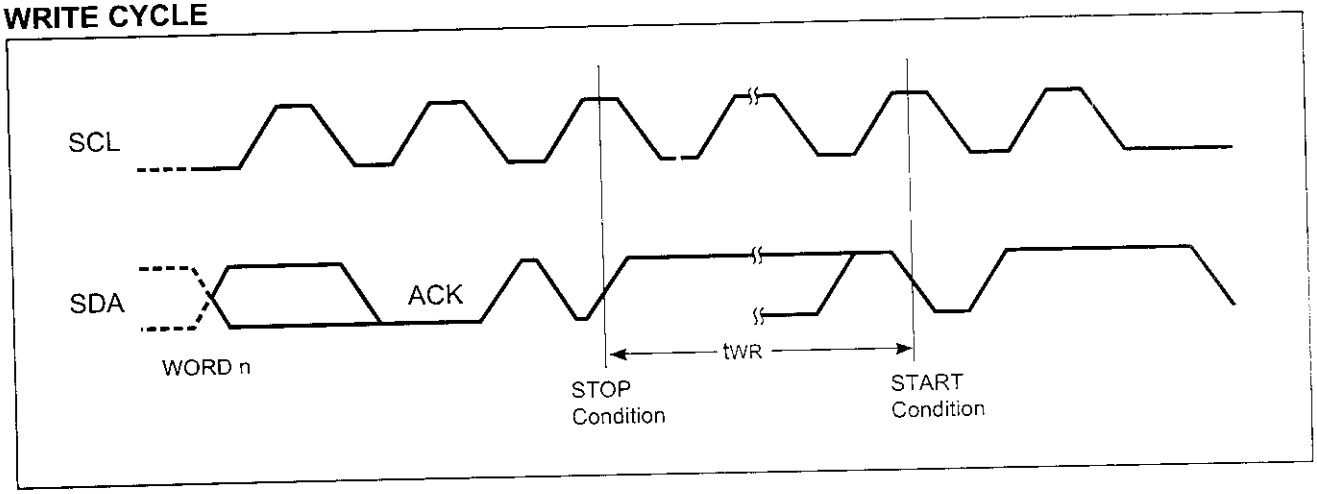
tes:
This parameter is sampled and not 100% tested.

AC WAVEFORMS

BUS TIMING



WRITE CYCLE



ORDERING INFORMATION

Commercial Range: 0°C to +70°C

Frequency	Order Part Number	Package
100 KHz	IS24C02-3P	300-mil Plastic DIP
100 KHz	IS24C02-3G	Small Outline (JEDEC STD)

ORDERING INFORMATION

Industrial Range: –40°C to +85°C

Frequency	Order Part Number	Package
100 KHz	IS24C02-3PI	300-mil Plastic DIP
100 KHz	IS24C02-3GI	Small Outline (JEDEC STD)

Integrated Silicon Solution, Inc.
2231 Lawson Lane
Santa Clara, CA 95054
Fax: (408) 588-0806
Toll Free: 1-800-379-4774
e-mail: sales@issiusa.com
http://www.issiusa.com

tures

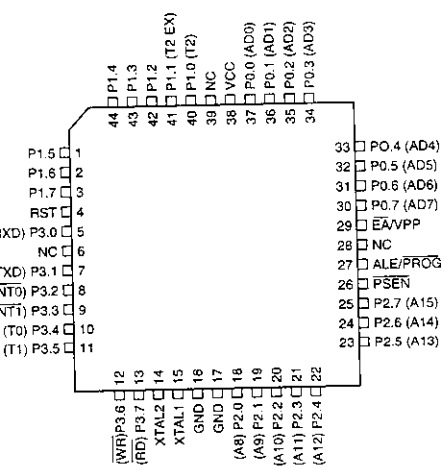
- npatible with MCS-51™ Products
- Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- ly Static Operation: 0 Hz to 24 MHz
- ee-level Program Memory Lock
- x 8-bit Internal RAM
- Programmable I/O Lines
- o 16-bit Timer/Counters
- Interrupt Sources
- ogrammable Serial Channel
- w-power Idle and Power-down Modes

scription

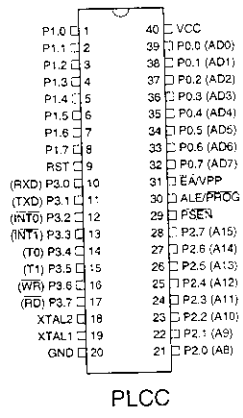
AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K s of Flash programmable and erasable read only memory (PEROM). The device anufactured using Atmel's high-density nonvolatile memory technology and is patible with the industry-standard MCS-51 instruction set and pinout. The on-chip h allows the program memory to be reprogrammed in-system or by a conven- al nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides ghly-flexible and cost-effective solution to many embedded control applications.

n Configurations

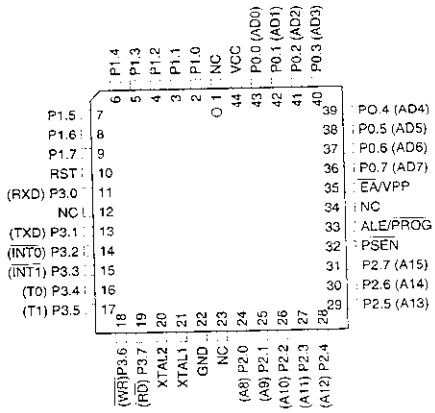
PQFP/TQFP



PDIP



PLCC

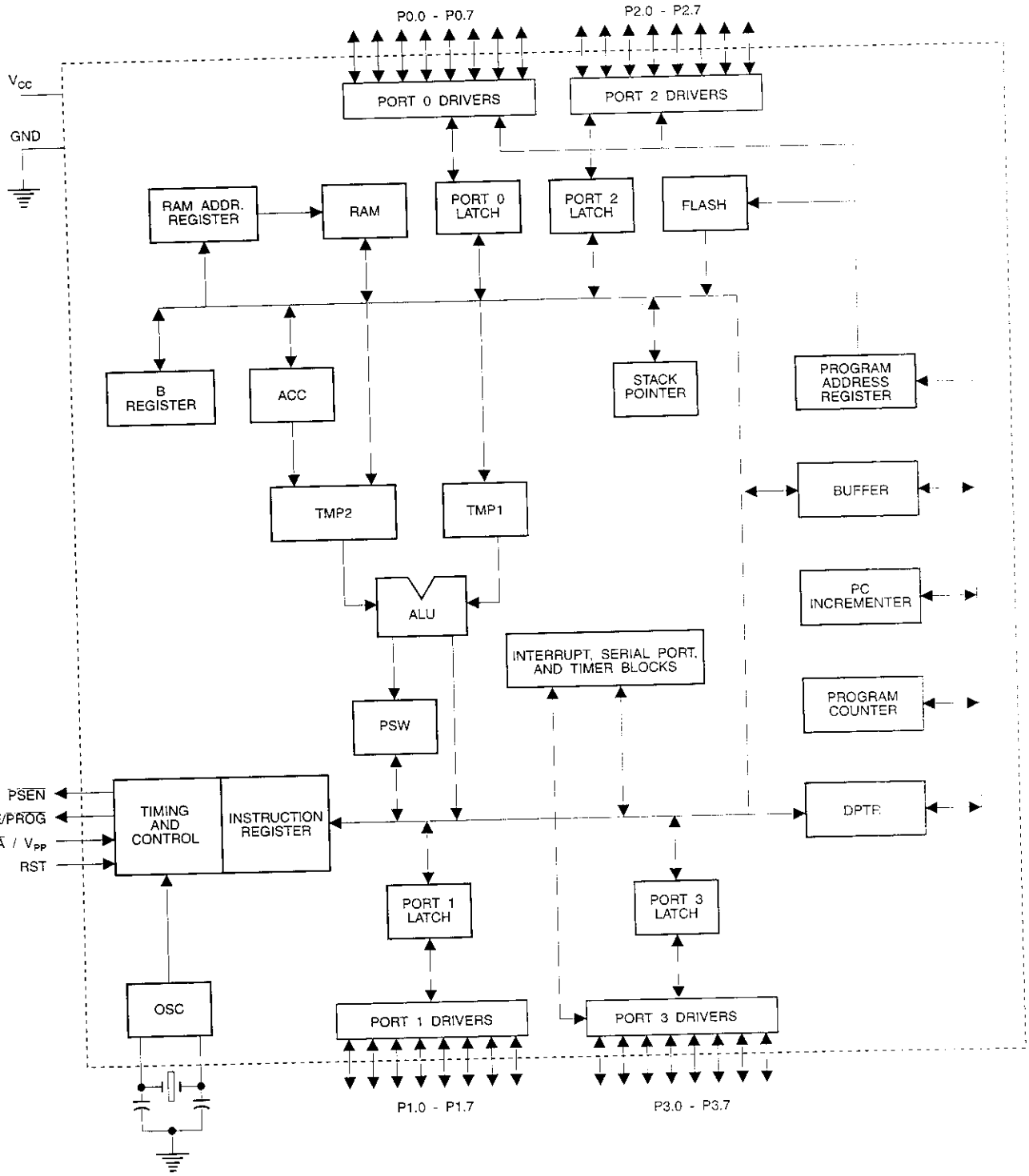


8-bit Microcontroller with 4K Bytes Flash

AT89C51

Rev. 0265G-02/00

ck Diagram


P0.0 - P0.7
P2.0 - P2.7
V_{CC}
GND
PORT 0 DRIVERS
PORT 2 DRIVERS
RAM ADDR. REGISTER
RAM
PORT 0 LATCH
PORT 2 LATCH
FLASH
B REGISTER
ACC
STACK POINTER
PROGRAM ADDRESS REGISTER
BUFFER
TMP2
TMP1
ALU
PC INCREMENTER
INTERRUPT, SERIAL PORT, AND TIMER BLOCKS
PROGRAM COUNTER
PSW
PSEN
/PROG
A / V_{PP}
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DPTR
PORT 1 LATCH
PORT 3 LATCH
OSC
PORT 1 DRIVERS
PORT 3 DRIVERS
P1.0 - P1.7
P3.0 - P3.7

AT89C51 provides the following standard features: 4K of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit /counters, a five vector two-level interrupt architecture, duplex serial port, on-chip oscillator and clock cir-. In addition, the AT89C51 is designed with static logic peration down to zero frequency and supports two /are selectable power saving modes. The Idle Mode s the CPU while allowing the RAM, timer/counters, l port and interrupt system to continue functioning. The er-down Mode saves the RAM contents but freezes scillator disabling all other chip functions until the next ware reset.

Description

C

ply voltage.

D

und.

t 0

t 0 is an 8-bit open-drain bi-directional I/O port. As an out port, each pin can sink eight TTL inputs. When 1s written to port 0 pins, the pins can be used as high-edance inputs.

t 0 may also be configured to be the multiplexed low-er address/data bus during accesses to external pro-m and data memory. In this mode P0 has internal ups.

t 0 also receives the code bytes during Flash program-ng, and outputs the code bytes during program ification. External pullups are required during program ification.

rt 1

rt 1 is an 8-bit bi-directional I/O port with internal pullups. e Port 1 output buffers can sink/source four TTL inputs. hen 1s are written to Port 1 pins they are pulled high by e internal pullups and can be used as inputs. As inputs, rt 1 pins that are externally being pulled low will source rrent (I_{IL}) because of the internal pullups.

rt 1 also receives the low-order address bytes during ash programming and verification.

rt 2

rt 2 is an 8-bit bi-directional I/O port with internal pullups. he Port 2 output buffers can sink/source four TTL inputs. hen 1s are written to Port 2 pins they are pulled high by e internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

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

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and verification.

RST

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

ALE/$\overline{PROG}$

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE

e is skipped during each access to external Data ory.

ired, ALE operation can be disabled by setting bit 0 of location 8EH. With the bit set, ALE is active only dur- MOVX or MOVC instruction. Otherwise, the pin is kly pulled high. Setting the ALE-disable bit has no t if the microcontroller is in external execution mode.

N

ram Store Enable is the read strobe to external pro- n memory.

en the AT89C51 is executing code from external pro- n memory, $\overline{PSEN}$ is activated twice each machine e, except that two $\overline{PSEN}$ activations are skipped during n access to external data memory.

VPP

ernal Access Enable. $\overline{EA}$ must be strapped to GND in er to enable the device to fetch code from external pro- m memory locations starting at 0000H up to FFFFH. e, however, that if lock bit 1 is programmed, $\overline{EA}$ will be rnally latched on reset.

should be strapped to V_{CC} for internal program cutions.

s pin also receives the 12-volt programming enable volt- e (V_{PP}) during Flash programming, for parts that require volt V_{PP}.

AL1

ut to the inverting oscillator amplifier and input to the ernal clock operating circuit.

AL2

tput from the inverting oscillator amplifier.

scillator Characteristics

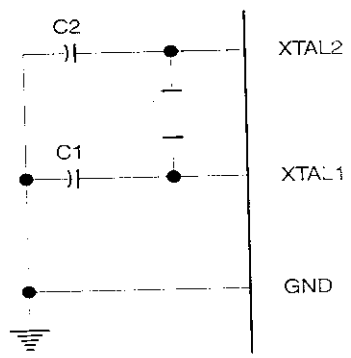
TAL1 and XTAL2 are the input and output, respectively, an inverting amplifier which can be configured for use as on-chip oscillator, as shown in Figure 1. Either a quartz ystal or ceramic resonator may be used. To drive the evice from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Figure 1. Oscillator Connections

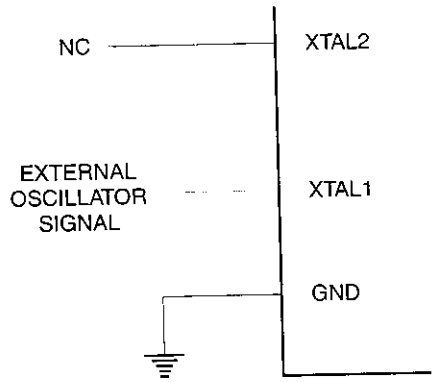


Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

tatus of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
dle	Internal	1	1	Data	Data	Data	Data
dle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

re 2. External Clock Drive Configuration



wer-down Mode

ne power-down mode, the oscillator is stopped, and the ruction that invokes power-down is the last instruction cuted. The on-chip RAM and Special Function Registers retain their values until the power-down mode is terminated. The only exit from power-down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below.

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

ck Bit Protection Modes

	Program Lock Bits			
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash is disabled
3	P	P	U	Same as mode 2, also verify is disabled
4	P	P	P	Same as mode 3, also external execution is disabled

gramming the Flash

AT89C51 is normally shipped with the on-chip Flash ory array in the erased state (that is, contents = FFH) ready to be programmed. The programming interface pts either a high-voltage (12-volt) or a low-voltage) program enable signal. The low-voltage program- g mode provides a convenient way to program the 9C51 inside the user's system, while the high-voltage ramming mode is compatible with conventional third- y Flash or EPROM programmers.

AT89C51 is shipped with either the high-voltage or voltage programming mode enabled. The respective side marking and device signature codes are listed in following table.

	V_{PP} = 12V	V_{PP} = 5V
o-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
gnature	(030H) = 1EH (031H) = 51H (032H) =F FH	(030H) = 1EH (031H) = 51H (032H) = 05H

e AT89C51 code memory array is programmed byte-by- e in either programming mode. *To program any non- nk byte in the on-chip Flash Memory, the entire memory st be erased using the Chip Erase Mode.*

ogramming Algorithm: Before programming the 89C51, the address, data and control signals should be up according to the Flash programming mode table and ure 3 and Figure 4. To program the AT89C51, take the owing steps.

Input the desired memory location on the address lines.

Input the appropriate data byte on the data lines.

Activate the correct combination of control signals.

Raise $\overline{EA}/V_{PP}$ to 12V for the high-voltage programming mode.

Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms.

Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

Data Polling: The AT89C51 features Data Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. Data Polling may begin any time after a write cycle has been initiated.

Ready/Busy: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 51H indicates 89C51
(032H) = FFH indicates 12V programming
(032H) = 05H indicates 5V programming

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

sh Programming Modes

ıe		RST	PSEN	ALE/PROG	EA/V_{PP}	P2.6	P2.7	P3.6	P3.7
e Code Data		H	L		H/12V	L	H	H	H
ıd Code Data		H	L	H	H	L	L	H	H
te Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
ip Erase		H	L	(1)	H/12V	H	L	L	L
ead Signature Byte		H	L	H	H	L	L	L	L

e: 1. Chip Erase requires a 10 ms PROG pulse.

ure 3. Programming the Flash

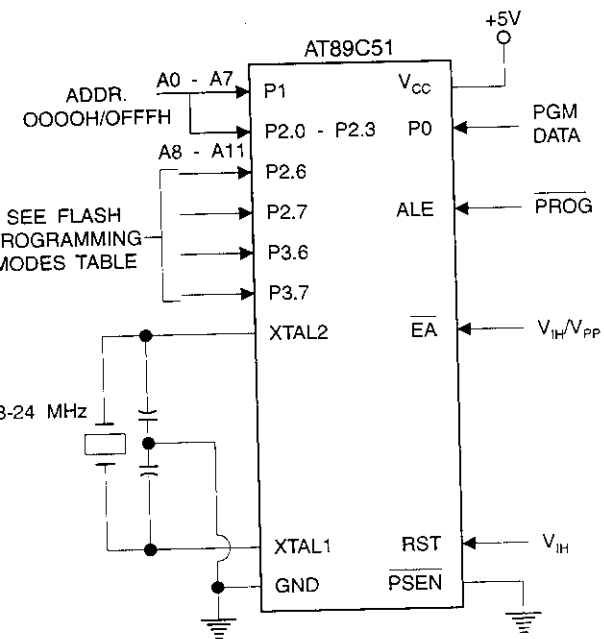
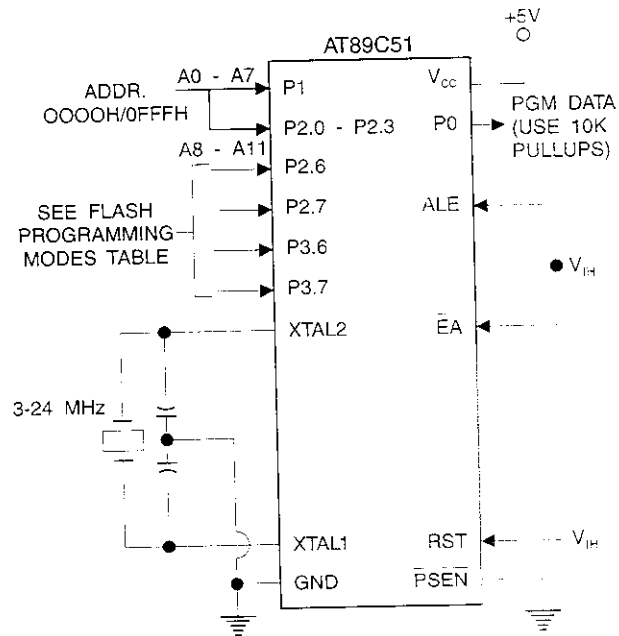
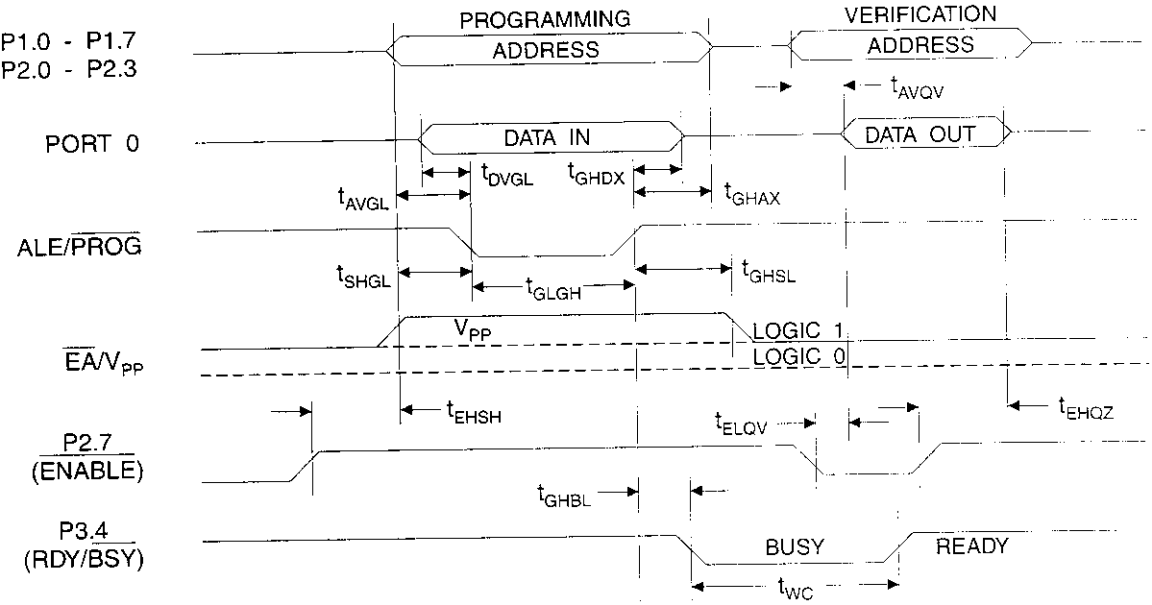


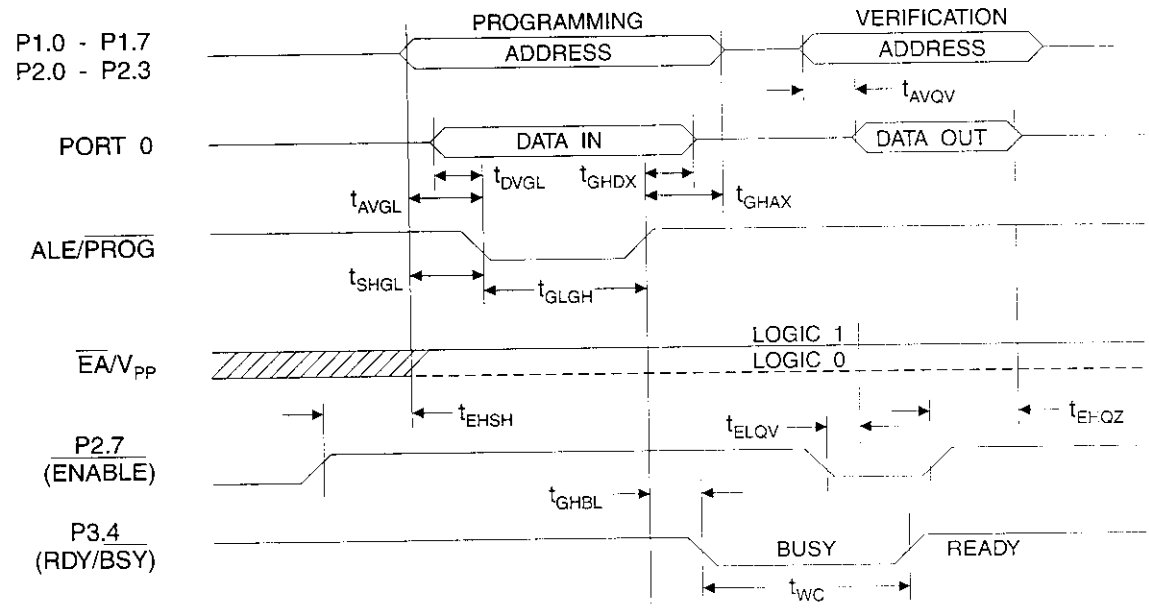
Figure 4. Verifying the Flash



sh Programming and Verification Waveforms - High-voltage Mode (V_{PP} = 12V)



ash Programming and Verification Waveforms - Low-voltage Mode (V_{PP} = 5V)



sh Programming and Verification Characteristics

)°C to 70°C, V_{CC} = 5.0 ± 10%

bol	Parameter	Min	Max	Units
)	Programming Enable Voltage	11.5	12.5	V
)	Programming Enable Current		1.0	mA
LCL	Oscillator Frequency	3	24	MHz
L	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
AX	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
DL	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
DX	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
SH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
GL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
SL(1)	V_{PP} Hold After $\overline{PROG}$	10		µs
GH	$\overline{PROG}$ Width	1	110	µs
AV	Address to Data Valid		$48t_{CLCL}$	
QV	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
HQZ	Data Float After $\overline{ENABLE}$	0	$48t_{CLCL}$	
HBL	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
WC	Byte Write Cycle Time		2.0	ms

te: 1. Only used in 12-volt programming mode.

solute Maximum Ratings*

rating Temperature	-55°C to +125°C
age Temperature	-65°C to +150°C
age on Any Pin Respect to Ground	-1.0V to +7.0V
ximum Operating Voltage	6.6V
Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Characteristics

-40°C to 85°C, V_{CC} = 5.0V ± 20% (unless otherwise noted)

mbol	Parameter	Condition	Min	Max	Units
	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
	Input Low-voltage ($\overline{EA}$)		-0.5	0.2 V_{CC} - 0.3	V
	Input High-voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
1	Input High-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
L	Output Low-voltage[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
L1	Output Low-voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
H	Output High-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
H1	Output High-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
-	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, VCC = 5V ± 10%		-650	µA
	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RST	Reset Pull-down Resistor		50	300	KΩ
IO	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
CC	Power-down Mode[2]	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

tes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum I_{OL} per 8-bit port: Port 0: 26 mA
   Ports 1, 2, 3: 15 mA
   Maximum total I_{OL} for all output pins: 71 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V.

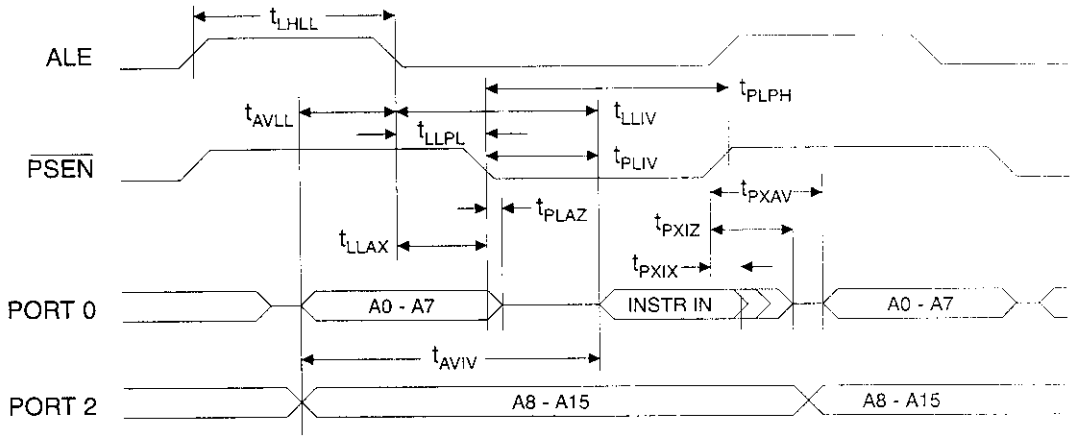
Characteristics

er operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other uts = 80 pF.

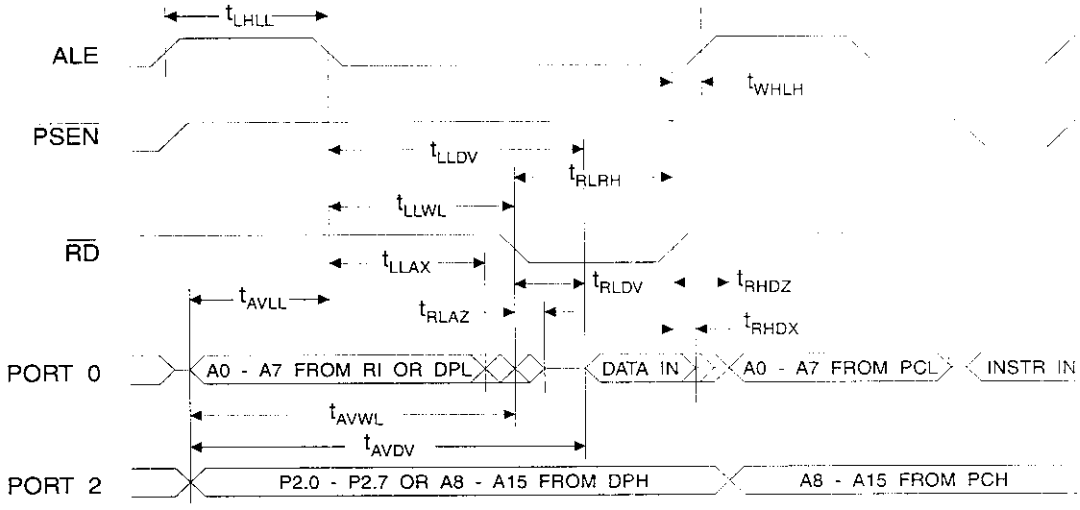
ternal Program and Data Memory Characteristics

mbol	Parameter	12 MHz Oscillator		16 to 24 MHz Oscillator		Units
		Min	Max	Min	Max	
CLCL	Oscillator Frequency			0	24	MHz
LL	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
LL	Address Valid to ALE Low	43		t_{CLCL}-13		ns
AX	Address Hold After ALE Low	48		t_{CLCL}-20		ns
IV	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
PL	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-13		ns
PH	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-20		ns
IV	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-45	ns
IX	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
IZ	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-10	ns
AV	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
IV	Address to Valid Instruction In		312		$5t_{CLCL}$-55	ns
AZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
LRH	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
LWH	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
LDV	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
HDX	Data Hold After $\overline{RD}$	0		0		ns
HDZ	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
LDV	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
AVDV	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
LWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
AVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
QVWX	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-20		ns
QVWH	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-120		ns
WHQX	Data Hold After $\overline{WR}$	33		t_{CLCL}-20		ns
RLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
WHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-20	t_{CLCL}+25	ns

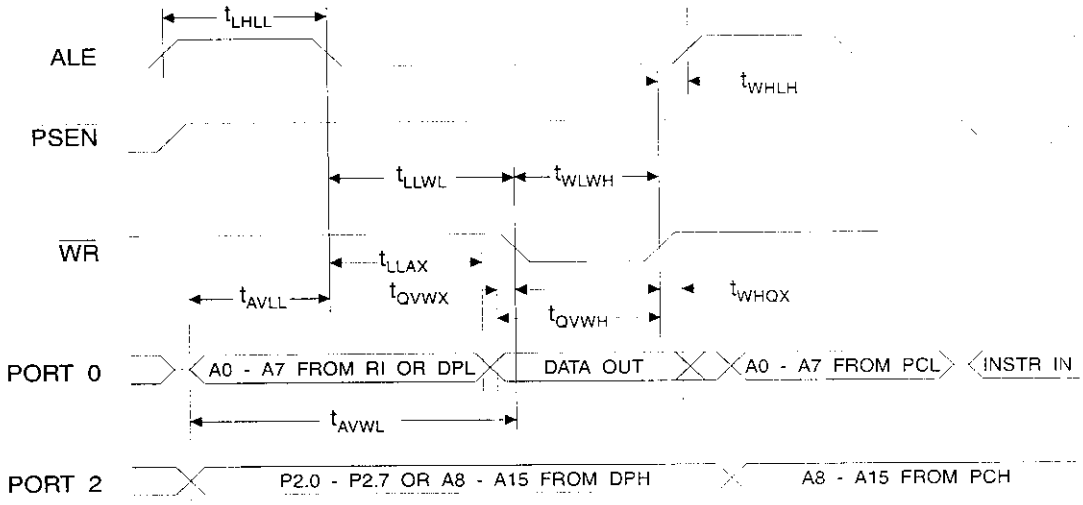
ernal Program Memory Read Cycle



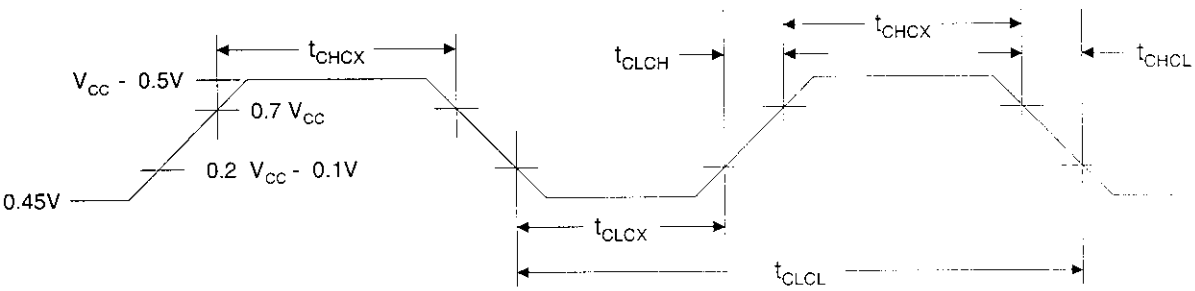
ernal Data Memory Read Cycle



ternal Data Memory Write Cycle



ternal Clock Drive Waveforms



xternal Clock Drive

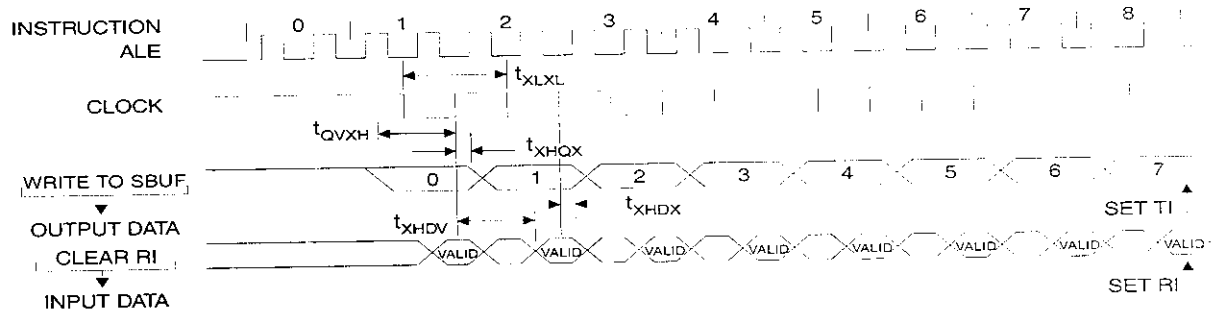
ymbol	Parameter	Min	Max	Units
t_{CLCL}	Oscillator Frequency	0	24	MHz
CL	Clock Period	41.6		ns
HCX	High Time	15		ns
CX	Low Time	15		ns
CH	Rise Time		20	ns
HCL	Fall Time		20	ns

al Port Timing: Shift Register Mode Test Conditions

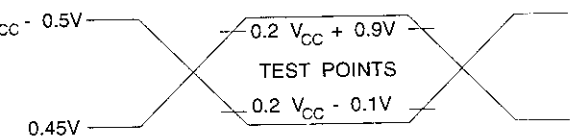
5.0 V ± 20%; Load Capacitance = 80 pF)

bol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
H	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
X	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
X	Input Data Hold After Clock Rising Edge	0		0		ns
V	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

ft Register Mode Timing Waveforms



Testing Input/Output Waveforms[1]



e: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms[1]

Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

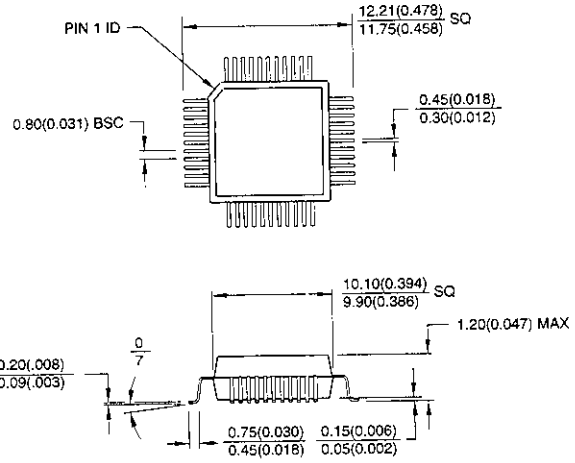
ering Information

eed Hz)	Power Supply	Ordering Code	Package	Operation Range
12	5V ± 20%	AT89C51-12AC	44A	Commercial
		AT89C51-12JC	44J	(0°C to 70°C)
		AT89C51-12PC	40P6	
		AT89C51-12QC	44Q	
		AT89C51-12AI	44A	Industrial
		AT89C51-12JI	44J	(-40°C to 85°C)
		AT89C51-12PI	40P6	
		AT89C51-12QI	44Q	
16	5V ± 20%	AT89C51-16AC	44A	Commercial
		AT89C51-16JC	44J	(0°C to 70°C)
		AT89C51-16PC	40P6	
		AT89C51-16QC	44Q	
		AT89C51-16AI	44A	Industrial
		AT89C51-16JI	44J	(-40°C to 85°C)
		AT89C51-16PI	40P6	
		AT89C51-16QI	44Q	
20	5V ± 20%	AT89C51-20AC	44A	Commercial
		AT89C51-20JC	44J	(0°C to 70°C)
		AT89C51-20PC	40P6	
		AT89C51-20QC	44Q	
		AT89C51-20AI	44A	Industrial
		AT89C51-20JI	44J	(-40°C to 85°C)
		AT89C51-20PI	40P6	
		AT89C51-20QI	44Q	
24	5V ± 20%	AT89C51-24AC	44A	Commercial
		AT89C51-24JC	44J	(0°C to 70°C)
		AT89C51-24PC	40P6	
		AT89C51-24QC	44Q	
		AT89C51-24AI	44A	Industrial
		AT89C51-24JI	44J	(-40°C to 85°C)
		AT89C51-24PI	40P6	
		AT89C51-24QI	44Q	

	Package Type
A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
P6	40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Q	44-lead, Plastic Gull Wing Quad Flatpack (PQFP)

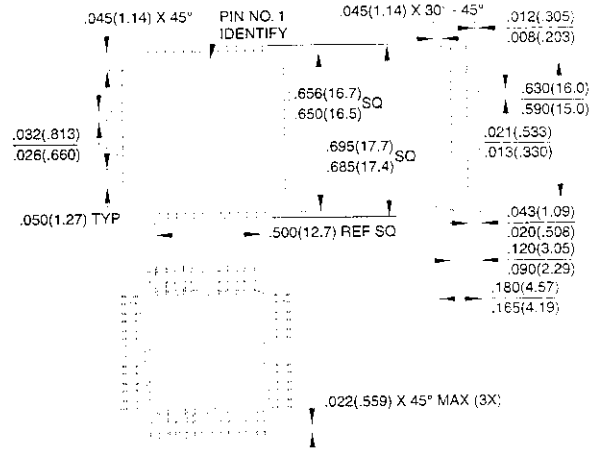
kaging Information

4A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad latpack (TQFP)
imensions in Millimeters and (Inches)*
EDEC STANDARD MS-026 ACB

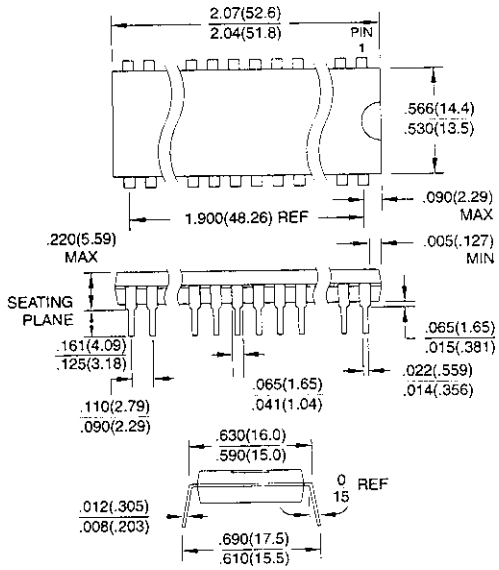


Controlling dimension: millimeters

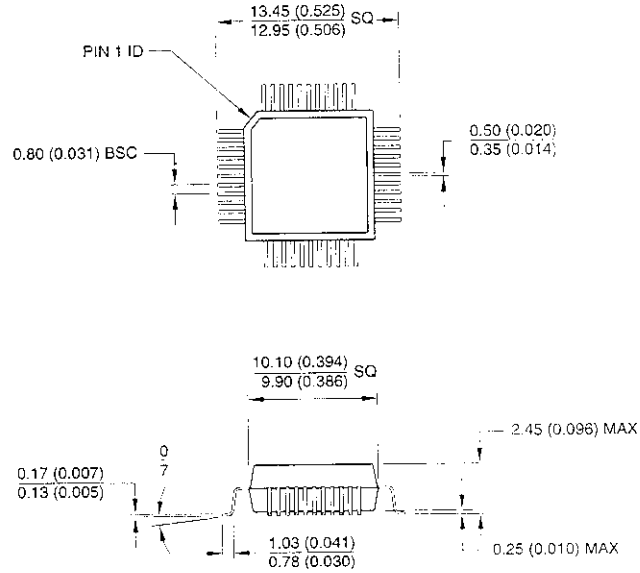
44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC



40P6, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)



44Q, 44-lead, Plastic Quad Flat Package (PQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-022 AB



Controlling dimension: millimeters